SPSS

Assuming you have a family identifier (Family) an individual identifier (Individual – coded 1, 2 for twins 1 and 2) and one or more traits.

	Family	Individual	Trait
1	1.00	1.00	24.00
2	1.00	2.00	42.00
3	2.00	1.00	21.00
4	2.00	2.00	4.00
5	3.00	1.00	2.00
6	3.00	2.00	1.00
7	4.00	1.00	3.00
8	4.00	2.00	12.00

The following syntax will restructure the data – the number of traits does not affect the way this syntax works and you do not need to list the traits in the syntax

```
SORT CASES BY Family Individual .
CASESTOVARS
 /ID = Family
 /INDEX = Individual
 /GROUPBY = VARIABLE .
```

The resulting file will look like this

	Family	Trait.1.00	Trait.2.00
1	1.00	24.00	42.00
2	2.00	21.00	4.00
3	3.00	2.00	1.00
4	4.00	3.00	12.00

SAS – 2 examples of code

Thanks to Hermine Maes for the following code

```
* code to change datafile from one record(line) per individual to one
record per twin pair;
libname here ' ';
DATA TWINDATA; set here.yourdataset;
* FAMNO: family number;
* ID: 1=twin1, 2=twin2;
* ZYG: 1=MZF, 2=DZF, 3=MZM, 4=DZM, 5=DZO;
* SEX: 1=male, 2=female;
* new variable OSEX to change order in DZO twins such that twin1=male
and twin2=female;
 osex=.;  if zyg=5 and ((id=1 and sex=2) or (id=2 and sex=1)) then
osex=1;
 if zyg=5 and osex=1 then do; if id=1 then id=3; if id=2 then id=1;
if id=3 then id=2; end;

DATA T1 T2; SET TWINDATA;
 array VARS var1 var2 var3 var4;
 array VARS1 var1t1 var2t1 var3t1 var4t1;
 array VARS2 var1t2 var2t2 var3t2 var4t2;
 if id=1 then do;  do over VARS; VARS1=VARS; end; output T1; end;
 if id=2 then do;  do over VARS; VARS2=VARS; end; output T2; end;

DATA TT1; SET T1; keep famno zyg var1t1 var2t1 var3t1 var4t1;
DATA TT2; SET T2; keep famno zyg var1t2 var2t2 var3t2 var4t2;
DATA TWINS; MERGE TT1 TT2; by famno;
* code to make ordinal/rectangular file;
file "twins.ord"; *or "twins.rec";
put  famno 7. zyg 2. (var1t1 var2t1 var3t1 var4t1 var1t2 var2t2
var3t2 var4t2) (3.);
```

Thanks to Yan Zhou for the following code

Subsample.txt looks like this

Voca	famid	subid	time
26	308	1071	1
25	308	1071	2
26	308	1071	3
25	308	1071	4
26	308	1071	5
26	308	2929	1
24	308	2929	2
31	308	2929	3
26	308	2929	4
34	309	1001	1

```
/*
The original data file is "subsample.txt".

It is longitudinal twin data, where we have:
famid(family id), subid (subject id), time, and voca (vocabulary score).

The initial format is one record per line, and we can have up to records per
subject.

Import data into SAS and create a library "nyt", then run the following codes:
1) make a subject-level multiple variable data file;
2) make a family-level multiple variable data file;

Then, change them back:
3) From subject-level multiple variable data to multiple record data;
4) From family-level multiple variable to multiple record data.
*/

LIBNAME nyt 'DATA LOCATION';

PROC SORT DATA=nyt.subsample; BY famid subid time; RUN;

/* 1) Converting multiple record data set to subject-level multiple variable data*/
DATA mvar_sub;
      ARRAY vo[5] voca1-voca5;
      i=time;
      DO i=1 TO 5 UNTIL (LAST.subid);
            SET nyt.subsample; BY famid subid time;
            vo[i]=voca;
      END;
      OUTPUT;
```

```
    KEEP famid subid voca1-voca5;
RUN;

/* 2) Converting multiple record data to family-level multiple variable data*/
DATA mvar_fam;
    ARRAY vo[10] avoca1-avoca5 bvoca1-bvoca5;
    DO i=1 TO 10 UNTIL (LAST.famid);
        SET nyt.subsample; BY famid subid time;
        IF (FIRST.famid) THEN a=subid;
            IF (subid=a) THEN DO;
                vo[time]=voca;
                asubid=subid;
                END;
            IF (subid NE a) THEN DO;
                vo[time+5]=voca;
                bsubid=subid;
                END;
    END;
    OUTPUT;
    KEEP famid asubid bsubid
        avoca1-avoca5
        bvoca1-bvoca5;
RUN;

/* 3) Converting subject-level multiple variable data to multiple record data*/
DATA mrec_1;
    SET mvar_sub;
    ARRAY vo[5] voca1-voca5;
    DO i=1 TO 5;
        voca=vo[i];
        time=i;
        OUTPUT;
    END;
    KEEP famid subid time voca;
RUN;

/* 4) Converting family-level multiple variable data set to multiple record data*/
DATA mrec_2;
    SET mvar_fam;
    ARRAY vo[10] avoca1-avoca5 bvoca1-bvoca5;
    DO i=1 TO 10;
        IF (i LE 5) THEN DO;
            voca=vo[i];
            time=i;
            subid=asubid;
            END;
```

```
            IF (i GE 6) THEN DO;
                voca=vo[i];
                time=i-5;
                subid=bsubid;
                END;
            OUTPUT;
        END;
        KEEP famid subid time voca;
RUN;
```

R

Thanks to Jeff Spies for the function make.multiv.R which is in this folder

From http://oit.utk.edu/scc/RforSAS&SPSSusers.pdf

```
# R Program to Reshape Data.
load(file="c:\\mydata.Rdata")
print(mydata)

# We need an ID variable for this exercise.
# We can extract it from rownames with this.
mydata$subject <- as.numeric( rownames(mydata) )
# Or we could generate it from scratch like this.
mydata$subject <- 1:8
print(mydata)
library(reshape)
mylong<-melt(mydata,id=c("subject","workshop","gender"))
print(mylong)

# By leaving "value" out, it becomes the measure
# for variable.
mywide<-cast(mylong, subject+workshop+gender~variable)
print(mywide)
```

Thanks to Tim York for the following code

```
#Example2:  function for organizing twin data

#create data
twins <- data.frame(famid= rep(1:10,each=2), twinid= rep(1:2,10), zyg=
c(rep(1,10),rep(2,10)),
                    weight= rnorm(20), height= rnorm(20))


#INPUT:  data.frame where each data for each individual is listed in separate
record
#OUTPUT: data.frame where twin pairs are listed in a single record
twindat <- function(dat, famid, twinid, zygosity) {
  datA <- dat[dat[,twinid]==min(dat[,twinid]),]    #twin1
  datB <- dat[dat[,twinid]==max(dat[,twinid]),]    #twin2
  DAT <- merge(datA, datB, by=famid, all.x=TRUE, all.y=TRUE,
suffixes=c("_T1","_T2"))
  DAT[,paste(twinid,"_T2",sep="")] <- NULL
```

```
  DAT[,paste(zygosity,"_T2",sep="")] <- NULL
  DAT[,twinid] <- DAT[,paste(twinid,"_T1",sep="")]
  DAT[,paste(twinid,"_T1",sep="")] <- NULL
  DAT[,zygosity] <- DAT[,paste(zygosity,"_T1",sep="")]
  DAT[,paste(zygosity,"_T1",sep="")] <- NULL
  return(DAT)
}

# Four arguements needed for twindat() function
twindat(dat=twins, famid="famid", twinid="twinid", zygosity="zyg")
```

STATA

From http://www.ats.ucla.edu/stat/stata/modules/reshapew.htm

Example #3: Reshaping wide with character suffixes

The examples above showed how to reshape data using numeric suffixes, but **reshape** can handle character suffixes as well. Consider the **dadmoml** data file shown below.

use dadmoml, clear

list

	famid	name	inc	dadmom
1.	2	Art	22000	dad
2.	1	Bill	30000	dad
3.	3	Paul	25000	dad
4.	1	Bess	15000	mom
5.	3	Pat	50000	mom
6.	2	Amy	18000	mom

Let's reshape this to be in a wide format, containing one record per family. The **reshape** command below uses **string** to tell reshape that the suffix is character.

reshape wide name inc, i(famid) j(dadmom) string

Let's look at the data before and after reshaping.

list

	famid	name	inc	dadmom
1.	2	Art	22000	dad
2.	1	Bill	30000	dad
3.	3	Paul	25000	dad
4.	1	Bess	15000	mom
5.	3	Pat	50000	mom
6.	2	Amy	18000	mom

reshape wide name inc, i(famid) j(dadmom) string

(note: j = dad mom)

Data long -> wide

```
Number of obs.                     6   ->       3
Number of variables                4   ->       5
j variable (2 values)          dadmom   ->   (dropped)
xij variables:
                               name   ->   namedad namemom
                                inc   ->   incdad incmom
-----------------------------------------------------------------------------
```

list

	famid	namedad	incdad	namemom	incmom
1.	1	Bill	30000	Bess	15000
2.	2	Art	22000	Amy	18000
3.	3	Paul	25000	Pat	50000

Summary

Reshaping data long to wide

Long format

	famid	birth	age
1.	1	1	9
2.	1	2	6
3.	1	3	3
4.	2	1	8
5.	2	2	6
6.	2	3	2
7.	3	1	6
8.	3	2	4
9.	3	3	2

reshape wide age, j(birth) i(famid)

Wide format

	famid	age1	age2	age3
1.	1	9	6	3
2.	2	8	6	2
3.	3	6	4	2

The general syntax of **reshape wide** can be expressed as:

reshape wide long-var(s)**, i(** wide-id-var **) j(** var-with-suffix **)**
where
long-var(s) is the name of the long variable(s) to be made wide e.g. age
wide-id-var is the variable that uniquely identifies wide
observations, e.g. famid

var-with-suffix is the variable from the long file that contains
the suffix for the wide variables, e.g. age