History Carried Forward
Future Carried Back:
Mixing Time Series of Differing Frequencies

BASUG: 08MAY2024

Mark Keintz
mkeintz@outlook.com

Mixing Time Series Frequencies
The "Take Home" Points

- SET statement variables are retained over data step iterations - until a SET reading the same variables is executed. (also true of MERGE).

- So conditional SETs (*if xxx then set mydata*) can preserve variables over many iterations, providing a means of carrying forward "historic" data …
- … over a "window".

Mixing Time Series Frequencies
The "Take Home" Points

- For sorted data sets, this is much faster than commonly used PROC SQL approach.

- Using strategically placed "sentinel" variables provides a way to set "stale" data to missing values.

The Problem: How to "carry forward" Low Frequency Data Records

- Start with data set YEAR (one obs per year) and QTR (four per year)
  - Both sorted by ID and DATE
  - Otherwise with mutually exclusive variables

- Task: propagate the yearly data through all subsequent quarterly records preceding next yearly DATE

Sample YEAR Data (2 ID's, 3 Years)

ID	DATE	AstA ($mm)	LbtA ($mm)
XX	30JUN2014	821	281
XX	30JUN2015	799	303
XX	30JUN2016	804	322
YY	31DEC2014	1,401	904
YY	31DEC2015	1,427	950
YY	31DEC2016	1,550	962

Sample QTR Data (showing 1st ID)

ID	DATE	SalQ ($mm)	EmpQ(1,000's)
XX	30JUN2014	132.3	13.5
XX	30SEP2014	128.9	12.6
XX	31DEC2014	138.3	13.5
XX	31MAR2015	112.0	11.3
XX	30JUN2015	115.5	11.7
XX	31SEP2015	140.2	14.2
XX	…	…	…
XX	31MAR2017	98.9	9.9

The Goal: Yearly Data Carried Forward Into Quarterly Series

ID	DATE	YRDATE	AstA	LbtA	SalQ	EmpQ
XX	30JUN2014	30JUN2014	821	281	132.3	13.5
XX	30SEP2014	30JUN2014	821	281	128.9	12.6
XX	31DEC2014	30JUN2014	821	281	138.3	13.5
XX	31MAR2015	30JUN2014	821	281	112.0	11.7
XX	30JUN2015	30JUN2015	799	303	115.5	14.2
XX	31SEP2015	30JUN2015	799	303	140.2	10.6
XX	…	…	…	…	…	…
XX	31MAR2017	31JUN2016	804	322	98.9	9.9

Match-Merge (MERGE + BY) Simple and Compact …

```
data match_merge_yq;
  merge year qtr;
  by id date;
run;
```

… and Easily Scaled …

```
data match_merge_yqm;
  merge year qtr month;
  by id date;
run;
```

… but …

Match-Merge Results
Does Not Carry Data Forward

ID	DATE	AstA	LbtA	SalQ	EmpQ
XX	30JUN2014	821	281	132.3	13.5
XX	30SEP2014	.	.	128.9	12.6
XX	31DEC2014	.	.	138.3	13.5
XX	31MAR2015	.	.	112.0	11.7
XX	30JUN2015	799	303	115.5	14.2
XX	31SEP2015	.	.	140.2	10.6
XX	…	…	…	…	…
XX	31MAR2017	.	.	98.9	9.9

PROC SQL can Carry Data Forward But it's Expensive

```
proc sql noprint;
  create table yrqtr as
  select *
    from qtr
  left join
    year (rename=(date=yrdate))

  on year.id=qtr.id

  and yrdate<=date<intnx('year',yrdate,1,'s')
  order by id,date;
quit;
```

Merge + Conditional SET Statements Faster and Simpler

```
data yrqtr;
  merge YEAR (in=iny keep=id date)
        QTR (in=inq keep=id date) ;
  by id date;

  /*Conditional SET of ALL the year vars*/
  if iny then set YEAR (rename=(date=yrdate));

  /*Conditional Set of all the quarter vars */
  if inq then set QTR  (rename=(date=qtrdate));

  if inq then output;

  /* Don't carry data across ID boundaries*/
  if last.id then call missing(of _all_);
run;
```

Merge + Conditional SET Statements Easily Scaled to 3 or More Series

```
data YQM ;
  merge YEAR  (in=iny keep=id date)
        QTR   (in=inq keep=id date)
        MONTH (in=inm keep=id date);
  by id date;

  if iny then set YEAR (rename=(date=YR_date));
  if inq then set QTR  (rename=(date=QTR_date));
  if inm then set MONTH (rename=(date=MON_date));

  if inm then output;
  if last.id then call missing(of _all_);
run;
```

What About Low Freq Series Holes? (e.g. Missing 30JUN2015 for ID XX)

ID	DATE	AstA ($mm)	LbtA ($mm)
XX	30JUN2014	821	281
XX	30JUN2016	804	322
YY	31DEC2014	1,401	904
YY	31DEC2015	1,427	950
YY	31DEC2016	1,550	962

What About Low Freq Series Holes? MERGE + Conditional SET → "Stale" Data

ID	DATE	YRDATE	AstA	LbtA	SalQ	EmpQ
XX	30JUN2014	30JUN2014	821	281	132.3	13.5
XX	30SEP2014	30JUN2014	821	281	128.9	12.6
XX	31DEC2014	30JUN2014	821	281	138.3	13.5
XX	31MAR2015	30JUN2014	821	281	112.0	11.7
XX	30JUN2015	30JUN2014	821	281	115.5	14.2
XX	31SEP2015	30JUN2014	821	281	140.2	10.6
XX	…	…	…	…	…	…
XX	31MAR2017	31DEC2016	804	322	98.9	9.9

What Abouts Low Freq Series Holes? Requirements for Eliminating Stale Data

- DETECT when data has become stale
  - (i.e. compare YR_date to date)

- SET TO MISSING all the stale variables
  - Tip: Assemble Program Data Vector such that stale variables are contiguous, which enables
    CALL MISSING(of first_stale_var -- last_stale_var);

- ESTABLISH VARIABLE NAMES for call missing routine
  - Make **SENTINEL VARIABLES** positioned just before and after the variables at risk of becoming stale

Using Sentinel Variables to Prevent Stale Historic Data

```
data YQTR_holes (drop=_sentinel:);
  merge YEAR  (in=iny keep=id date)
        QTR   (in=inq keep=id date);
  by id date;

  retain _sentinel1 .;
  if iny then set YEAR (rename=(date=YR_date));
  retain _sentinel2 .;

  if inq then set QTR (rename=(date=QTR_date));

  if YR_date^=. And intck('qtr',YR_date,QTR_date)>3
    then call missing(of _sentinel1--_sentinel2);

  if inq then output;
  if last.id call missing(of _all_);
run;
```

Sample Yearly Data With Missing Year Data for 30JUN2015

ID	DATE	YRDATE	AstA	LbtA	SalQ	EmpQ
XX	30JUN2014	30JUN2014	821	281	132.3	13.5
XX	30SEP2014	30JUN2014	821	281	128.9	12.6
XX	31DEC2014	30JUN2014	821	281	138.3	13.5
XX	31MAR2015	30JUN2014	821	281	112.0	11.7
XX	30JUN2015	.	.	.	115.5	14.2
XX	31SEP2015	.	.	.	140.2	10.6
XX	…	…	…	…	…	…
XX	31MAR2017	31DEC2016	804	322	98.9	9.9

Mixing Irregular Data Series

- The Goal: Mix Three Irregular Series
  - Admit/Discharge
  - Services
  - Tests
- Output One Record for Each Event
- Each Record contains most recent data from each source

Irregular Series Data Set Admit/Discharge

ID	DATE	ACTION	ACTIONMD
101	15JAN2015	A	X2
101	20JAN2015	D	C5
102	15FEB2014	A	X1
102	25FEB2014	D	C9

Irregular Series Data Set Services

ID	DATE	SVCID	CHGUNITS
101	18JAN2015	323	3.2
101	19JAN2015	488	1.2
102	15FEB2014	101	3.0
102	16FEB2014	229	1.7

Irregular Series Data Set
TESTS

ID	DATE	TSTLAB	TSTTYPE
101	21JAN2015	788	VIS
102	15JAN2015	823	HRT

Mixing Irregular Series

```
data want;
  merge admdis (in=inA keep=id date)
        srvcs  (in=inS keep=id date)
        tests  (in=inT keep=id date);
  by id date;
  if inA then set admdis (rename=(date=date_A));
  if inS then set srvcs  (rename=(date=date_S));
  if inT then set tests  (rename=(date=date_T));

  output;   /* No IF test needed*/

  if last.id then call missing (of _all_);
run;
```

Mixing Irregular Series: Results

ID	DATE	A/D	AID	SID	CHG	LAB	LTYPE
101	15JAN2015	A	X2				
101	18JAN2015	A	X2	323	3.2		
101	19JAN2015	A	X2	488	1.2		
101	20JAN2015	D	C5	488	1.2		
101	21JAN2015	D	C5	488	1.2	788	VIS
102	15FEB2014	A	X1	101	3.0	823	HRT
102	15FEB2014	A	X1	229	1.7	823	HRT
102	25FEB2015	D	C9	229	1.7	823	HRT

What about Next Observation Carried Back?

- So far, only LOCF (Last Observation Carried Forward) tasks. What about NOCB? How to carry the future back?

- Intuitive, but expensive:
  - Reverse each dataset sort order
  - Run the LOCF logic
  - Re-sort the results to original order

- Or …

What about Next Observation Carried Back?

- Recognize that NOCB, like LOCF, simply assigns a date window to each observation.
- Consider the YEAR record for 30JUN2015
  - For LOCF, window is 30JUN2015 -- 29JUN2016
  - But for NOCB, it is 01JUL2014 -- 30JUN2015
- This simplifies the NOCB task: simply establish the appropriate window start date.
  - NO SORTING REQUIRED

What about NOCB?
Make a “window start date”

```
/*Create windows start date for each obs in YEAR*/
data YCB / view=YCB ;
  set year;
  by id;

  /* Each window starts one day after prior date */
  _wbeg = sum(lag(date),1);

  /* Begin each ID with arbitrary window start*/
  if first.id then _wbeg='01jan1900'd;
run;
```

What about NOCB?
Use the Windows Start Date, but
LOOK OUT!

```
data nocb_yq (drop=_:) ;
  merge
    YCB (in=iny keep=id _wbeg)
    QTR (in=inq keep=id date rename=(date=_wbeg));
  by id _wbeg;

  if iny then set YCB (rename=(date=yrdate));

  if inq then set QTR (rename=(date=qtrdate));

  if inq then output;
  if last.id then call missing(of _all_);
run;
```

What about NOCB?
Guard Against Passing the Future

ID	QTRDATE	YRDATE	AstA	LbtA	SalQ	EmpQ
XX	30JUN2014	30JUN2014	821	281	132.3	13.5
XX	30SEP2014	30JUN2014	821	281	128.9	12.6
XX	…	…	…	…	…	…
XX	30SEP2015	30JUN2016	804	322	140.2	14.2
XX	…	…	…	…	…	…
XX	30JUN2016	30JUN2016	804	322	127.8	13.0
XX	30SEP2016	30JUN2016	804	322	130.0	12.9
XX	31DEC2016	30JUN2016	804	322	120.6	11.8
XX	31MAR2017	30JUN2016	804	322	98.9	9.9

What about NOCB?
Future Expired? Delete it.

```
data nocb_yq (drop=_:) ;
  merge
    YCB (in=iny keep=id _wbeg)
    QTR (in=inq keep=id date rename=(date=_wbeg));
  by id _wbeg;

  if iny then set YCB (rename=(date=yrdate));

  if inq then set QTR  (rename=(date=qtrdate));
  if qtrdate < yrdate then call missing(astA,lbtA);

  if inq then output;
  if last.id then call missing(of _all_);
run;
```

Revisit the Main Points

- Conditional SETs allow observations to be retained over multiple DATA step iterations.
- This allows easy LOCF for sorted data sets.
- Easily expanded to several data sets.
- And easily modified to execute NOCB without re-sorting data.

Questions?

Mark Keintz
mkeintz@outlook.com