Supplementary Materials: Documentation for *elSup.R*

1. Overview of scripts contained in *elSup.R*

After sourcing the file in a folder with a clean slate, objects() (an R-command) should return the following list of 50 new objects.

Table S1. 50 R objects (41 are functions, 4 are matrix constants, 5 are data lists)

area2	area3	c52	c62	c63	c73	cnect	config3	deex	dell
down	eldef	elstat	emplik0	equid	esimp	fig1	fig2	fig3	fig4
fig5	fig6	fig7	fig8	fig9	fig9a	fig10	intrm	isect	isj
isk	isl	lohi	mixw	myginv	ptty	pttz	rquaz	safetest	stdf
tdat1	tdatp1	tictactoe	twop	twopn	W0	W1	W2	W4	W5

The five list objects W0, W1, W2, W4 and W5 represent test data: W0 from Table 4 of the manuscript, W1 and W2 are Type I and Type II subsets of W0, respectively, and W4 and W5 are a 4D added Type II configuration and a quasi-separated data set, respectively, both used later in this supplement (sections 26 and 27). The four constant matrices are c52, c62, c63 and c73. Function dependencies are described hierarchically in the next three tables. The groupings in the next four tables loosely describe the purpose served by the top-level functions.

Table S2. Plotting Function Dependencies

tdat1	tdatp1	fig1	fig2	fig3	fig4	fig5	fig6	fig7	fig8	fig9	fig9a	fig10
tdatp1	lohi	cnect	cnect	lohi	tdatp1	tdatp1		lohi	lohi	tdatp1	tdatp1	cnect
	cnect	ptty	ptty	cnect				cnect	cnect			ptty
	isect	twop	twop	ptty				isect	isect			twopn
	ptty			twop				ptty	ptty			pttz
	twop							twop	twop			

Table S3. Function Dependencies (Grouped by Objective)

Status	Deflate	Identify Minimal Overlapping Configurations								
elstat	eldef	config3								
emplik0	elstat	eldef	isj	isk	isl	area2	area3	c52	c62	c63
deex		elstat	down	down	down					
			area3	c62	c62					
			c63	c73	c73					
			c73							

Table S4. Function Dependencies (Grouped by Objective)

Standard Forms for Type I			Prove Completeness	Make Quasi
equid	intrm	stdf	tictactoe	rquaz
esimp	elstat		config3	elstat
	eldef			
	myginv			

Table S5. Remaining lists and stand-alone functions							
dell	mixw	safetest	W0	W1	W2	W4	W5

2. Four matrix constants $c_{52}, c_{62}, c_{63}, c_{73}$ and five data sets (lists) W0, W1, W2, W4 and W5

The four matrix constants are used in *config*3 to distinguish one configuration among the 38 cataloged from another. The rows of c_{ij} are ordered combinations of i things taken j at a time.

Table S6. Four Matrix Constants

i	*c*52		*c*62		*c*63			*c*73		
1	1	2	1	2	1	2	3	1	2	3
2	1	3	1	3	1	2	4	1	2	4
3	1	4	1	4	1	2	5	1	2	5
4	1	5	1	5	1	2	6	1	2	6
5	2	3	1	6	1	3	4	1	2	7
6	2	4	2	3	1	3	5	1	3	4
7	2	5	2	4	1	3	6	1	3	5
8	3	4	2	5	1	4	5	1	3	6
9	3	5	2	6	1	4	6	1	3	7
10	4	5	3	4	1	5	6	1	4	5
11			3	5	2	3	4	1	4	6
12			3	6	2	3	5	1	4	7
13			4	5	2	3	6	1	5	6
14			4	6	2	4	5	1	5	7
15			5	6	2	4	6	1	6	7
16					2	5	6	2	3	4
17					3	4	5	2	3	5
18					3	4	6	2	3	6
19					3	5	6	2	3	7
20					4	5	6	2	4	5
21								2	4	6
22								2	4	7
23								2	5	6
24								2	5	7
25								2	6	7
26								3	4	5
27								3	4	6
28								3	4	7
29								3	5	6
30								3	5	7
31								3	6	7
32								4	5	6
33								4	5	7
34								4	6	7
35								5	6	7

Table S7. Three data sets (as lists) from Table 4 of the manuscript

W0						W1						W2					
	x			y	r		x			y	r		x			y	r
1	6.7	5.8	4.0	0	1	11	10.4	4.1	3.5	0	1	1	6.7	5.8	4.0	0	1

2	9.5	4.7	7.0	0	2
3	11.9	4.1	6.0	1	3
4	10.1	2.6	3.0	0	4
5	11.2	4.7	8.0	1	5
6	9.0	5.0	5.0	1	6
7	10.0	4.4	9.0	1	7
8	9.0	4.8	10.0	1	8
9	7.5	4.2	2.5	0	9
10	8.4	4.2	3.5	1	10
11	10.4	4.1	3.5	0	11
12	11.0	4.6	8.1	1	12
13	7.0	2.6	4.1	1	13
14	7.3	5.1	2.1	0	14
15	8.4	4.6	2.0	0	15
16	10.0	4.0	11.0	0	16

12	11.0	4.6	8.1	1	2
13	7.0	2.6	4.1	1	3
15	8.4	4.6	2.0	0	4
16	10.0	4.0	11.0	0	5

2	9.5	4.7	7.0	0	2
4	10.1	2.6	3.0	0	3
6	9.0	5.0	5.0	1	4
7	10.0	4.4	9.0	1	5
10	8.4	4.2	3.5	1	6

Table S8. An Add Type II 4D data set and a 4D quasi-separate data set (as lists)

W4						
	x				*y*	*rid*
1	1.2	1.4	1.8	0.0	0	1
2	-1.4	-0.8	-2.0	0.0	0	2
3	0.6	0.0	1.2	0.0	1	3
4	-0.8	0.6	-1.4	0.0	1	4
5	1.0	-1.0	0.0	2.0	1	5
6	1.0	-1.0	0.0	2.0	0	6
7	1.2	-1.1	-0.7	-0.4	1	7
8	1.2	-1.1	-0.7	-0.4	0	8

W5						
	x				*y*	*rid*
1	1.2	1.4	1.8	0.0	0	1
2	-1.4	-0.8	-2.0	0.0	0	2
3	0.6	0.0	1.2	0.0	1	3
4	-0.8	0.6	-1.4	0.0	1	4
5	1.0	-1.0	0.0	2.0	1	5
6	1.0	-1.0	0.0	2.0	0	6
7	1.2	-1.1	-0.7	-0.4	1	7

Notes: Runs 5 and 6 (1 pair) and runs 7 and 8 (a second pair) of $W4\$x$ are two Type I overlap configurations of dimension zero. They are depicted in Figure 2 of the manuscript as configuration *a*. Runs 1 through 4 form an example of the 2D Type I configuration *d* depicted in Figure 2 of the manuscript. $W4$ is a Type II add configuration, which is apparent as runs 1 and 2 are $p_1 \pm (1.3, 1.1, 1.9, 0)$ and runs 3 and 4 are $p_1 \pm (.7, -.3, 1.3, 0)$, respectively, where $p_1 = (-.1, .3, -.1, 0)$. According to the manuscript (section 14, on quasi-separation), if one run is dropped from $W4$'s pair of runs 5 and 6 (or 7 and 8), both overlapping configurations of dimension zero, what remains is an example of a quasi-separate configuration of dimension four. $W5$ is such an example, i.e., it is $W4$ without run number 8. A much larger quasi-separate configurations is built on $W5$ in section 26 using the *rquaz* function and is then analyzed in section 27.

3. $rv = deex(w, r = 0)$

This function computes the displacements δ. Removed rows of $w\$x$ (i.e., predictors) are specified with r for all $1 \le r \le length(y)$. The value(s) of r may differ from the value(s) of $w\$rid[r]$; r may be a vector but usually identifies just one row. The returned values (rv's) are: (1) $rv\$w0$ = the original data set w; (2) $rv\$w1$ = the reduced data set (if $r \neq 0$), otherwise NULL; $rv\$x0$ = the rows for which $y = 0$; $rv\$x1$ = the rows for which $y = 1$; $rv\$dX$ = the displacements δ; $rv\$dij$ = the two rows involved in each displacement; and $rv\$d$ = the vector of distances of δ from the origin.

The important returned value of *deex*() is *rv*\$*dX*, which is δ. δ is the basis for assessing overlap status of the data set *rv*\$*w*0 (or *rv*\$*w*1 when $r \neq 0$).

Table S9. $rv = deex(W2)$, where *rv* (the returned value) has the following components

```
$w0                    $w1                  $dX                        $dij
$w0$x                  NULL                  [1,]  2.3 -0.8  1.0       [1,]    1    1
1   6.7  5.8  4.0                           [2,] -0.5  0.3 -2.0       [2,]    1    2
2   9.5  4.7  7.0      $x1                  [3,] -1.1  2.4  2.0       [3,]    1    3
4  10.1  2.6  3.0      6   9.0  5.0  5.0    [4,]  3.3 -1.4  5.0       [4,]    2    1
6   9.0  5.0  5.0      7  10.0  4.4  9.0    [5,]  0.5 -0.3  2.0       [5,]    2    2
7  10.0  4.4  9.0      10  8.4  4.2  3.5    [6,] -0.1  1.8  6.0       [6,]    2    3
10  8.4  4.2  3.5                          [7,]  1.7 -1.6 -0.5       [7,]    3    1
                       $x0                  [8,] -1.1 -0.5 -3.5       [8,]    3    2
$w0$y                  1  6.7  5.8    4     [9,] -1.7  1.6  0.5       [9,]    3    3
[1] 0 0 0 1 1 1        2  9.5  4.7    7
                       4 10.1  2.6    3

$w0$rid                $d (This is the vector of distances of rows of dX to the origin)
[1] 1 2 3 4 5 6        2.6325 2.0833 3.3121 6.1522 2.0833 6.2650 2.3875 3.7027 2.3875
```

4. $emplik0(z, \mu = 0, \lambda = 0, \varepsilon = 1/nrow(z), M = \infty, thresh = 1e-30, itermax = 100, verbose = F)$

From the comments within the *emplik*0 program:

Table S10. *emplik*0 inputs

```
z,            # matrix with one data vector per row, a column vector is ok when d=1
mu,           # hypothesized mean, default (0 ... 0) in R^d
lam,          # starting lambda, default (0 ... 0)
eps,          # lower cutoff for -log( ), default 1/nrow(z)
M,            # upper cutoff for -log( ), default Inf
thresh=1e-30, # convergence threshold for log likelihood (default is aggressive)
itermax=100,  # upper bound on number of Newton steps (seems ample)
verbose=FALSE)# controls printed output
```

For our purposes, $z = \delta$, i.e., *deex*(*W*), and the default settings are always used. This amounts to performing an empirical likelihood test that the true mean of δ is the zero vector. The actual mean of δ is $\overline{\delta} = \overline{x}_{y=1} - \overline{x}_{y=0}$, which is the same quantity that determines the sign of the slope coefficient in a logistic regression [Manuscript reference (Owen and Roediger, 2014)].

Table S11. The returned values of $rv = emplik0(\ deex(W1\$)dX\ )$

```
$logelr               $wts                    $converged      $ndec
[1] -1.935504         [1,] 0.30014455         [1] TRUE        [1] 1.173486e-21
                      [2,] 0.05813245
$lam                  [3,] 0.39172300         $iter           $gradnorm
[1,]  0.6555483       [4,] 0.09985545         [1] 6           [1] 7.16266e-15
[2,] -1.9213006       [5,] 0.04186755
[3,]  0.0266542       [6,] 0.10827700
```

5. *dell*(*W*)

This function returns one vector: $\bar{\delta} = \overline{W\$x}_{W\$y=1} - \overline{W\$x}_{W\$y=0} = \bar{x}_{y=1} - \bar{x}_{y=0}$

Table S12. The returned value of $rv = dell(W0)$

```
0.9500 -0.0875  2.3250
```

6. $elstat(w, r = 0, thresh = 1e-8, abbr = T)$

For a given list data set w, this function determines the overlap status, "O", "Q", "C", or "NMR" for overlap, quasi-separated, completely-separated or no-mixed-results, respectively. The function accommodates run dropping of the r[th] predictor(s), where r can be a vector of row ID's. Examples are provided for the overlap and quasi-separated cases in the next two tables.

Table S13. $rv = elstat(W2)$, where $W2$ is a Type II subset of Table 4 of the manuscript

```
$w
$w$x
1   6.7  5.8  4.0
2   9.5  4.7  7.0
4  10.1  2.6  3.0
6   9.0  5.0  5.0
7  10.0  4.4  9.0
10  8.4  4.2  3.5

$w$y
[1] 0 0 0 1 1 1

$w$rid
[1] 1 2 3 4 5 6
```

```
$rank
[1] 3

$sum
[1] 1

$converged
[1] TRUE

$overlap
[1] TRUE
```

```
$k1
   6    7   10
0.25 0.25 0.50

$k0
   1    2    4
0.25 0.50 0.25

$min1
[1] 0.25

$min0
[1] 0.25
```

```
$ess
[1] 8.95 4.45 5.25

$eff
[1] 8.95 4.45 5.25

$status
[1] "O"
```

Note. The weights k1 and k0 were renamed u1 and u0 in the submitted manuscript to avoid the impression they were integers, which they are not.

Table S14. $rv = elstat(W5)$ where $W5$ is a quasi-separated data set (from Table S8)

```
w
$w$x
1  1.2  1.4  1.8  0.0
2 -1.4 -0.8 -2.0  0.0
3  0.6  0.0  1.2  0.0
4 -0.8  0.6 -1.4  0.0
5  1.0 -1.0  0.0  2.0
6  1.0 -1.0  0.0  2.0
7  1.2 -1.1 -0.7 -0.4

$w$y
[1] 0 0 1 1 1 0 1

$w$rid
[1] 1 2 3 4 5 6 7
```

```
$rank
[1] 4

$sum
[1] 0.75

$converged
[1] FALSE

$overlap
[1] FALSE

$min1
[1] 4.656613e-10

$min0
[1] 0.25
```

```
$k1
3       4    5                        7
2.500000e-01 2.500000e-01 2.500000e-01 4.656613e-10

$k0
   1     2     6
0.25 0.25 0.25

$ess
[1]  0.20000000 -0.09999999 -0.05000000  0.49999999

$eff
[1]  0.20000001 -0.10000000 -0.04999999  0.50000000

$status
[1] "Q"
```

7. *eldef(w)*

This function deflates an overlapping configuration to either a Type I or a Type II configuration, or deflates a quasi-separated data set to an overlapping substructure ***of lower dimension***. This function is the only one used to determine if the input w represents a minimal overlapping configuration or a minimal quasi-separated configuration.

Table S15. $rv = eldef(W0)$, where $W0$ is the $n = 16$ overlapping data set of Tables S7 and Table 4 of the manuscript

```
eldef(W0)
$x
11 10.4  4.1  3.5
12 11.0  4.6  8.1
13  7.0  2.6  4.1
15  8.4  4.6  2.0
16 10.0  4.0 11.0

$y
[1] 0 1 1 0 0

$rid
[1] 11 12 13 15 16

$status
[1] "O"

$ORIGtype
[1] -1
```

Table S16. $rv = eldef(W5)$, where $W5$ is a quasi-separated data set of Table S8

```
$wq
$wq$x
1  1.2  1.4  1.8    0
2 -1.4 -0.8 -2.0    0
3  0.6  0.0  1.2    0
4 -0.8  0.6 -1.4    0
5  1.0 -1.0  0.0    2
6  1.0 -1.0  0.0    2

$wq$y
[1] 0 0 1 1 1 0

$wq$rid
[1] 1 2 3 4 5 6

$sumq
[1] 1

$rankq
[1] 3

$wc
$wc$x
[1]  1.2 -1.1 -0.7 -0.4

$wc$y
[1] 1

$wc$rid
[1] 1

$status
[1] "Q"

$ORIGtype
[1] -1
```

The returned value rv$wq (above) is a quasi-separated data set, i.e., a lower dimensional overlapping data set, minimal in the sense that removal of any one run further reduces rank. The returned value rv$wc represents the completely-separated portion surrounding the original argument w to *eldef* (in this case W5). When the configuration represented by w is quasi-completely separated, the trimmed value rv$wq has the same quasi status. The trimming is based upon the subset of the u_{1i}'s and u_{0j}'s that have non-negligible magnitudes. The value of *rv$ORIGtype* is described below.

Table S17. Status Assessment with $rv = eldef(w)$

Before Status of w	ORIGtype	After Status of w	Example Data Set
O	-1	Not Minimal	W0
O	1	Type I	W1
O	2	Type II	W2
Q	-1	Not Minimal	W5
Q	0	Minimal	D_3 (Table S41)
C	-1	Complete Separation	D_8 (Table S41)
NMR	-1	No Mixed Results	

8. $equid(d_1, d_0)$

This function returns the equidistant standard form E for a Type I overlapping configuration for the two given dimensions (bottom of pp. 15 of manuscript).

Table S18. $rv = equid(1,2)$

```
$x
[1,]    1  0.0  0.0000000
[2,]   -1  0.0  0.0000000
[3,]    0  1.0  0.0000000
[4,]    0 -0.5  0.8660254
[5,]    0 -0.5 -0.8660254

$y
[1] 1 1 0 0 0

$rid
[1] 1 2 3 4 5

$d1
[1] 1

$d0
[1] 1.0000000 0.8660254

$d
[1] 1.000000 1.000000 1.154701
```

9. $esimp(d)$

This function returns the standard equidistant simplex of dimension d consisting of $d+1$ runs. The values of rv$di reported in the return is the main diagonal of rv$u. These simplexes most closely correspond to Figure 1 of the manuscript.

Table S19. $rv = esimp(2)$

```
$u
[1,]  1.0  0.0000000
[2,] -0.5  0.8660254
[3,] -0.5 -0.8660254

$di
[1] 1.0000000 0.8660254
```

10. *myginv*(*w*$*x*)

This function is a copy of the generalized inverse function, *ginv*, taken from R's MASS library.

Table S20. $rv = myginv(W1\$x)$

	11	12	13	15	16
[1,]	0.3084177	-0.01003055	0.19023901	-0.27535566	-0.1115894
[2,]	-0.5099801	0.03620213	-0.35936169	0.71797454	0.1390115
[3,]	-0.1105202	0.03067476	-0.03578853	-0.01606764	0.1197476

Table S21. $rv = myginv(W2\$x)$

	1	2	4	6	7	10
[1,]	-0.06640495	-0.01179539	0.14509815	0.003125785	-0.02671656	0.03934660
[2,]	0.23098245	-0.01591470	-0.13846747	0.068140285	-0.09996968	0.04625749
[3,]	-0.05176625	0.06593527	-0.09862016	-0.029429986	0.16130053	-0.07519321

11. *intrm*(*w*)

This function computes the interim standard form $V = \begin{pmatrix} V^1 \\ V^0 \end{pmatrix}$ as described in the numerical example presented in Table 8 of the manuscript. Interim standards appear as rv$v in the two tables below. The function works for both Type I and Type II configurations and the returned values are created from *myginv*.

Table S22. $rv = intrm(W1)$, where $W1$ is a Table 4 (manuscript) Type I sub-configuration

```
$w
$w$x
11 10.4  4.1  3.5
12 11.0  4.6  8.1
13  7.0  2.6  4.1
15  8.4  4.6  2.0
16 10.0  4.0 11.0

$w$y
[1] 0 1 1 0 0

$w$rid
[1] 1 2 3 4 5

$ww
$ww$x
12 11.0  4.6  8.1
13  7.0  2.6  4.1
11 10.4  4.1  3.5
15  8.4  4.6  2.0
16 10.0  4.0 11.0

$ww$y
```

```
$v
[1,]  0.75  0.375  0.75
[2,] -0.75 -0.375 -0.75
[3,]  0.16  0.000 -1.44
[4,] -0.16  0.050 -0.51
[5,]  0.00 -0.050  1.95

$vm
[1,]  0.75 0.375  0.75
[2,]  0.16 0.000 -1.44
[3,] -0.16 0.050 -0.51

$g
[1,] 0.40677966  1.2923729 -3.0508475
[2,] 1.76271186 -1.4830508  6.7796610
[3,] 0.04519774 -0.5508475 -0.3389831

$l
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

$r
```

```
[1] 1 1 0 0 0

$ww$rid
[1] 2 3 1 4 5
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Table S23. $rv = intrm(W2)$, where $W2$ is a Table 4 (manuscript) Type II sub-configuration

```
$w
$w$x
1   6.7  5.8  4.0
2   9.5  4.7  7.0
4  10.1  2.6  3.0
6   9.0  5.0  5.0
7  10.0  4.4  9.0
10  8.4  4.2  3.5

$w$y
[1] 0 0 0 1 1 1

$w$rid
[1] 1 2 3 4 5 6

$ww
$ww$x
6   9.0  5.0  5.0
7  10.0  4.4  9.0
10  8.4  4.2  3.5
1   6.7  5.8  4.0
2   9.5  4.7  7.0
4  10.1  2.6  3.0

$ww$y
[1] 1 1 1 0 0 0

$ww$rid
[1] 4 5 6 1 2 3
```

```
$v
[1,]  0.0125  0.1375 -0.0625
[2,]  0.2625 -0.0125  0.9375
[3,] -0.2750 -0.1250 -0.8750
[4,] -0.5625  0.3375 -0.3125
[5,]  0.2750  0.1250  0.8750
[6,]  0.2875 -0.4625 -0.5625

$vm
[1,]  0.0125  0.1375 -0.0625
[2,]  0.2625 -0.0125  0.9375
[3,] -0.5625  0.3375 -0.3125
[4,]  0.2750  0.1250  0.8750

$g
[1,]  3.070078 -1.690768 -1.8242492 1.379310
[2,]  4.041528 -1.742677  0.4004449 2.298851
[3,] -1.161290  1.161290  0.5161290 0.000000

$l
[1,]    1    0  0e+00
[2,]    0    1 -1e-08
[3,]    0    0  1e+00

$r
[1,]  0.6666667 -0.3333333    0 0.3333333
[2,] -0.3333333  0.6666667    0 0.3333333
[3,]  0.0000000  0.0000000    1 0.0000000
[4,]  0.3333333  0.3333333    0 0.6666667
```

12. $stdf(d_1, d_0)$

This function returns the standard form $\Lambda^{(d_1,d_0)} = \begin{pmatrix} \Lambda^{d_1} \\ \Lambda^{d_0} \end{pmatrix}$ as defined in Tables 7 and 8 of the manuscript.

Table S24. $rv = stdf(1,2)$

```
$x
[1,]    1    0    0
[2,]   -1    0    0
[3,]    0    1    0
[4,]    0    0    1
[5,]    0   -1   -1

$y
[1] 1 1 0 0 0

$rid
[1] 1 2 3 4 5
```

13. $tdat1(ic, ipl = T)$, $1 \le ic \le 38$

This function retrieves a numerical example and a visual representation of each of the 38 configurations cataloged in the manuscript. The assigned number associated with the various configuration id's appearing in the manuscript's figures are tabulated as follows:

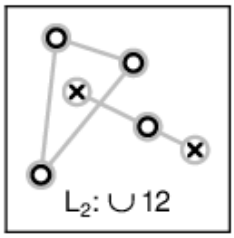
Table S25. Key to retrieving graphs of the various configurations cataloged in the manuscript's figures

Figure	ic	id	ic	id	ic	id	ic	id	ic	id	ic	id
2	1	*a*	2	*b*	3	*c*	4	*d*	5	*e*	6	*f*
3	7	*g*	8	*h*	9	*i*	10	*j*	11	*k*	12	*l*
7	13	A_1	14	A_2	15	A_3	16	A_4	17	A_5	18	A_6
	19	A_7	20	A_8	21	A_9						
8	22	L_1	23	L_2	24	L_3	25	L_4	26	L_5	27	L_6
	28	L_7	29	L_8	30	L_9	31	L_{10}	32	L_{11}	33	L_{12}
	34	L_{13}	35	L_{14}	36	L_{15}	37	L_{16}		L_{17}		

The 2D graphs which are produced when $ipl = T$ are NOT the coordinates given by $rv\$x$. $rv\$x$ is simply a numerical example of the plotted configuration.

Table S26. $rv = tdat1(23)$

```
$x
1    1    1    0
2    1    1    4
3    0    4    0
4    0    0    0
5    4    0    0
6    1    1    2

$y
[1] 1 1 0 0 0 0

$rid
[1] 1 2 3 4 5 6

$n1
[1] 2

$n0
[1] 4

$id
[1] "L2"
```



14. *lohi*()

This function is used in *tdatp*1 to compute internal 2D coordinates for the plots it produces.

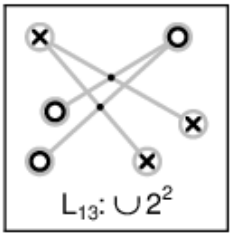
Table S27. $rv = lohi()$

```
$xlo          $xhi          $ylo            $yhi
[1] -0.2      [1] 25.7      [1] -1.648798   [1] 24.5988
```

15. $tdatp1(ic, cro=F, ilab=T, flip=F, inum=T, special=F, ncolm=5)$, $1 \le ic \le 47$

This function is called from tdat1 by the default ipl=T option, but tdatp1 can also be conveniently called directly. It retrieves an example of each cataloged configuration using the *ic* associated with the particular configuration *id* (see Table S25). For example, a representation of configuration L_{13} is obtained as follows

Table S28. *tdatp1*(34)

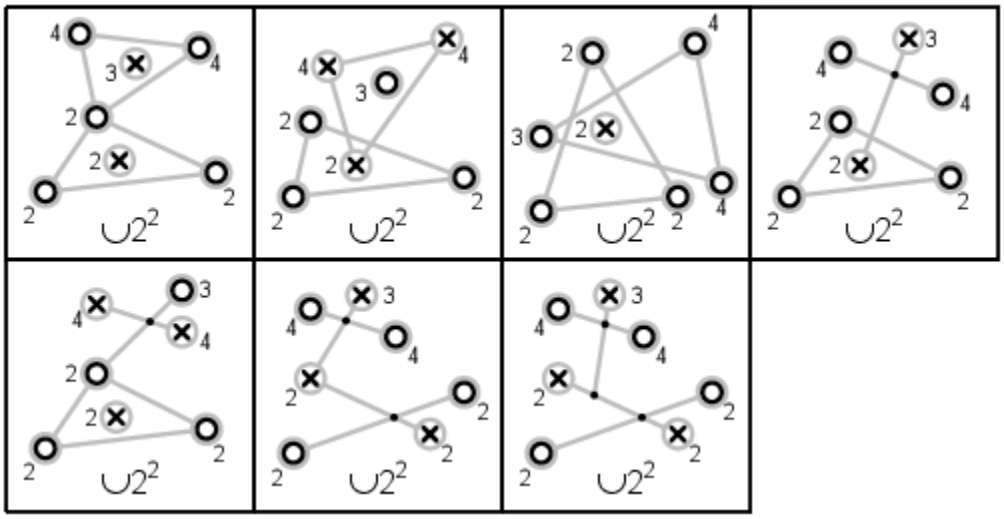


Unlike $rv = tdat1(34)$, which returns a numerical example (a list) of the configuration, $rv = tdatp1(34)$ returns NULL.

In addition, for $41 \le ic \le 47$ the graphs produced are representations of 4D Type II linked configurations following the $\cup 2^2$ format. For these examples, the default $ilab = T$ is recommended.

tdatp1 is versatile in creating multiple plots of configurations in a 5 by 5 (or smaller) grid. The mandatory ic argument accepts a vector. For example, here are the seven different 4D $\cup 2^2$ linked configurations that have been found, packed into a grid with 4 columns (ncolm=4 argument) and saved in a generic tdatp1.eps file generated by using the cro=T option.

Table S29. *tdatp1*(41:47, *cro*=*T*, *ncolm*=4)



16. $ptty(p, typ, r = .05)$, $cnect(p_1, p_2, r = .05)$, $isect(p_1, p_2, p_3, p_4)$, and $twop(p_1, p_2, pct)$

The above plotting functions perform the following four tasks, respectively:

(a) plot a light circle of radius r with either a dark O ($typ = 0$) or X ($typ = 1$) inside
(b) connect two 2D points with coordinates p_1 and p_2
(c) calculate the intersection of two 2D lines $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$, and
(d) calculate $p_1 \times (1 - pct) + p_2 \times pct$

17. $config3(w)$

If w is a minimal overlap configuration of dimension three or fewer, this function identifies the cataloged Type I or Type II configuration it represents. If w is not one of the 38 configurations catalogued, then "unk" or NULL is returned by the function depending on whether $ipr = F$ or $ipr = T$, respectively.

Table S30. Two $config3$ examples

$rv = config3(W1)$	$rv = config3(W2)$
`Type I, Cfig = f` `[1] "Type I" "f"`	`Type II, Cfig = A9` `[1] "Type II" "A9"`

18. $isj(w)$, $isk(w)$ and $isl(w)$

These three functions are only called from config3 when $w\$x$ is a 6×3 matrix, i.e., when w is one of the twelve 3D linked configurations: $L_i, 1 \le i \le 12$. The three function are not intended to be called for any other use.

Table S31. $rv = isk(tdat1(22))$, Note: the argument here is an example of a configuration L_1

```
$i
[1] 15

$n0
[1] 3

$n1
[1] 1
```

```
$ww
$ww$x
     1    1    1    0
2    0    4    0
3    0    0    0
4    4    0    0

$ww$y
[1] 1 0 0 0
```

```
$ww$rid
[1] 1 2 3 4
```

19. *area*2(*w*$*x*)

This function is zero when three 2D points are colinear. It is used in config3 to distinguish between the two 2D, $n=5$ configurations k and l.

Table S32. *area*2 example

$rv=W1\$x[1:3,1:2]$	*area*2(*rv*)
11 10.4 4.1 12 11.0 4.6 13 7.0 2.6	[1] 0.8

20. *area*3(*w*$*x*)

This function is zero when three 3D predictors are colinear. It evaluates the cross product of predictor 2 minus predictor 1 and predictor 3 minus predictor 1. It is used in config3 to distinguish configuration A_4 from A_5 and configuration A_8 from A_9.

Table S33. *area*3 example

$rv=W2\$x[1:3,]$	*area*3(*rv*)
1 6.7 5.8 4 2 9.5 4.7 7 4 10.1 2.6 3	[1] 28.92

21. $mixw(w, iset=-1)$

This function randomly mixes the rows, responses and row ID's (rid) of w. This allows one to potentially obtain different outputs from $eldef(w)$. A non-default $iset \neq -1$ generates a random mix that can be gotten repeatedly with the same $iset$ value.

Table S34. $rv=mixw(W4)$

```
$x
   [,1] [,2] [,3] [,4]
5   1.0 -1.0  0.0  2.0
8   1.2 -1.1 -0.7 -0.4
3   0.6  0.0  1.2  0.0
7   1.2 -1.1 -0.7 -0.4
6   1.0 -1.0  0.0  2.0
1   1.2  1.4  1.8  0.0
4  -0.8  0.6 -1.4  0.0
2  -1.4 -0.8 -2.0  0.0

$y
[1] 1 0 1 1 0 0 1 0

$rid
[1] 1 2 3 4 5 6 7 8
```

22. *twopn*(p_1, p_2, pct) and *pttz*($p, typ, r = .05$)

These two functions are only used within the function $fig10$. *pttz* is like *ptty* except is overwrites any grid lines inside the plotted circles. The function *twopn* is the matrix form of the function *twop*, which is used multiple times inside the function $fig10$ for the purpose of rendering line-segments.

23. The figures appearing in the manuscript (el4Overlap.pdf)

The call to the eleven functions that produce the ten figures depicted in the manuscript are described in the following table.

Table S35. How to recreate the graphs appearing in the manuscript

i	General Call	Sufficient Call	Remarks
1	$fig1(cro = F)$	$fig1()$	
2	$fig2(cro = F, titl = T)$	$fig2()$	
3	$fig3(cro = F, titl = T)$	$fig3()$	
4	$fig4(cro = F)$	$fig4()$	
5	$fig5(cro = F)$	$fig5()$	
6	$fig6(cro = F, titl = T, li = 1.3)$	$fig6(titl = F)$	li to adjust linewidth
7	$fig7(cro = F, titl = T)$	$fig7()$	
8	$fig8(cro = F, titl = T)$	$fig8()$	
9	$fig9(cro = F, ncolm = 3)$	$fig9(ncolm = 3)$	ncolm is number of columns
10	$fig9a(cro = F)$	$fig9a(cro = F)$	
11	$fig10(cro = F, titl = T)$	$fig10()$	

The $cro = T$ option, when used, creates an .eps output in the workspace folder, whereas the $cro = F$ option opens a graph in the current environment. The .eps file is editable (open with WordPad) to adjust the four %%BoundingBox coordinates.

24. *tictactoe*()

This function offers a proof of completeness for the $d = 3$ link catalog (Figure 8 of manuscript). The results of the following table are reformatted in Table 12 of the manuscript.

Table S36. *rv* = *tictactoe*()

```
[1] "cee, top:  L10, L2, L3, L8, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (2,3,8,10), (Reps) = (1,3,1,9)

[1] "cee, mid:  L1, L11, L4, L7, L9, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (1,4,7,9,11), (Reps) = (1,3,6,3,1)

[1] "cee, bot:  L10, L2, L3, L8, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (2,3,8,10), (Reps) = (1,3,1,9)

[1] "dee, top:  L11, L13, L6, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (6,11,13), (Reps) = (4,2,8)

[1] "dee, mid:  L10, L12, L5, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (5,10,12), (Reps) = (4,8,2)

[1] "dee, bot:  L11, L13, L6, out"
Type II (No, Yes, NEW) = (86, 14, 0): (L#'s) = (6,11,13), (Reps) = (4,2,8)

[1] "jay, top:  L17, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (17), (Reps) = (1)

[1] "jay, mid:  L14, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (14), (Reps) = (1)

[1] "jay, bot:  L17, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (17), (Reps) = (1)

[1] "kay, top:  L15, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (15), (Reps) = (1)

[1] "kay, mid:  L16, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (16), (Reps) = (1)

[1] "kay, bot:  L15, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (15), (Reps) = (1)

[1] "ell, top:  L16, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (16), (Reps) = (1)

[1] "ell, mid:  L17, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (17), (Reps) = (1)

[1] "ell, bot:  L16, out"
Type II (No, Yes, NEW) = (99, 1, 0): (L#'s) = (16), (Reps) = (1)

Time difference of 1.28347 mins (the text returned value of rv)
```

25. *down*(*w*,*v*)

This function includes just the *v* rows from *w*$*x* , and the associated *v* responses from *w*$*y* and *v* row ID's from *w*$*rid* . *v* is a vector of row numbers rid (and ignores row-names).

Table S37. $rv = down(W4, c(2,3,5))$

W_4	rv
$x 1 1.2 1.4 1.8 0.0 2 -1.4 -0.8 -2.0 0.0 3 0.6 0.0 1.2 0.0 4 -0.8 0.6 -1.4 0.0 5 1.0 -1.0 0.0 2.0 6 1.0 -1.0 0.0 2.0 7 1.2 -1.1 -0.7 -0.4 8 1.2 -1.1 -0.7 -0.4 $y [1] 0 0 1 1 1 0 1 0 $rid [1] 1 2 3 4 5 6 7 8	$x 2 -1.4 -0.8 -2.0 0 3 0.6 0.0 1.2 0 5 1.0 -1.0 0.0 2 $y [1] 0 1 1 $rid [1] 2 3 5

26. $rv = rquaz(w, remo, n_{stop}, iset = -1, mix = F)$

The list w should be a minimal overlapping data set from which one run will be removed (by remo).

Table S38. 4D Quasi-Completely Separated data set $D_0 = rquaz(W4, 8, 59, 47)$

	$D_0$$x				$D_0$$y	$D_0$$rid		$D_0$$x				$D_0$$y	$D_0$$rid
1	1.2	1.4	1.8	0.0	0	1	31	0.0	-0.8	-2.7	-2.5	1	30
2	-1.4	-0.8	-2.0	0.0	0	2	32	0.1	3.7	0.0	0.7	0	31
3	0.6	0.0	1.2	0.0	1	3	33	3.1	1.4	-0.3	1.0	1	32
4	-0.8	0.6	-1.4	0.0	1	4	34	0.1	-0.7	-0.6	3.1	0	33
5	1.0	-1.0	0.0	2.0	1	5	35	-0.6	1.1	-0.4	-0.4	0	34
6	1.0	-1.0	0.0	2.0	0	6	36	1.0	-0.9	0.2	2.7	0	35
7	1.2	-1.1	-0.7	-0.4	1	7	37	2.9	1.9	-0.4	-1.5	1	36
							38	2.5	-1.9	1.1	0.8	1	37
9	3.2	1.1	0.3	-0.5	1	8	39	-1.3	-3.4	-1.1	0.6	0	38
10	1.2	0.4	0.2	1.1	1	9	40	2.4	1.1	1.7	-2.2	1	39
11	-1.4	-0.9	0.0	0.5	0	10	41	0.8	0.4	-0.1	-3.8	1	40
12	-2.7	1.3	-0.5	-1.3	0	11	42	0.7	-1.9	-2.9	-1.9	1	41
13	-1.3	-0.1	-2.8	0.4	1	12	43	-0.2	2.7	2.2	2.0	0	42
14	-0.7	1.4	0.8	0.6	0	13	44	3.6	0.3	0.5	-1.0	1	43
15	-2.6	-2.2	0.0	0.5	0	14	45	-3.0	1.5	1.9	-1.1	0	44
16	-0.9	0.3	-1.7	-1.7	1	15	46	-1.7	-0.5	-2.7	0.4	0	45
17	-2.2	0.8	1.6	1.8	0	16	47	-2.4	2.4	0.7	0.2	0	46
18	0.9	3.4	1.5	0.5	0	17	48	-0.4	-1.5	1.7	-2.8	1	47
19	-2.0	2.0	-0.8	2.3	0	18	49	0.0	0.8	0.3	1.9	0	48
20	3.0	0.7	-0.4	-0.3	1	19	50	0.3	-0.5	-0.1	-3.2	1	49
21	-2.4	1.3	-1.1	-0.7	0	20	51	2.1	-1.7	1.2	1.7	1	50
22	-1.2	-0.5	2.1	0.5	0	21	52	0.1	1.8	2.6	0.7	0	51
23	2.1	-0.2	2.6	2.0	0	22	53	0.6	-1.1	1.6	1.8	0	52
24	0.8	0.7	-3.1	2.0	1	23	54	-2.1	-1.2	2.2	-0.3	0	53
25	1.6	3.1	0.3	-1.3	1	24	55	1.7	2.1	2.9	0.3	0	54
26	1.1	1.2	0.3	-1.2	1	25	56	2.0	-1.6	0.8	-1.9	1	55
27	0.2	-0.5	-0.9	1.1	1	26	57	0.0	-0.2	-2.0	-0.1	1	56
28	1.4	-3.1	-1.8	-1.0	1	27	58	1.4	0.1	-3.3	-0.6	1	57
29	1.9	0.0	2.3	-2.3	1	28	59	-2.6	-1.1	1.5	1.1	0	58
30	1.0	-1.0	1.3	2.2	0	29	60	-1.9	3.1	-0.9	0.0	0	59

Notice that run 8 was removed from $W4$ to produce D_0 (same as W5). The remaining 53 runs were randomly generated to sandwich-in the first 7 runs, which is the quasi-separated configuration $W5$, between a randomly generated 4D completely separated data set.

27. $safetest(w, tf = F)$

The *safetest* function requires loading the the R *safeBinaryRegression* package, which works by masking R's *glm* function and solving an LP minimization to ascertain whether the data is separated (completely or quasi-completely, without identifying which). Once loaded into an R session, *glm* has a "separation" option, "find" or "test", in addition to the standard glm options. For our purposes, the *safeBinaryRegression* version of *glm* is made with a call to s*afetest(w)*, where ***w* is the list equivalent of a data frame usually provided to *glm***. Once the package is loaded, the s*afetest(w, tf)* calls *glm* with the separation option equal to "find" or "test" for *tf* equal to FALSE or TRUE, respectively. Both options prevent an "unsafe" call to *glm* when separation is detected, with the "find" option going a bit further in reporting the "problem" columns (i.e., dimensions). The linear model used in the s*afetest(w)* call to *glm* is $Y = X_1 + X_2 + \cdots + X_d$ with family option equal to binomial (link="probit") and data equal to a dataframe equivalent of w. For comparison purposes, s*afetest(w)* can be thought of returning either O (for overlap) or S (for separation).

Table S39. From the *safeBinaryRegression* documention

The arguments (to *safeBinaryRegression's* masked *glm*) are identical to the arguments of the (regular) glm function provided in the 'stats' package with the exception of either "find" or "test". Both options prevent the model from being fit to binary data when the maximum likelihood estimate does not exist. Additionally, when separation = "find", the terms separating the sample points are identified when the maximum likelihood estimate is found not to exist. The following arguments are passed to the glm function

To verify that *elstat* and *safetest* come to the same conclusions for various data sets, a limited scale comparison check using D_0 from Table S41 as a starter data set was conducted. Calls to the EL function *elstat(w)* return, among other things, vectors u_1 and u_0 that satisfy $S \cap F \neq \phi$, where $S = \sum_{i=1}^{n_1} u_{1i} X_i^1$ and $F = \sum_{j=1}^{n_0} u_{0j} X_j^0$, when overlap exists. Such is also the case for quasi-complete configurations due to the presence of an overlapping sub-structure. In the EL regime, overlap status assessment depends upon the rank of $\delta = deex(w)$ and the sum of the weights w_{tot} (see Table 5 of the manuscript). For quasi-separated data, w_{tot} is exactly one when δ is rank deficient, otherwise it is properly between zero one.

elstat(w) returns one of four statuses: O (for overlap), Q (for quasi-complete separation), C (for complete separation), and NMR (for no mixed results). When *w* represents an overlapping or quasi separated data set, *eldef(w)* sequentially drops runs until it settles on, and returns, a minimal

configuration. The configuration returned by *eldef* is typically not unique. When the configuration represented by w is quasi-completely separated, $eldef(w)$ returns a smaller trimmed configuration (with the same quasi status) based upon a subset of the u_{1i}'s and u_{0j}'s that have prominent magnitudes.

The so-called "causes" reported by *safeBinaryRegression* for separation problems encountered in the tf=F mode were always the same (i.e., not informative about culprit dimensions when overlap was not present), as shown below in Table S40.

Table S40. *safeBinaryRegression* response when *safetest* indicated separation (S) in Table S41

tf	*safetest(tf)*	*safeBinaryRegression*
F	S	`Error in glm(Y ~ X1 + X2 + X3 + X4, data = D_i, family = binomial(link = "probit")). The following terms are causing separation among the sample points: (Intercept), X1, X2, X3, X4`
T	S	`Separation exists among the sample points. This model cannot be fit by maximum likelihood.`

The example of a completely separated data set given in the safeBinaryResponse documentation is x = c(-2, -1, 1, 2) , y <- c(0, 0, 1, 1) and the example of a quasi-completely separated data set is x = c(-2, 0, 0, 2) , y <- c(0, 0, 1, 1) .

Table S41. Benchmark Comparisons of $elstat(D_i)$ and $safetest(D_i)$ (EL and ST, resp.)

i	Data (D_i)	EL_{rank}	EL_{sum}	Description of D_i	$Status_{EL}$	$Status_{ST}$
0	$D_0 = full\ dataset$	4	0.010	Table S38	*Q*	*S*
1	$D_1 = eldef(D_0)\$wq$	3	1	Runs 1, 2, 3, 4, 5, 6	*Q*	*S*
2	$D_2 = D_0 \mid y_8 = 1 - y_8$	4	1	Modified data set	*O*	*O*
3	$D_3 = eldef(D_2)$	4	1	Runs 8, 38, 49, 50, 56, 59	*O*	*O*
4	$D_4 = D_2 - D_3$	4	0.013	Reduced data set	*Q*	*S*
5	$D_5 = eldef(D_4)\$wq$	3	1	Runs 1, 2, 3, 4, 5, 6	*Q*	*S*
6	$D_6 = D_0 \mid y_9 = 1 - y_9$	4	1	Modified data set	*O*	*O*
7	$D_7 = eldef(D_6)\$wq$	4	1	Runs 4, 5, 9, 45, 53, 59	*O*	*O*
8	$D_8 = D_6 - D_7$	4	0	Reduced data set	*C*	*S*

Note: Status Key: S=Q or C, where O=Overlap, Q=Quasi-separated, C=Completely-separated

The results are unaffected if the run orders of D_i's are randomly shuffled with the function *mixw* .