



»» TRANSFORMING PROMISING IDEAS INTO COMMERCIAL REALITY

Getting the most of out of keys – using Hash objects to select, subset, or summarise

Pamela Giese

Principal Statistical Programmer

InVentiv Health Clinical





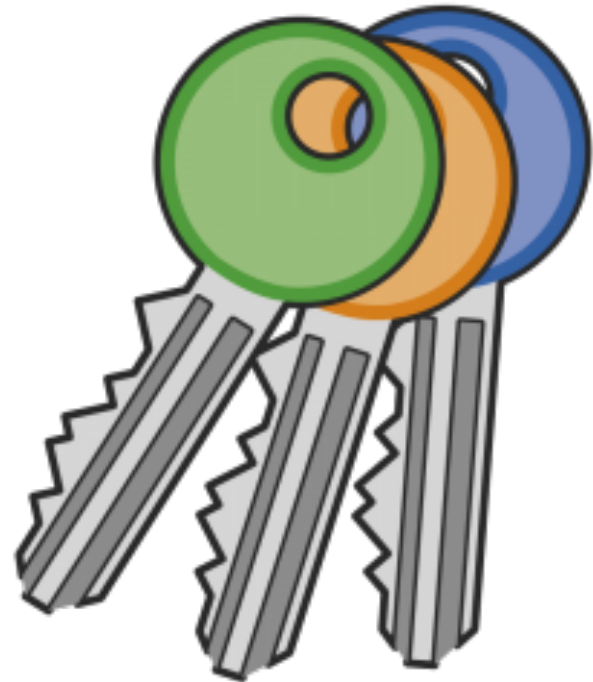
What are keys?

- We know keys via the BY statement

- > Proc Sort data=myfile;
- > BY trialno subject visit;
- > Run;

- > Data newfile;
- > Merge myfile otherfile;
- > BY trialno subject visit;

- > Proc Freq data=newfile;
- > BY trialno subject visit;
- > Table bodyweight ;





What are keys?

- We know keys via the BY statement
 - > %let KEYS= trialno subject visit;
 - > Proc Freq data=myfile;
 - > BY &keys;
 - > Table bodyweight;



Keys can be problem with big data

- In Base SAS, keys are created via a PROC SORT
 - Big data requires
 - TIME
 - SPACE
 - RESOURCES
 - Other procedures require data to be sorted
- Even PROC SQL in SAS, using ORDER and GROUPBY creates similar problems



Keys in dataset

Hash Objects enables you to handle data using dynamic keys within temporary datasets.



Hash Components

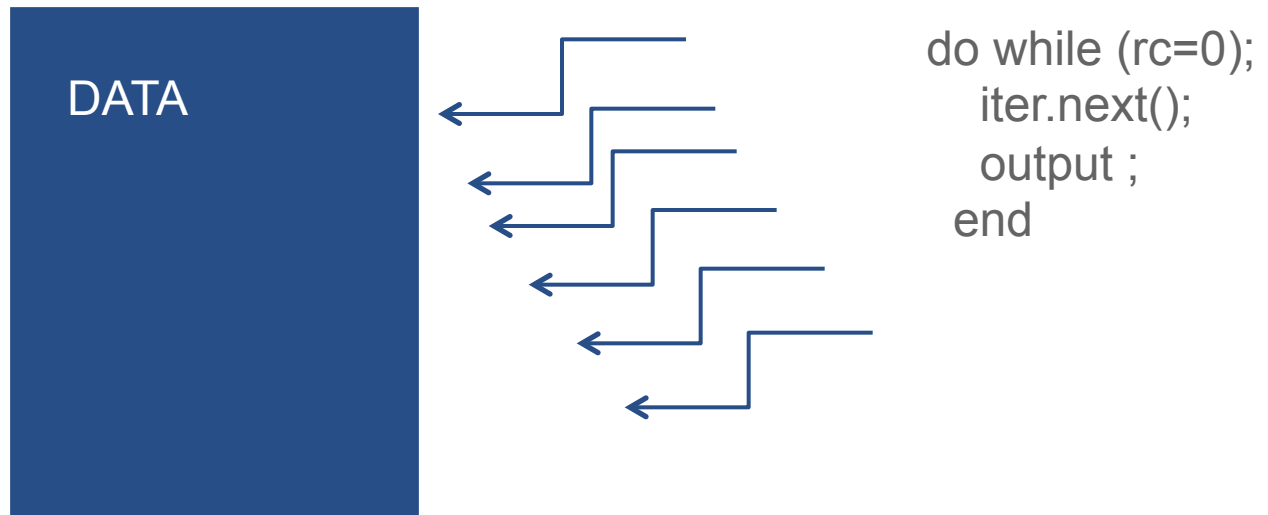
- Option 1: Using separate files for keys and data.





Hash Components

- Option 2: Use hash iterator
 - > Keys are defined in the hash iterator
 - > A hash iterator (iter) works as a pointer to move through the dataset.





Example 1: Look-up file

Problem: The VISIT dataset with 901128 records with keys of TRIALNO and VISIT. There is a “lookup” dataset which contains visit descriptions. This is VISDESC, again with keys TRIALNO and VISIT and consists of 1049 records. The goal is to merge these two files together to pick up the visit label VISITSP

Method 1: Proc SQL

```
proc sql;  
    create table sqlallvis as  
    select a.*, b.visitsp from visit as a, visdesc as b  
        where a.trialno = b.trialno and a.visit = b.visit;  
quit;
```



Example 1: Look-up file

Method 2: Merge

```
proc sort data=visdesc out=visitdesc (keep=trialno visit visitsp);  
    by trialno visit;  
RUN;  
proc sort data=visit out=visit;  
    by trialno visit;  
RUN;  
data oldallvis;  
    merge visitdesc(in=in1 ) visit(in=in2);  
    by trialno visit;  
    if in1 & in2;  
RUN;
```



Example 1: Look-up file

Method 3: HASH code

```
data allvis;
```

```
length visitsp $60;
```

```
if _n_ = 1 then do;
```

```
  declare hash visdes(hashexp:8, Dataset: 'visdec');
```

```
  visdes.DefineData('visitsp');
```

```
  visdes.definekey('trialno','visit');
```

```
  visdes.definedone();
```

```
  call missing(visit, visitsp);
```

```
END;
```

```
set visit;
```

```
rc=visdes.find(key:trialno, key:visit);
```

```
if rc = 0 then do;
```

```
  output allvis;
```

```
END;  RUN;
```

Look-up Dataset

Return Code





Example 1: Look-up file

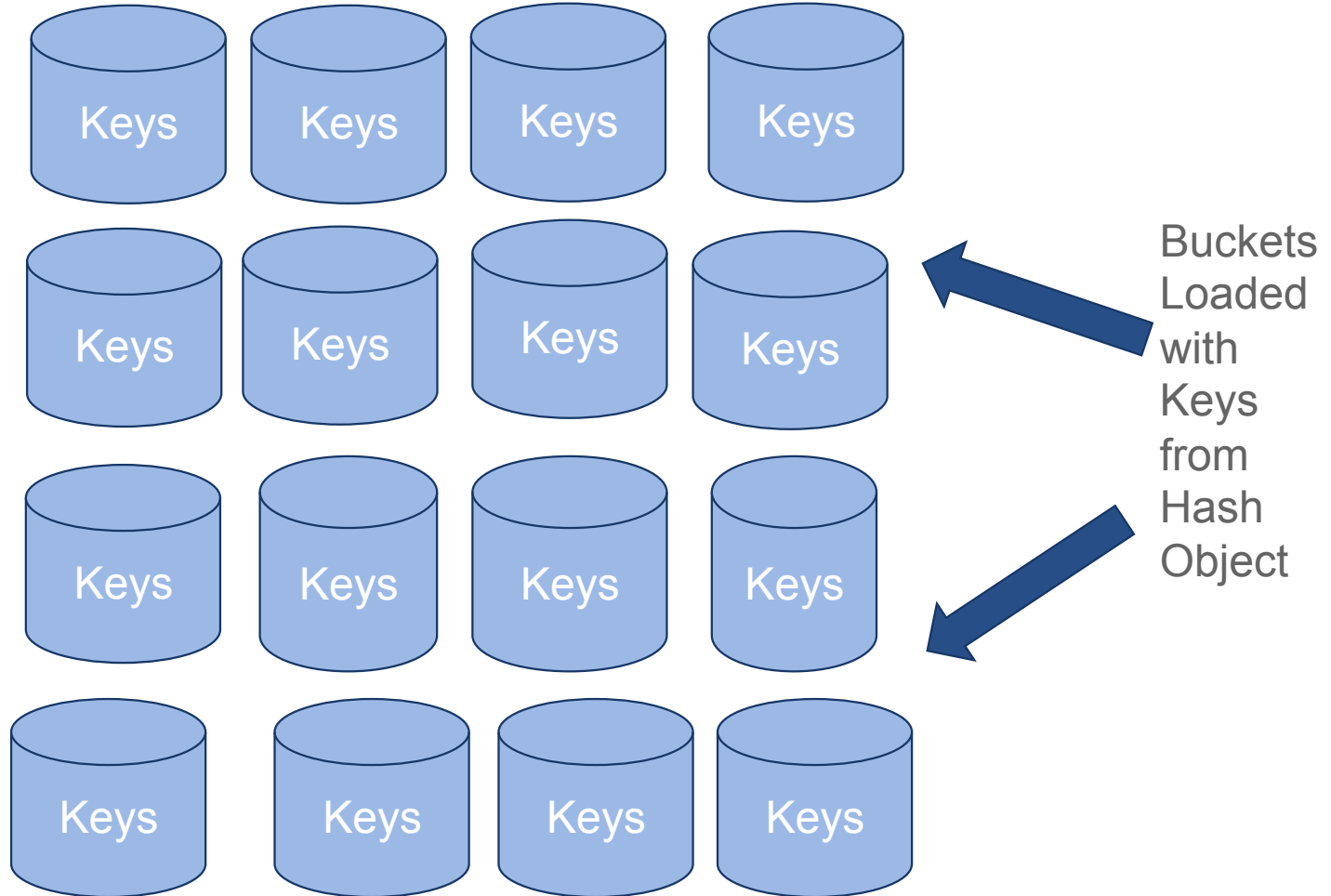
Results (all final datasets were equal)

Method	Real time (seconds)	CPU time (seconds)
Hash Objects	7.04	4.03
SQL	10.16	4.86
Total Merge (including sorts)	17.13	9.35

Reminder: Base dataset contained 901128 records with keys of TRIALNO and VISIT. The “lookup” contained 1049 records.



Hashexp represents the number of “buckets”





HASHEXP parameter

```
declare hash visdes(hashexp:8, Dataset: 'visdesc');
```

- The size of the internal hash table of represents 2^n where $n=\text{hashexp}$.
- This doesn't equate to number of records.
- Can vary up to 20

HASHEXP	Real time (seconds)	CPU time (seconds)
8	7.04	4.03
10	7.83	3.99
12	9.48	3.41
16	11.23	4.03
20	10.24	4.14



Example 2: Subset

Problem: Select all records where VALUE_C $\geq$ 11.5

Method 1: Dataset

```
data datasetselect;  
    set wbc;  
    if value_c  $\geq$  11.5;  
RUN;
```

Method 2: SQL

```
proc sql;  
    create table sqlselect as  
        select * from wbc where value_c  $\geq$  11.5;  
quit;
```



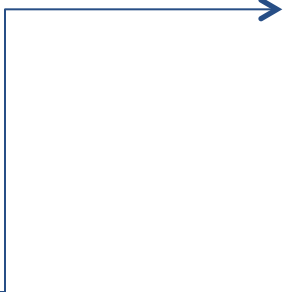
Example 2: Subset

Method 3: Hash

```
data hashhi;  
length country $2 trialno patno value_c 8 ;  
declare hash myhash(hashexp: 4, dataset:"work.wbc", multidata:"yes",  
    ordered:'yes');  
myhash.DefineData ('trialno', 'country','patno', 'value_c' );  
declare hiter iter('myhash');  
myhash.defineKey ('value_c');  
myhash.defineDone();
```

```
do while (not done);  
set wbc end=done;  
myhash.find();  
end;
```

```
done = 0
```



```
rc = iter.setcur(key: 11.5);  
do while (rc=0);  
    iter.next();  
    output ;  
end;  
run;
```



Example 2: Subset

Results (all final datasets were equal)

Method	Real time (seconds)	CPU time (seconds)
Hash Objects	0.93	0.78
SQL	0.08	0.06
Data Step	0.11	0.11

Original dataset WBC had 258878 observations. Resulting dataset had 22080.



Example 3: Summarising

Problem: Calculate the number of days that a patient is on a concomitant medication (a medication in addition to the study drug), saving the last date.

Complications:

- Program needs to retain flexibility over different situations
- Different medications can be taken simultaneously
- Interruptions of dosing may occur
- Specific types of medications may have different windowing rules
- Result is a huge file with daily snapshot of medications per length of study analysis.



Example 3: Summarising

Contains 124,663,088
obs. Real time : > 1 hour 47
minutes; CPU time : > 30
minutes

- Initial Solution:

```
proc sort data=patyr_drug;  
  by calcno cmno subno sublabel subcode subvalue refint treatmgr  
  cmny _valid trial country trunit patno obs_dt;  
run;  
data patyr_drug;  
  set patyr_drug;  
  by calcno cmno subno sublabel subcode subvalue refint treatmgr  
  cmny _valid trial country trunit patno;  
retain onset_days;  
if first.patno then onset_days=1;  
else      onset_days+1;  
if last.patno;  
run;
```



Example 3: Summarising

Using Hash

Create Key file:

```
proc sort data=patyr_drug out=patkeys nodupkey;  
  by calcno cmno subno sublabel subcode subvalue refint treatmgr  
     cmny _valid trial county trunit patno ;  
RUN;
```

Still requires a PROC SORT , but NODUPKEY reduces time
(time for Proc Sort: 34 minutes Real, 15 minutes CPU)

PATYR_DRUG

P
A
T
K
E
Y
S



Example 3: Summarising

Duplicate: 'r' replaces the previous key so the last observation is kept

Using Hash (continued)

data onset;

```
declare hash myhash (hashexp:20, suminc:"count", Dataset: '_patyr_drug',  
duplicate: 'r');
```

```
rc=myhash.definekey('calcno', 'cmno', 'subno', 'sublabel', 'subcode',  
'subvalue', 'refint', 'treatmgr', 'cmny', '_valid', 'trial', 'county', 'trunit', 'patno');
```

```
myhash.definedata('obs_dt');
```

```
myhash.definedone();
```

```
count = 1;
```

```
do while (not done);
```

```
set _patyr_drug end=done;
```

```
myhash.find();
```

```
end;
```

```
done=0;
```

```
do while (not done);  
    set patkeys end=done;  
    myhash.sum(sum:onset_days);  
    output;  
END;  
stop;  
RUN;
```



Example 3: Summarizing

Results (all final datasets were equal)

Method	Real time (seconds)	CPU time (seconds)
Hash Object: Key Sort	34:41	14:46
Hash Object: Dataset	1:16:56	25:45
Total Hash	1:51:37	40:31
Original: Full Data Sort	53:29	18:56
Original: Dataset	1:47:44	30:27
Total Original	2:41:13	49:31

Original dataset PATYR_DRUG had 124,663,088 observations..



HASH Objects Summary

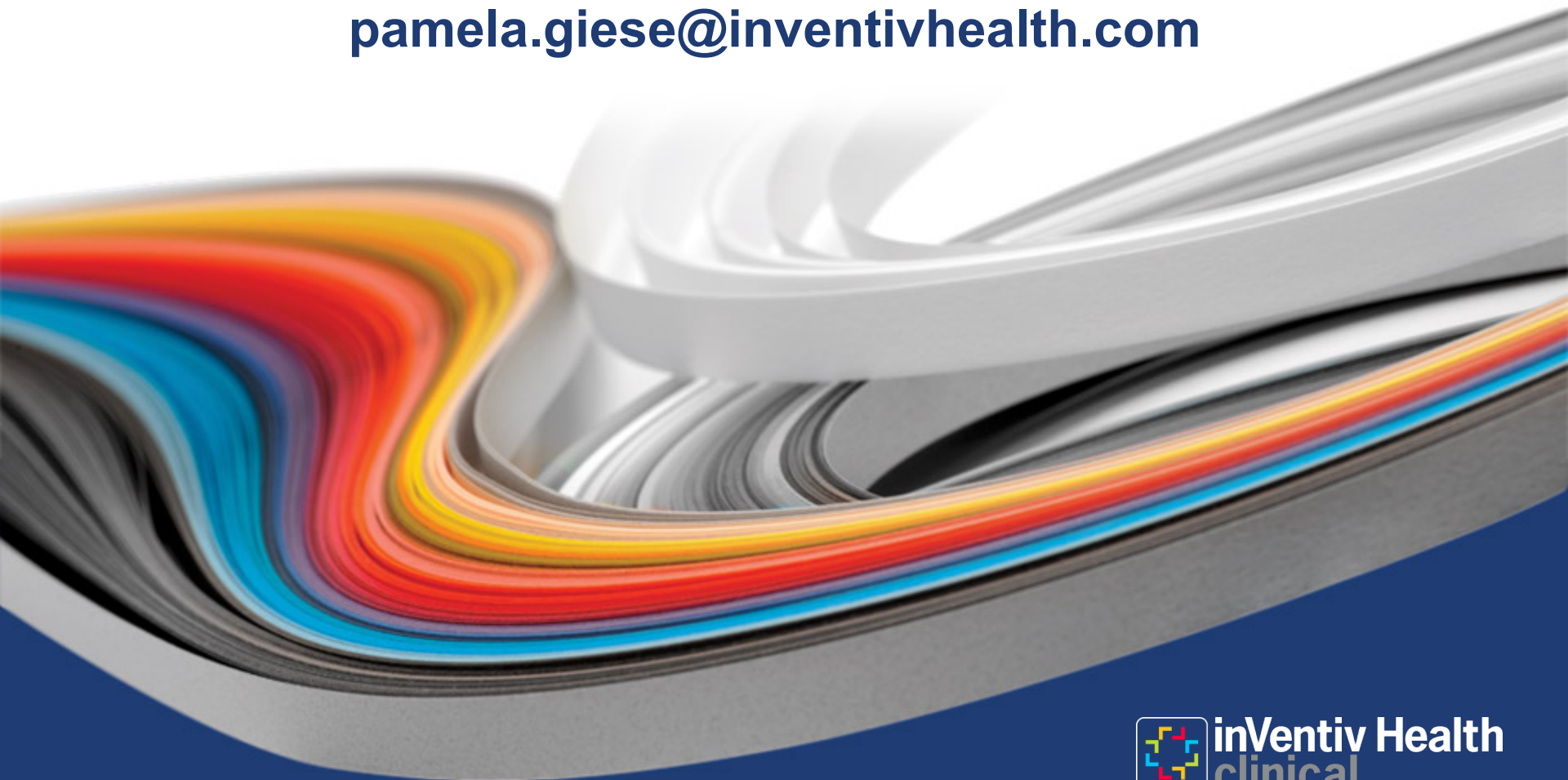
- Pro
 - > Can reduce time
 - > Can eliminate need for Proc Sort
 - > Lends itself to big data
- Con
 - > Coding not intuitive
 - > All data points need to be defined



Contact:

Pamela Giese

pamela.giese@inventivhealth.com





More from the PhUSE Community

Recommended reading

From the 2012 PhUSE conference, Coding Solutions stream:

CS05

Re-programming a many-to-many merge with Hash Objects

by

David J Garbutt, GarbuttConsult, Basel, Switzerland



Be Smart with Keys

- Generally, Key order should follow data hierarchy
- Within a program, keys should follow function
 - > The fewer the better
 - > Save listing order until last