

Lecture 20

1. Working with multiple observations: long form and wide form
2. Proc SQL
3. Counting observations with SQL and Transpose
4. Proc SQL to combine datasets: fuzzy merge
5. Longitudinal data: response feature analysis
6. Area under the curve (AUC)

1

Long form: multiple observations per subject,
one measurement (income) in each observation,
with variables identifying subject ID and observation number (year)

Obs	family_ id	income	year
1	1	66483	1990
2	1	69146	1991
3	1	74643	1992

Wide form: one observation per subject,
all measurements (income) in one observation,
with variables identifying subject ID,
and observation number (year) encoded in variable name

Obs	family_ id	income_ 1990	income_ 1991	income_ 1992	income_ 1993	income_ 1994	income_ 1995
1	1	66483	69146	74643	79783	81710	86143
2	2	17510	.	19484	20979	21268	22998

2

Long form. Required for graphing and analysis procedures

Wide form. All measurements from a subject are available for computation because they are all in the same row.

SAS Data step works with one observation (row) at a time.

3 ways to access the values in a column from more than one observation:

- using RETAIN statement to store values “across” rows
- Proc Transpose pivots data to put multiple values in the same row (= wide form)
- Proc SQL (Structured Query Language)

3

Transpose Help chapter:

SAS Help > SAS Products > Base SAS > Base SAS 9.2 Procedures Guide > Procedures
> Proc Transpose

SQL Help chapter:

SAS Help > SAS Products > Base SAS > SAS 9.2 SQL Procedure User's Guide

4

Proc SQL applied to one dataset

Family economic data, in long form. 3 to 6 annual observations per family.

Obs	family_ id	income	year	expenses	debt	cohort	
1	1	66483	1990	49804	no	A	
2	1	69146	1991	65634	no	A	
3	1	74643	1992	61820	no	A	1993 missing
4	1	81710	1994	85504	yes	A	
5	1	86143	1995	75640	no	A	
6	2	17510	1990	21609	yes	B	
7	2	17947	1991	12085	no	B	1992 missing
8	2	20979	1993	22985	yes	B	
9	2	21268	1994	11097	no	B	
10	2	22998	1995	21768	no	B	

Count number of observations per family.

5

Proc SQL;

```
create table pubh.M as
select  family_id, count(income) as n_years_income
from    family_economic_data
group by family_id;
quit;
```

Create SAS permanent dataset pubh.M by selecting these variables from family_economic_data:

group observations for each distinct family_ID,

create n_years_income = COUNT(income)

make one observation per family_ID with 2 variables: family_ID, n_years_income

Ends with quit ,not run.

6

Obs	family_ id	n_years_ income
1	1	5
2	2	5
3	3	6
4	4	6
5	5	5

Summary: pattern of missing observations from Proc _____

n_years_income	Frequency	Percent	Cumulative Frequency	Cumulative Percent
3	2	4.00	2	4.00
4	2	4.00	4	8.00
5	17	34.00	21	42.00
6	29	58.00	50	100.00

7

Select variables that do not repeat within family_id groups:

```
Proc SQL;
  create table M2 as
  select * ,                               * = all variables
  count(income) as n_years_income         no comma after last variable selected
  from family_economic_data
  group by family_id;
quit;
```

Summary count is merged back:

Obs	family_ id	income	year	expenses	debt	cohort	n_years_ income
1	1	66483	1990	49804	no	A	5
2	1	81710	1994	85504	yes	A	5
3	1	74643	1992	61820	no	A	5
4	1	86143	1995	75640	no	A	5
5	1	69146	1991	65634	no	A	5

8

Another way to count nonmissing observations

Use Proc Transpose to reshape from long to wide form:

```
Proc Transpose data=long out=wide prefix =income_ ;
  ID year;
  VAR income;
  BY family_id;
```

Obs	family_	income_	income_	income_	income_	income_	income_	
	id	_NAME_	1990	1991	1992	1994	1995	1993
1	1	income	66483	69146	74643	81710	86143	.
2	2	income	17510	17947	.	21268	22998	20979
3	3	income	57947	62964	68717	75198	75722	70957
4	4	income	64831	71060	71918	73100	74379	72514
5	5	income	18904	19949	21335	23829	23913	.
6	6	income	32057	34770	35834	40899	42372	37387

9

In wide format, we can use the N function

Data count;

```
set wide;
n_years_income = N(of income_1990 - income_1995) ;
```

The N function counts nonmissing values, whereas the NMISS and the CMISS functions count missing values. N requires numeric arguments, whereas CMISS works with both numeric and character values.

Examples

SAS Statements	Results
x1=n(1,0,.,2,5,.);	4
x2=n(1,2);	2
x3=n(of x1-x2);	2

Combining datasets (tables) in SQL

Join two datasets without conditions in Proc SQL to get all combinations of rows = *Cartesian product*.

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

3 rows in A × 3 rows in B = 9 combinations

Just as in MERGE, order of joining matters.

11

```
Proc SQL;
```

```
  create table D as
```

```
  select * from A, B ; select * = select all variables
```

Obs	id	color	mass	pH
1	12	orange	3650	7.8
2	12	orange	3650	8.2
3	12	orange	3650	5.1
4	13	blue	3877	7.8
5	13	blue	3877	8.2
6	13	blue	3877	5.1
7	15	yellow	4103	7.8
8	15	yellow	4103	8.2
9	15	yellow	4103	5.1

ID and mass are common variables:

only IDs and masses from first dataset (A) in product.

12

```
Proc SQL;
  create table E as
  select * from B, A ;
```

Obs	id	mass	pH	color
1	13	11267	7.8	orange
2	13	11267	7.8	blue
3	13	11267	7.8	yellow
4	14	3568	8.2	orange
5	14	3568	8.2	blue
6	14	3568	8.2	yellow
7	15	4103	5.1	orange
8	15	4103	5.1	blue
9	15	4103	5.1	yellow

ID and mass are common variables:

only IDs and masses from first dataset (B) in product.

13

Merge with SQL

No sorting is required, which can speed merging of very large datasets.

Merge observations from A and B with matching IDs, and keep mass measurements separate:

```
Proc SQL;
  create table F as
  select * from A(rename=(mass=mass_A)), B(rename=(mass=mass_B))
  where A.id = B.id ;
```

Since variable ID is in both A and B, need to identify `dataset.variable`

14

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

Merge where A.id = B.id

Obs	id	color	mass_A	mass_B	pH
1	13	blue	3877	11267	7.8
2	15	yellow	4103	4103	5.1

Gives the *intersection* of the two datasets: only IDs that appear in both.

15

Data-step merge by ID gives non-matching observations:

Data E;

```
MERGE A(rename=(mass=mass_A)) B(rename=(mass=mass_B));
BY id;
```

Obs	id	color	mass_A	mass_B	pH
1	12	orange	3650	.	.
2	13	blue	3877	11267	7.8
3	14		.	3568	8.2
4	15	yellow	4103	4103	5.1

To get this from SQL, use a `full join`

16

```
Proc SQL;
  create table F1 as
  select *
  from A(rename=(mass=mass_A)) full join B(rename=(mass=mass_B))
  on A.id = B.id;
```

Obs	id	color	mass_A	mass_B	pH
1	12	orange	3650	.	.
2	13	blue	3877	11267	7.8
3	.		.	3568	8.2
4	15	yellow	4103	4103	5.1

Because A is read first, ID = 14 from B is missing.

17

Use COALESCE function: takes first non-missing value from arguments.

```
Proc SQL;
  create table F2 as
  select * , coalesce(A.id, B.id) as item
  from A(rename=(mass=mass_A)) full join B(rename=(mass=mass_B))
  on A.id = B.id;
```

Obs	id	color	mass_A	mass_B	pH	item
1	12	orange	3650	.	.	12
2	13	blue	3877	11267	7.8	13
3	.		.	3568	8.2	14
4	15	yellow	4103	4103	5.1	15

18

Fuzzy merge in Proc SQL

Patients in a study are supposed to have measurements of their weight and blood pressure within 7 days.

Weight data				Blood pressure data				
Obs	id	date	kg	Obs	patient	date	dbp	sbp
1	33	10MAR2009	78	1	33	15MAR2009	72	112
2	33	12APR2009	81	2	33	10APR2009	68	105
3	33	15MAY2009	80	3	33	30MAY2009	71	110
4	34	02FEB2009	65	4	34	07FEB2009	55	95
5	34	05JUN2009	66	5	34	15JUN2009	60	99
6	35	21MAY2009	71	6	35	21MAY2009	66	110
7	35	08JUN2009	71	7	35	06JUN2009	68	105
8	35	14AUG2009	70					

SQL can merge on variables with different names (ID, patient) and on dates that are within 7 days.

19

Start by merging on ID = patient

```
Proc SQL;
  create table G1 as
  select *
  from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
  where weights.id = BP.patient;
```

Use dataset option `rename` to save dates from weights and blood pressures.

Want to check all combinations of observations where ID = patient:

Result has 9 rows for ID=33, 4 for ID=34, and 6 for ID=35

20

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp
1	33	10MAR2009	78	33	15MAR2009	72	112
2	33	15MAY2009	80	33	15MAR2009	72	112
3	33	12APR2009	81	33	15MAR2009	72	112
4	33	10MAR2009	78	33	10APR2009	68	105
5	33	15MAY2009	80	33	10APR2009	68	105
6	33	12APR2009	81	33	10APR2009	68	105
7	33	10MAR2009	78	33	30MAY2009	71	110
8	33	15MAY2009	80	33	30MAY2009	71	110
9	33	12APR2009	81	33	30MAY2009	71	110
10	34	02FEB2009	65	34	07FEB2009	55	95
11	34	05JUN2009	66	34	07FEB2009	55	95
12	34	02FEB2009	65	34	15JUN2009	60	99
13	34	05JUN2009	66	34	15JUN2009	60	99
14	35	21MAY2009	71	35	21MAY2009	66	110
15	35	14AUG2009	70	35	21MAY2009	66	110
16	35	08JUN2009	71	35	21MAY2009	66	110
17	35	21MAY2009	71	35	06JUN2009	68	105
18	35	14AUG2009	70	35	06JUN2009	68	105
19	35	08JUN2009	71	35	06JUN2009	68	105

21

To check whether weight dates and blood pressure dates are within 7 days, calculate interval:

```
Proc SQL;
  create table G2 as
  select * , ABS(date_wt - date_BP) as interval
  from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
  where weights.id = BP.patient ;
```

Add comma after * for variable list

ABS is absolute value function

ABS(date_wt - date_BP) is positive number of days between dates

Note: used new names for the dates in calculating interval

22

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	33	15MAR2009	72	112	5
2	33	15MAY2009	80	33	15MAR2009	72	112	61
3	33	12APR2009	81	33	15MAR2009	72	112	28
4	33	10MAR2009	78	33	10APR2009	68	105	31
5	33	15MAY2009	80	33	10APR2009	68	105	35
6	33	12APR2009	81	33	10APR2009	68	105	2
7	33	10MAR2009	78	33	30MAY2009	71	110	81
8	33	15MAY2009	80	33	30MAY2009	71	110	15
9	33	12APR2009	81	33	30MAY2009	71	110	48
10	34	02FEB2009	65	34	07FEB2009	55	95	5
11	34	05JUN2009	66	34	07FEB2009	55	95	118
12	34	02FEB2009	65	34	15JUN2009	60	99	133
13	34	05JUN2009	66	34	15JUN2009	60	99	10
14	35	21MAY2009	71	35	21MAY2009	66	110	0
15	35	14AUG2009	70	35	21MAY2009	66	110	85
16	35	08JUN2009	71	35	21MAY2009	66	110	18
17	35	21MAY2009	71	35	06JUN2009	68	105	16
18	35	14AUG2009	70	35	06JUN2009	68	105	69
19	35	08JUN2009	71	35	06JUN2009	68	105	2

23

Add condition that interval between dates (for same ID) must be ≤ 7 days

Proc SQL;

```

create table G3 as
select * , ABS(date_wt - date_BP) as interval
from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
where weights.id = BP.patient and (calculated interval LE 7);

```

Must refer to interval as `calculated interval`

because it is not in either data set

24

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	33	15MAR2009	72	112	5
2	33	12APR2009	81	33	10APR2009	68	105	2
3	34	02FEB2009	65	34	07FEB2009	55	95	5
4	35	21MAY2009	71	35	21MAY2009	66	110	0
5	35	08JUN2009	71	35	06JUN2009	68	105	2

Last, fix the redundant variables ID and patient

25

```
Proc SQL;
  create table G4 as
  select * , ABS(date_wt - date_BP) as interval
  from weights(rename=(date=date_wt)),
       BP (rename=(date=date_BP patient=id ))
  where weights.id = BP.id and (calculated interval LE 7);
```

Obs	id	date_wt	kg	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	15MAR2009	72	112	5
2	33	12APR2009	81	10APR2009	68	105	2
3	34	02FEB2009	65	07FEB2009	55	95	5
4	35	21MAY2009	71	21MAY2009	66	110	0
5	35	08JUN2009	71	06JUN2009	68	105	2

This is a **fuzzy merge** on ID and visit dates within 7-day interval.

Very difficult in SAS data step.

26

Longitudinal data: Response Feature Analysis

Response feature analysis replaces repeated measurements with one outcome: no more longitudinal data, apply simpler analysis method: ANOVA, regression, *t*-test. Common response features:

- mean
- area under the curve (AUC)
- for peaked data: maximum or minimum value
- for peaked data: *time* to maximum or minimum value
- for growth data: slope of regression line

More than one feature can be used, with multiple analyses to compare groups.

27

Visual Analog Scale (VAS) example

A nutrition study compared the immediate effect on feelings of hunger after a breakfast muffin containing 0, 5, or 8 g of short-chain fructooligosaccharides (scFOS).

To measure hunger, participants marked a visual analog scale (VAS) to indicate how hungry they felt:



Distance from zero on scale was numeric response. Participants completed the VAS at 0, 15, 30, 45, 60, 90, 120, 180, and 240 minutes after eating the muffin.

28

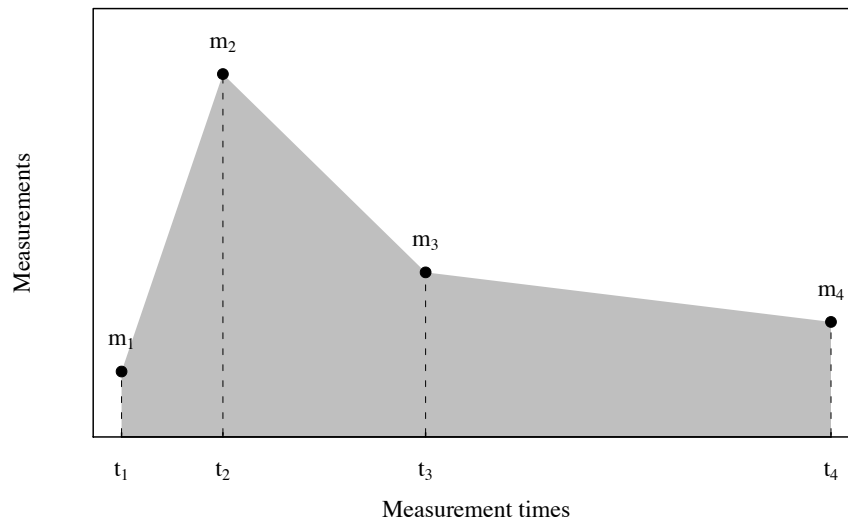


Mean curves in response to each treatment. Differences?

29

Finding the area under a curve: trapezoid rule

Sequence of individual's measurements m_i , taken at times t_i

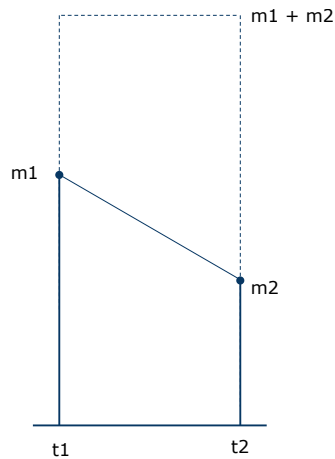


Trapezoid rule: connect measurements with line segments, find area below in gray.

30

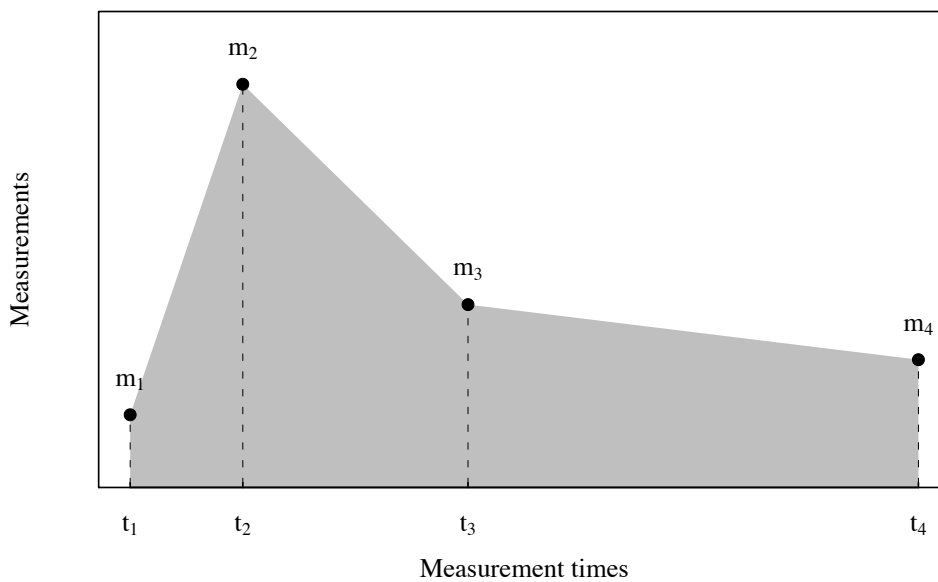
Trapezoid: 4-sided plane figure with 2 parallel sides.

Duplicate trapezoid on top gives rectangle that has twice the area.



$$\text{Trapezoid area} = \frac{1}{2} \{ (t_2 - t_1)(m_1 + m_2) \}$$

31



$$\text{Area under the curve} = \frac{1}{2} \{ (t_2 - t_1)(m_1 + m_2) + (t_3 - t_2)(m_2 + m_3) + (t_4 - t_3)(m_3 + m_4) \}$$

Approximates area under true curve of measured quantity m .

32

Calculating AUC (Area Under Curve)

In example, VAS hunger measured 9 times: at 0, 15, 30, 45, 60, 90, 120, 180, and 240 minutes after eating the muffin.

Convert times to hours: $t_i = 0, .25, .5, .75, 1, 1.5, 2, 3, 4$.

$$\text{AUC} = \frac{1}{2} \left\{ (t_2 - t_1)(m_1 + m_2) + (t_3 - t_2)(m_2 + m_3) + \dots + (t_9 - t_8)(m_8 + m_9) \right\}$$

How many trapezoids?

Use one array for times, one for measurements.

33

```
data AUC;
  set VAS_hunger;
  array m[9] hunger1-hunger9;
  array t[9] time1-time9;
  time1=0; time2=0.25; time3=0.5; time4=0.75; time5=1;
  time6=1.5; time7=2; time8=3; time9=4;
  AUC = 0;
  do j=1 to 8; why 8 instead of 9?
    next_trapezoid = 0.5 * (t[j+1] - t[j])*(m[j] + m[j+1]);
    AUC = sum(AUC, next_trapezoid);
  end;
```

How should we adapt this to find maximum hunger score?

34

What if some observations are missing?

Suppose a subject is missing VAS hunger measurement m_2 at time 0.25 hours.

What happens to the AUC calculation in SAS?

$$\text{AUC} = \frac{1}{2} \left\{ (t_2 - t_1)(m_1 + m_2) + (t_3 - t_2)(m_2 + m_3) + (t_4 - t_3)(m_3 + m_4) + \cdots + (t_9 - t_8)(m_8 + m_9) \right\}$$

35

Write code to alert you to problems: write observations with missing data to a separate data set.

To create 2 datasets, give 2 names, and separate output statements.

```
data hunger_AUC missing; create 2 data sets
  set VAS_hunger;
  array m[9] hunger1-hunger9;
  array t[9] time1-time9;
  time1=0; time2=0.25; time3=0.5; time4=0.75; time5=1;
  time6=1.5; time7=2; time8=3; time9=4;
```

36

```

AUC = 0;
do j=1 to 8;
  next_trapezoid = 0.5 * (t[j+1] - t[j])*(m[j] + m[j+1]);
  if (next_trapezoid = .) then do; deal with a missing value
    output missing;
    GOTO Duluth; jump to label 'Duluth'
  end;
  AUC= sum(AUC,next_trapezoid);
end;
output hunger_AUC;
Duluth: SAS label ends with full colon, not semicolon

```

37

```
proc print data=missing;
```

Obs	subject	hunger1	hunger2	hunger3	hunger4	hunger5	...
1	1	81	.	10	15	37	...

Common practice: replace missing k -th value at t_k by linear interpolation from measurements m_{k-1} , m_{k+1} on either side.

Solve for x :

$$\frac{m_{k-1} - x}{m_{k-1} - m_{k+1}} = \frac{t_{k-1} - t_k}{t_{k-1} - t_{k+1}}$$

Use imputation for missing measurements at the ends.

38