DATA MANIPULATION IN SAS

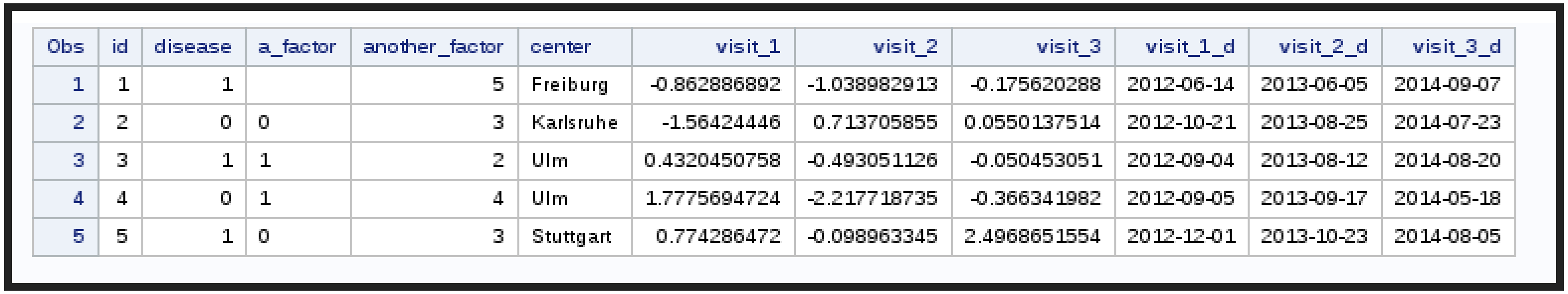
ARTHUR ALLIGNOL

SAS LABELS

ILLUSTRATION

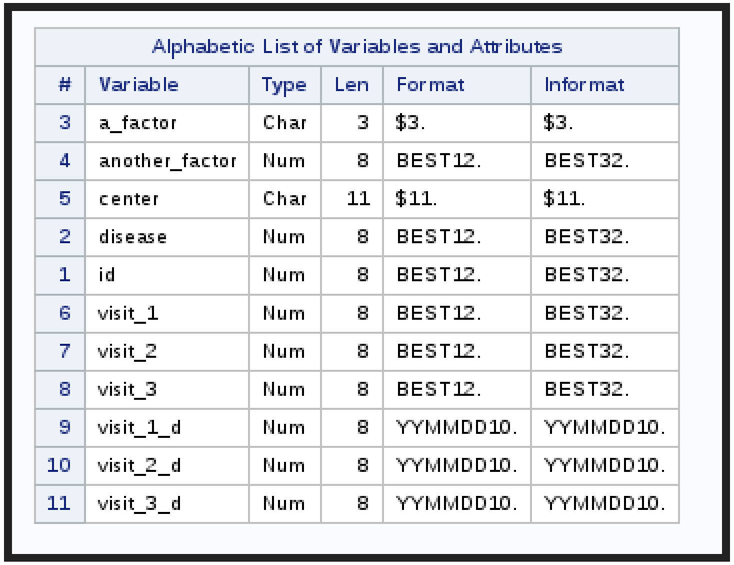
Fake data set `df.csv`

```
proc import datafile = '/folders/myshortcuts/WiMa_Praktikum/lectures/illustration/df.csv'
    out = df
    dbms = csv
    replace;
run;

proc print data = df(obs = 5);
run;
```



Obs	id	disease	a_factor	another_factor	center	visit_1	visit_2	visit_3	visit_1_d	visit_2_d	visit_3_d
1	1	1		5	Freiburg	-0.862886892	-1.038982913	-0.175620288	2012-06-14	2013-06-05	2014-09-07
2	2	0	0	3	Karlsruhe	-1.56424446	0.713705855	0.0550137514	2012-10-21	2013-08-25	2014-07-23
3	3	1	1	2	Ulm	0.4320450758	-0.493051126	-0.050453051	2012-09-04	2013-08-12	2014-08-20
4	4	0	1	4	Ulm	1.7775694724	-2.217718735	-0.366341982	2012-09-05	2013-09-17	2014-05-18
5	5	1	0	3	Stuttgart	0.774286472	-0.098963345	2.4968651554	2012-12-01	2013-10-23	2014-08-05

ILLUSTRATION



Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Informat
3	a_factor	Char	3	$3.	$3.
4	another_factor	Num	8	BEST12.	BEST32.
5	center	Char	11	$11.	$11.
2	disease	Num	8	BEST12.	BEST32.
1	id	Num	8	BEST12.	BEST32.
6	visit_1	Num	8	BEST12.	BEST32.
7	visit_2	Num	8	BEST12.	BEST32.
8	visit_3	Num	8	BEST12.	BEST32.
9	visit_1_d	Num	8	YYMMDD10.	YYMMDD10.
10	visit_2_d	Num	8	YYMMDD10.	YYMMDD10.
11	visit_3_d	Num	8	YYMMDD10.	YYMMDD10.

- Need to change the format of `a_factor` to numeric

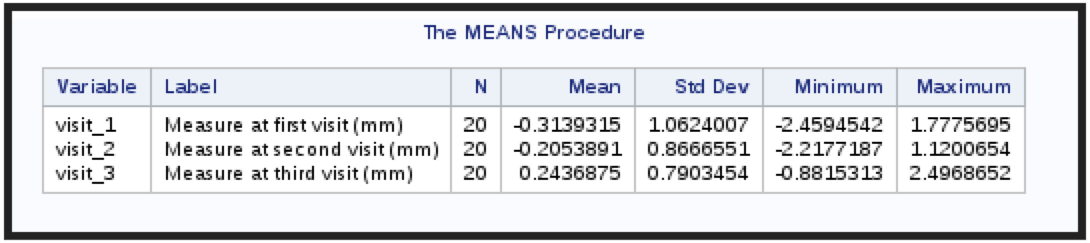
```
data df;
    set df;
    newfactor = input(a_factor, 3.);
run;
```

LABELLING

- Both variables and values can be labelled
- Once created, these labels will appear in the output of statistical procedures
    - Nicer and clearer output

LABELS FOR VARIABLES

```
data df;
    set df;
    label visit_1 = "Measure at first visit (mm)"
        visit_2 = "Measure at second visit (mm)"
        visit_3 = "Measure at third visit (mm)";
run;

proc means data = df;
var visit_1 visit_2 visit_3;
run;
```



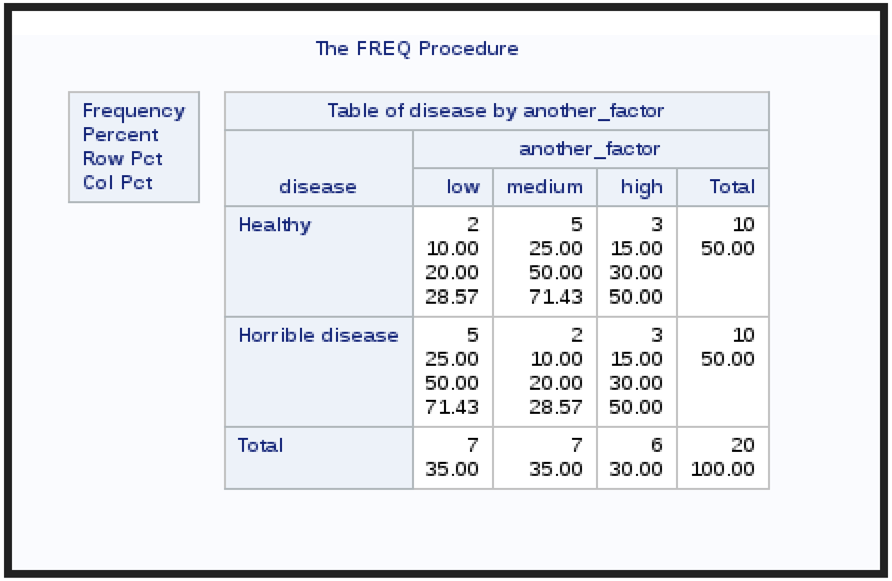
The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
visit_1	Measure at first visit (mm)	20	-0.3139315	1.0624007	-2.4594542	1.7775695
visit_2	Measure at second visit (mm)	20	-0.2053891	0.8666551	-2.2177187	1.1200654
visit_3	Measure at third visit (mm)	20	0.2436875	0.7903454	-0.8815313	2.4968652

LABELS FOR VALUES

Specified through creating new formats

```
proc format;
    value dis 1 = 'Horrible disease' 0='Healthy';
    value factor_one 1 = 'Factor present' 0 = 'Factor absent';
    value factor_two 1-2='low' 3 = 'medium' 4-5 = 'high';
run;

* an example in proc freq;
proc freq data = df;
    format disease dis. another_factor factor_two.;
    tables disease * another_factor;
run;
```



The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of disease by another_factor

	another_factor			
disease	low	medium	high	Total
Healthy	2 10.00 20.00 28.57	5 25.00 50.00 71.43	3 15.00 30.00 50.00	10 50.00
Horrible disease	5 25.00 50.00 71.43	2 10.00 20.00 28.57	3 15.00 30.00 50.00	10 50.00
Total	7 35.00	7 35.00	6 30.00	20 100.00

DATES

DATES AND TIMES

SAS handles 3 types of date and time values

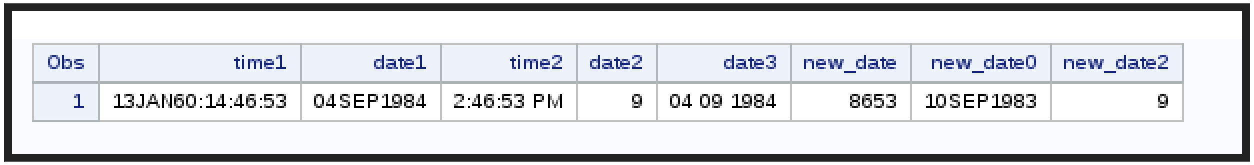
- Time values (*internal*: Number of seconds since midnight)
- Date values (*internal*: Number of days since 1.1.1970)
- Datetime values (*internal*: Number of seconds since 1.1.1970)

Example: Create some dates and times. Note that a `proc print` will display the internal representation

```
data some_dates;
    time1 = '15:00't;
    date1 = '18jun2016'd;
    datetime1 = '3nov1995:15:00:00'dt;
run;
```

FORMAT FOR DATES AND TIMES

```
data test_date;
    time1 = 1090013;  format time1  datetime.;
    date1 = 9013;    format date1  date9.;
    time2 = time1;    format time2  timeampm.;
    date2 = date1;    format date2  month.;
    date3 = date1;    format date3  DDMMYYB10.;

    new_date = date1 - 360; * math operation work with dates;
    new_date0 = new_date; format new_date0 date9.;
    new_date2 = month(new_date);    * extract month;

run;
```



Obs	time1	date1	time2	date2	date3	new_date	new_date0	new_date2
1	13JAN60:14:46:53	04SEP1984	2:46:53 PM	9	04 09 1984	8653	10SEP1983	9

TEXT PROCESSING

CONCATENATE

`cat`	preserve all spaces
`cats`	remove trailing blanks
`catt`	remove all blanks
`catx`	join with a separator (first argument)

```
data text1;
    x1 = 'cats ';
    x2 = ' apples';
    x3 = 'and dogs';

    all1 = cat(of x1-x3); /* same as cat(x1, x2, x3) */
    all2 = cats(of x1-x3);
    all3 = catt(of x1-x3);
    all4 = catx("|", of x1-x3);

    put all1=;
    put all2=;
    put all3=;
    put all4=;
run;
```

```
all1=cats  applesand dogs
all2=catsapplesand dogs
all3=cats applesand dogs
all4=cats|apples|and dogs
```

REMOVE CHARACTERS

```
data text2;
    expr1 = 'A; simple; sentence';
    new = compress(expr1, ";");
    put new=;

    expr2 = '122-1143 76';
    new2 = compress(expr2, "-", 'd'); * remove '-' and any digit;
    put new2=;

    expr3 = '1   2   4   5     7';
    new3 = compress(expr3, , 's'); * Remove spaces;
    put new3=;
run;
```

```
new=A simple sentence
new2=11
new3=12457
```

SIMPLE MATCH AND REPLACEMENT

```
data text3;
    a = count("banana", "a"); * count the number of a's;
    put a=;

    where = "university of california"; * Position of 'cal' in the string;
    i = index(where,"cal");
    put i=;

    hihi = reverse(where);
    put hihi=;

    up = upcase(where);
    put up=;

    new = translate(where, 'UC', 'uc'); * Change u's and c's into U's and C's
    put new=;

    new2 = tranwrd(where, 'university', 'beach'); * Replace words;
    put new2=;
run;
```

```
a=3
i=15
hihi=ainrofilac fo ytisrevinu
up=UNIVERSITY OF CALIFORNIA
new=University of California
new2=beach of california
```

DATA MANIPULATION

ROW SUBSCRIPTING

- With a condition

```
data ulm;
    set df;
    if upcase(center) eq 'ULM' then delete;
run;
```

- `where` statement

```
data high;
    set df(where = (another_factor in (4 5)));
run;
```

COLUMN SUBSCRIPTING

- Keep every variables starting with `visit` (`:` is a *wildcard*)

```
data visit;
    set df;
    keep visit:;
run;
```

- Drop every variables between `id` and `center`

```
data visit2;
    set df;
    drop id--center;
run;
```

- Keep only numeric variables

```
data numeric;
    set df;
    keep id-numeric-visit_3_d;
run;
```

COLUMN SUBSCRIPTING

- Keep character variables

```
data char;
    set df;
    keep id-character-visit_3_d;
    run;
```

- Remove `visit_1` to `visit_3`

```
data sans_visit;
    set df;
    drop visit_1-visit_3;
run;
```

PROC SQL

`proc sql` permits to sort, summarize, subset, join (merge), and concatenate datasets, create new variables, and print the results or create a new table or view all in one step.

- A mix of SAS and SQL syntax
- Does not need sorted data sets for merge operations

The command starts with `proc sql` and ends with `quit;` (not `run`)

SELECT VARIABLES WITH PROC SQL

Create a data set new from `df` containing the variables `visit_1` to `visit_3`

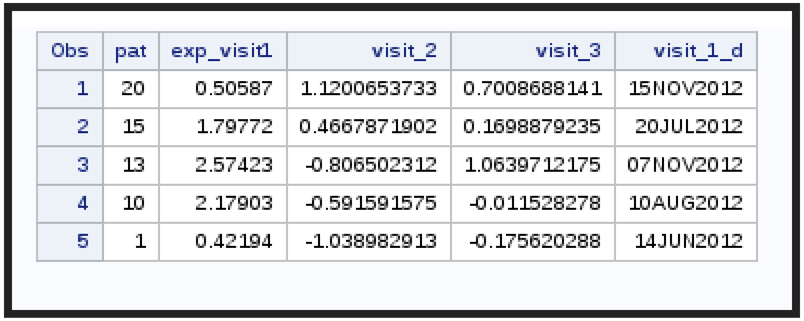
```
proc sql;
    create table new as
        select visit_1, visit_2, visit_3
        from df
quit;
```

MORE COMPLICATED

Create data set new_new based on `df`

- select variable `id` and rename as `pat`
- create `exp_visit` as $\exp(\text{visit})$
- select variables `visit_2`, `visit_3` and `visit_1_d` (with format change)
- Select only the individuals for which `center` equals `ulm` and `freiburg`
- Finally, order by descending `pat`

```
proc sql;
    create table new_new as
        select id as pat, exp(visit_1) as exp_visit1, visit_2, visit_3, visit_1_d format=date9.
        from df
        where center in ("ulm", "Freiburg")
        order by id desc;
quit;
```



Obs	pat	exp_visit1	visit_2	visit_3	visit_1_d
1	20	0.50587	1.1200653733	0.7008688141	15NOV2012
2	15	1.79772	0.4667871902	0.1698879235	20JUL2012
3	13	2.57423	-0.806502312	1.0639712175	07NOV2012
4	10	2.17903	-0.591591575	-0.011528278	10AUG2012
5	1	0.42194	-1.038982913	-0.175620288	14JUN2012

ARRAY

Arrays in SAS permit to perform the same task on a group of variables

```
array arrayname variable_list <$>;
```

1. All the variables in an array must be of the same type
2. An array can not have the same name as a variable
3. You can use the keyword `_temporary_` instead of a variable list

```
data test_array;
    set df;
    array x visit_1-visit_3;
    array res{3};
    do i=1 to dim(x);
        res{i} = x{i} * 10;
        end;
    keep visit_1-visit_3 res:;
run;
```

DATA RESHAPING

PROC TRANSPOSE

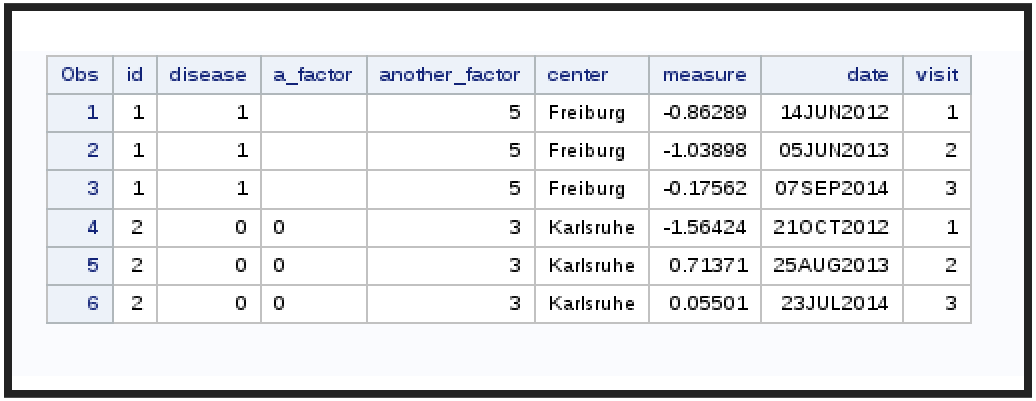
The name says it all. The problem is that `proc transpose` can only manage one variable at a time. Thus we need to

- Transpose `visit_X`
- Transpose `visit_X_d`
- Merge back with the whole data set

```
proc transpose data = df
    out = long1(rename=(col1=measure)) name = visit;
    by id;
    var visit_1-visit_3;
run;

proc transpose data = df
    out = long2(rename=(col1=date)) name = visit;
    by id;
    var visit_1_d--visit_3_d;
run;

* and merge;
data df_long;
    merge long1
          long2
          df(keep = id a_factor another_factor disease center);
    by id;
run;
```

RESHAPE USING A DATA STEP

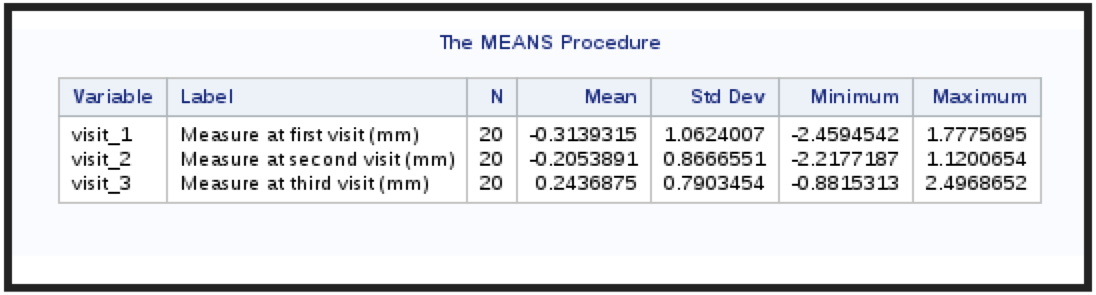
```
data df_long2;
    set df;

    array m visit_1-visit_3;
    array d visit_1_d--visit_3_d;

    do _i = 1 to dim(m);
        measure = m(_i);
        date = d(_i);
        visit = _i;
        output;
    end;

    format date date9.;
    keep id center a_factor another_factor disease measure date visit;
run;
```



Obs	id	disease	a_factor	another_factor	center	measure	date	visit
1	1	1		5	Freiburg	-0.86289	14JUN2012	1
2	1	1		5	Freiburg	-1.03898	05JUN2013	2
3	1	1		5	Freiburg	-0.17562	07SEP2014	3
4	2	0	0	3	Karlsruhe	-1.56424	21OCT2012	1
5	2	0	0	3	Karlsruhe	0.71371	25AUG2013	2
6	2	0	0	3	Karlsruhe	0.05501	23JUL2014	3

ALWAYS CHECK !!!

```
proc means data = df;
    var visit_1-visit_3;
run;
```

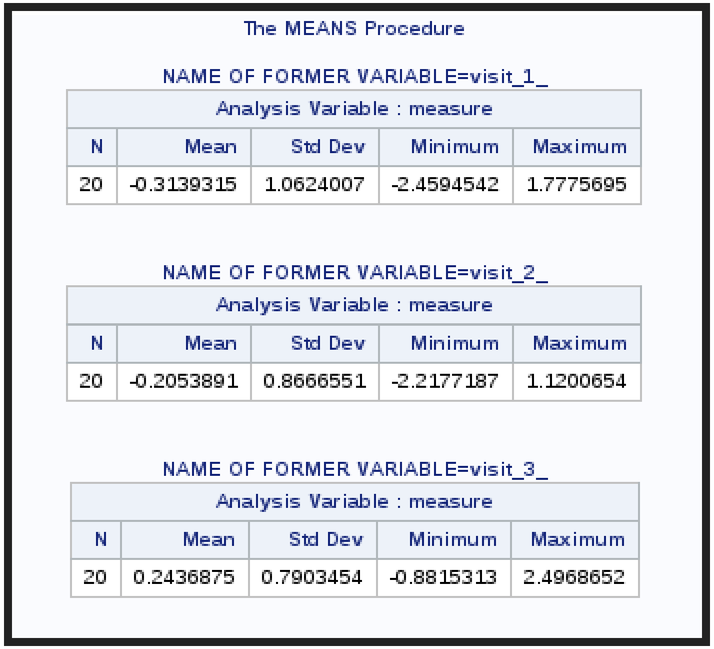


The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
visit_1	Measure at first visit (mm)	20	-0.3139315	1.0624007	-2.4594542	1.7775695
visit_2	Measure at second visit (mm)	20	-0.2053891	0.8666551	-2.2177187	1.1200654
visit_3	Measure at third visit (mm)	20	0.2436875	0.7903454	-0.8815313	2.4968652

ALWAYS CHECK !!!

```
* need to sort before using the by statement in proc means;
proc sort data = df_long
    out = df_long;
    by visit;
proc means data = df_long;
    var measure;
    by visit;
    run;
```



The MEANS Procedure

NAME OF FORMER VARIABLE=visit_1_

Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	-0.3139315	1.0624007	-2.4594542	1.7775695

NAME OF FORMER VARIABLE=visit_2_

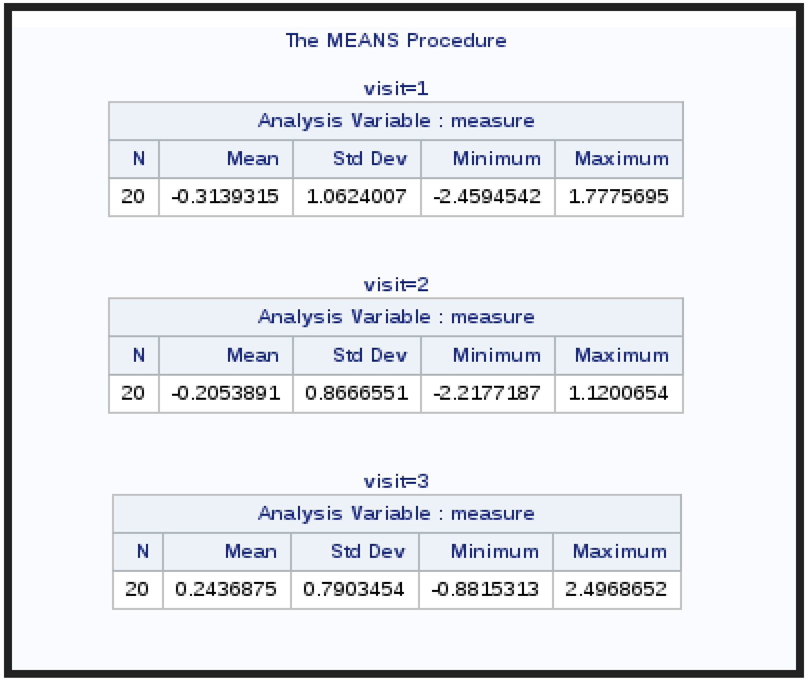
Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	-0.2053891	0.8666551	-2.2177187	1.1200654

NAME OF FORMER VARIABLE=visit_3_

Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	0.2436875	0.7903454	-0.8815313	2.4968652

ALWAYS CHECK !!!

```
proc sort data = df_long2
    out = df_long2;
    by visit;
proc means data = df_long2;
    var measure;
    by visit;
    run;
```



The MEANS Procedure

visit=1

Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	-0.3139315	1.0624007	-2.4594542	1.7775695

visit=2

Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	-0.2053891	0.8666551	-2.2177187	1.1200654

visit=3

Analysis Variable : measure				
N	Mean	Std Dev	Minimum	Maximum
20	0.2436875	0.7903454	-0.8815313	2.4968652

DATA MERGING

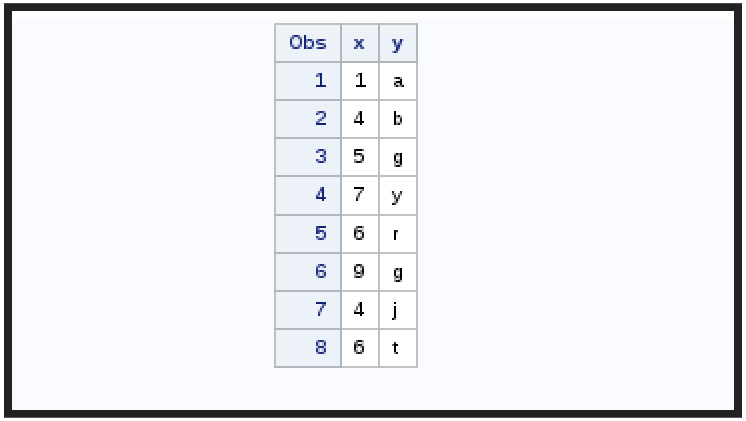
COMBINE DATA BY ROWS

```
data d1;
    input x y $;
    datalines;
    1 a
    4 b
    5 g
    7 y
    ;
run;

data d2;
    input x y $;
    datalines;
    6 r
    9 g
    4 j
    6 t
    ;
run;
```

Note the $ sign after y to specify this variable as character

COMBINE DATA BY ROW

```
data row_bind;
    set d1 d2;
proc print data = row_bind;
run;
```



Obs	x	y
1	1	a
2	4	b
3	5	g
4	7	y
5	6	r
6	9	g
7	4	j
8	6	t

COMBINE DATA BY COLUMN

We first need to rename the columns of d2, otherwise SAS does nothing

```
data d3;
    set d2(rename=(x=z y=w));
run;
```

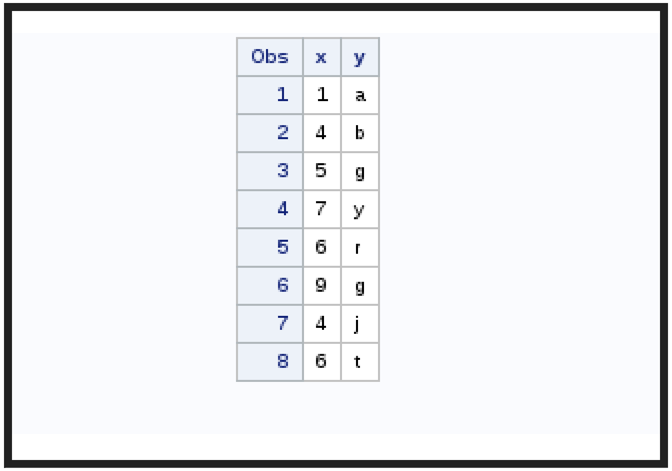
COMBINE DATA BY COLUMN

- Either use 2 `set` statements

```
data col_bind;
    set d1;
    set d3;
run;
```

- Or merge by row numbers (which means, don't specify any variable to merge on)

```
data col_bind2;
    merge d1 d3;
run;
```



Obs	x	y
1	1	a
2	4	b
3	5	g
4	7	y
5	6	r
6	9	g
7	4	j
8	6	t

MERGING

```
data dd1;
    input id letter $;
    datalines;
    20 k
    1  j
    3  h
    7  a
    13 c
    8  s
    ;
run;

data dd2;
    input id digit;
    datalines;
    13 3
    14 8
    7  7
    1  6
    54 0
    ;
run;
```

MERGING

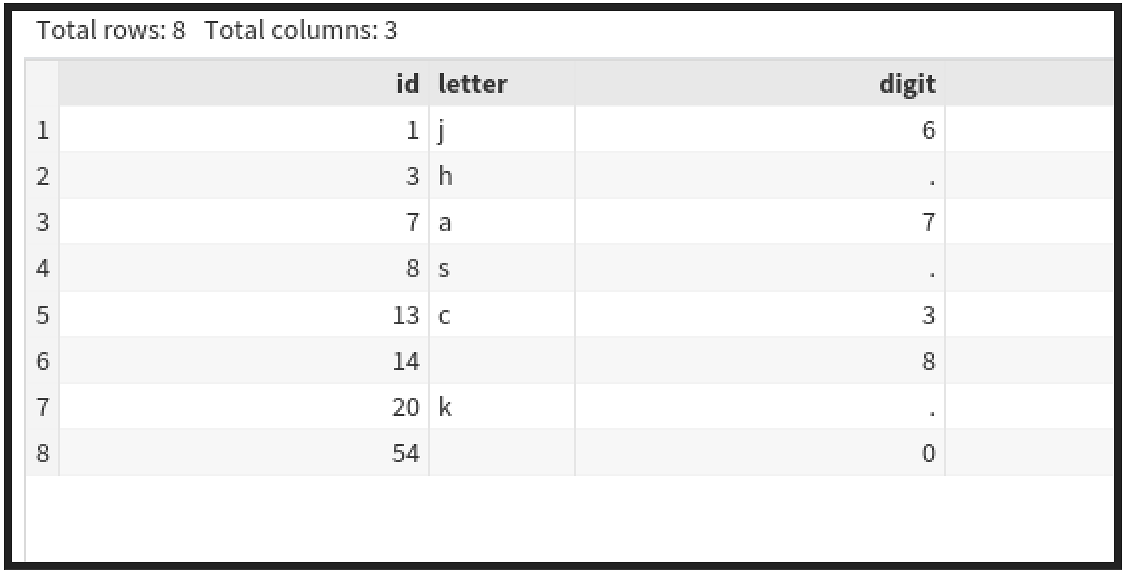
Before merging using the `merge` statement, data sets have to be sorted wrt the variables used for merging

```
proc sort data=dd1
    out=dd1_sort;
    by id;
run;

proc sort data=dd2
    out=dd2_sort;
    by id;
run;
```

MERGING

By default SAS performs a full join

```
data ddmerge;
    merge dd1_sort dd2_sort;
    by id;
run;
```



Total rows: 8   Total columns: 3

	id	letter	digit
1	1	j	6
2	3	h	.
3	7	a	7
4	8	s	.
5	13	c	3
6	14		8
7	20	k	.
8	54		0

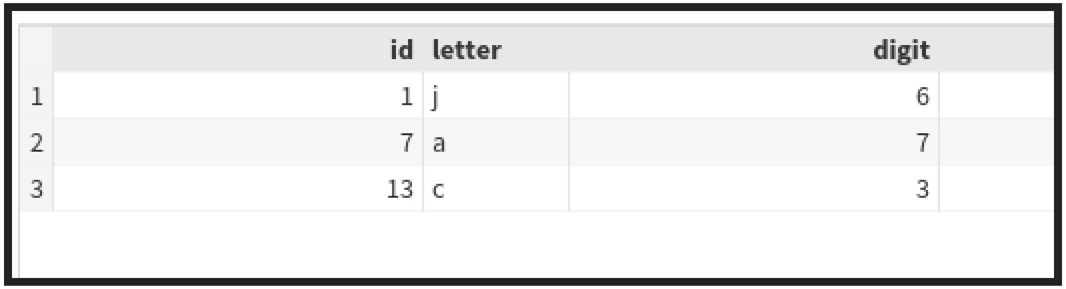
MERGING

To perform other types of joins, use the `in=` directive

- That creates temporary variables indicating from which data the observations are coming from

Natural join

```
data ddmerge_natural;
    merge dd1_sort(in=in1) dd2_sort(in=in2);
    by id;
    if in1 eq 0 or in2 eq 0 then delete;
run;
```

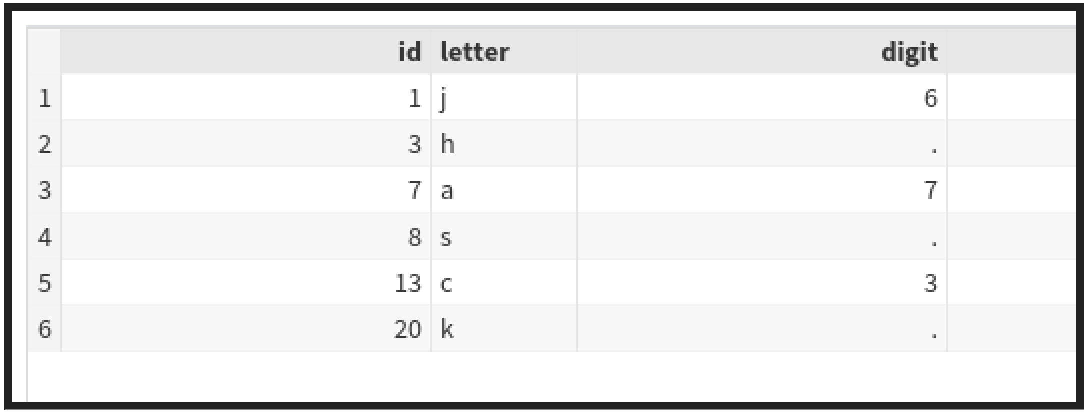


	id	letter	digit
1	1	j	6
2	7	a	7
3	13	c	3

MERGING

Left join

```
data ddmerge_left;
    merge dd1_sort(in=in1) dd2_sort(in=in2);
    by id;
    if in1 eq 0 then delete;
run;
```



	id	letter	digit
1	1	j	6
2	3	h	.
3	7	a	7
4	8	s	.
5	13	c	3
6	20	k	.

MERGING WITH PROC SQL

Natural join

```
proc sql;
    create table ddmerge_natural_sql as
        select *
        from dd1 inner join dd2
        on dd1.id=dd2.id;
quit;
```

- When the variables you join on don't share the same name

```
data dd2_alt;
    set dd2(rename=(id=pat));
run;

proc sql;
    create table ddmerge_natural_sql2 as
        select *
        from dd1 inner join dd2_alt
        on dd1.id=dd2_alt.pat;
quit;
```

MERGING WITH PROC SQL

Left join

```
proc sql;
    create table ddmerge_left_sql as
    select *
    from dd1 left join dd2
    on dd1.id=dd2.id;
quit;
```