

PROC COMPARE no longer enough to save the day on its own!

Eoin Byrne, Plus Project Partnership, eoin.byrne@plus-project.co.uk



Introduction

Statistical Analysis relies heavily on quality of data (paramount to ensure patient safety)
Greater emphasis on the validation = trust in the analysis.

When using PROC COMPARE in SAS too many focus on the "key words":

```
Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 26.
NOTE: No unequal values were found. All values compared are exactly equal.
```



HOWEVER!

Programming for derivations may be vastly different yet yield the same results for the wrong reason. (Data is seen as clean but data's integrity is in question, but missed by compare)



should

What additional methods can we implement alongside PROC COMPARE?

Benefits of PROC COMPARE

PROC COMPARE is an essential tool for validation or checking changes between updated datasets. A compare can be produced with not much more than

Original / base dataset New / updated dataset

```
proc compare base=previous comp=new; run;
```

The addition of ID variables allows us to specify by which variables we want to compare the dataset.

Important to use with ID as records might not match on ID's.

```
proc compare base = prod_adv
compare = qc_adv listall;
id studyid usubjid paramcd avisit ;
run;
```



Have you checked that these are the correct ID variables, i.e. unique?

We can use WITH and VAR statements to allow us to compare a data set with variables that have different names in the comparative data set.

```
data base_data;
do i = 1 to 200;
predict=round(rand("uniform")*5,1);
output;
end;
run;
```

Randomly create 200 results between 0 and 5 for our predicted result

```
data act_data;
set base_data;
rename predict=actual;
run;
```

Renamed in a separate dataset as actual and compared against each

```
proc compare base=base_data comp=act_data;
var predict;
with actual;
run;
```

All results matching between our predicted and actual results!



```
Number of Observations in Common: 200.
Total Number of Observations Read from WORK.BASE_DATA: 200.
Total Number of Observations Read from WORK.ACT_DATA: 200.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 200.
NOTE: No unequal values were found. All values compared are exactly equal.
```

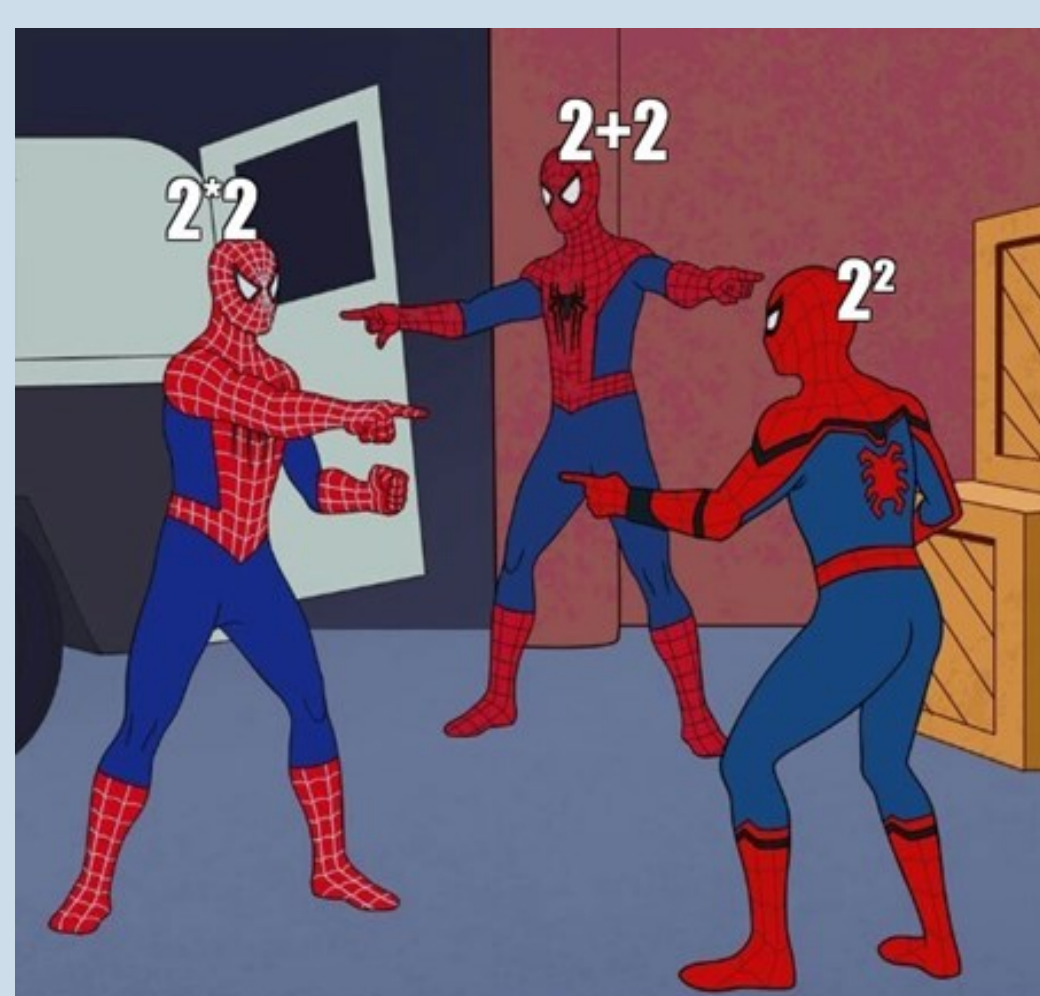
Drawbacks

Derived results are the most common example of where results can be the same, but the derivation can give vastly different results depending on the inputted data.

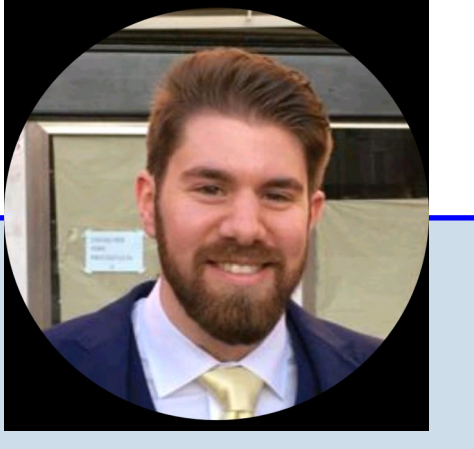
New data

Same production/validation programs

Different results!



Peter Crumey, Plus Project Partnership, peter.crumey@plus-project.co.uk



Programmatic Review

One method that could be applied at early stages of our ADaM programming is to run both the production and the QC Code over a vast dataset of generated results (often referred to as UAT data) that is then saved into a reference area that both sides can access, similar to the example of WITH and VAR statements previously. We need to ensure that when we have produced our random generator we save the outputted data set otherwise we will keep producing new results every time.

An example of a potentially complex derivation that we could use is the CKD_EPI Equation for estimating Glomerular Filtration rate. The equation is :

$$GFR = 141 \times \min(S_{cr}/\kappa, 1)^{\alpha} \times \max(S_{cr}/\kappa, 1)^{-1.209} \times 0.993^{Age} \times 1.018 [if female] \times 1.159 [if black]$$

where:

S_{cr} is serum creatinine in mg/dL,

κ is 0.7 for females and 0.9 for males,

α is -0.329 for females and -0.411 for males,

min indicates the minimum of S_{cr}/κ or 1, and

max indicates the maximum of S_{cr}/κ or 1.

First we have our randomly generated set of results, which will allow us to then apply this to our derivation.

```
data randgen_egfr;
do unique_id = 1 to 1000;
scr = round((rand("uniform")*4.75),0.01)+0.25; /*Greater than average range of serum creatinine in mg/dL*/
asexn = round(rand("uniform"),1)+1; /*numeric Version of sex: Female=1, Male=2*/
aracen = round(rand("uniform"),1)+1; /*numeric Version of race: African American=1, White/Other=2*/
age = round((rand("uniform")*67),1)+18; /*CKD_EPI requires a minimum age of 18 range to 85*/
output;
end;
run;
```

```
data egfr_deriv_qc ;
set randgen_egfr;
if asexn=1 and aracen=1 then egfr=141*(min((scr/0.7),1)**-0.329)*max((scr/0.7),1)**-1.209*(0.993**age)*1.018*1.159;
else if asexn=1 and aracen=2 then egfr=141*(min((scr/0.7),1)**-0.329)*max((scr/0.7),1)**-1.209*(0.993**age)*1.018;
else if asexn=2 and aracen=1 then egfr=141*(min((scr/0.9),1)**-0.411)*max((scr/0.9),1)**-1.209*(0.993**age)*1.159;
else if asexn=2 and aracen=2 then egfr=141*(min((scr/0.9),1)**-0.411)*max((scr/0.9),1)**-1.209*(0.993**age);
```

```
if aracen=1 then arace='African American';
else if aracen=2 then arace='White/Other';
else put "WARN" "ING: code check randgen derivations, result falls outside of required parameters" aracen;
```

```
if asexn=1 then asexn='Female';
else if asexn=2 then asexn='Male';
else put "WARN" "ING: code check randgen derivations, result falls outside of required parameters" asexn;
```

Do the same for egfr_deriv_prod and include id unique_id arace asexn in PROC COMPARE

run;

unique_id	arace	asexn	Base egfr	Compare egfr	Diff.	% Diff
24	White/Other	Female	128.9067	118.6771	-10.2296	-7.9357
31	White/Other	Female	162.4637	149.5711	-12.8926	-7.9357
41	African American	Male	172.5485	191.3241	18.7756	10.8813
65	African American	Male	122.3881	135.7056	13.3174	10.8813
67	African American	Male	120.1741	133.2507	13.0765	10.8813
73	White/Other	Female	120.1598	110.6243	-9.5355	-7.9357
100	African American	Male	129.6068	138.9335	9.3267	7.1962
128	White/Other	Female	87.8881	86.9133	-0.9748	-1.1092
129	White/Other	Female	117.0670	107.7770	-9.2900	-7.9357
176	White/Other	Female	112.7272	104.7479	-7.9793	-7.0794
185	African American	Male	128.5649	142.5545	13.9896	10.8813
226	White/Other	Female	140.9940	129.8052	-11.1888	-7.9357
229	African American	Male	174.7642	193.7809	19.0167	10.8813



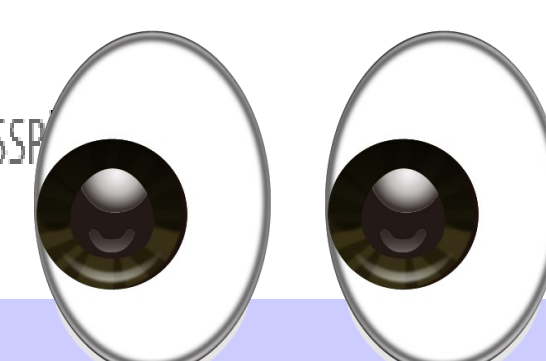
Adding ID makes it easier to see individual records than just obs number

Here we have 54 cases of mismatching data between the QC and Prod derivations. With the IDs added it is easy to spot the pattern.

Code Review

Another way to ensure that the derivations are being done as per the specs is to apply an independent code review. This would involve assigning a separate programmer who hasn't worked on either the production or validation of the program to review the code of the production program and check that all derivations are being done as per the spec. Reviewer can also compare against specs created by define.xml, and SAPI

```
data qc_ddata (keep = siteid cbpval1 usubjid subjid dsstdtc st_col1);
length cbpval1 $108 st_col1 $200 ;
set adds t;
cbpval1 = 'Site Id.: '||strip(siteid);
st_col1 = strip(SFRES)||'/'||strip(SFRESSP);
run;
```



Unique Subject Id.	Subject Id.	Date of Screen/Run-In Failure	Type of Failure	Reason (Term(s) for Screen/Run-In Failure/Specify Test)
05K123456.00105	000001	2012-04-05	Screen	DID NOT MEET INCLUSION/EXCLUSION CRITERIA
05K123456.00109	000001	2012-04-05	Screen	PHYSICIAN DECISION: CHANGED MIND DID NOT MEET INCLUSION/EXCLUSION CRITERIA
05K123456.002	000001	2012-04-05	Run-In	ADVERSE EVENT
05K123456.00001	000001	2012-04-05	Run-In	PROTOCOL DEVIATION: SUBJECT DID NOT STOP PROHIBITED MEDICATIONS
05K123456.00104	001003	2012-04-05	Screen	DID NOT MEET INCLUSION/EXCLUSION CRITERIA
			Screen	WITHDRAWAL BY SUBJECT/CHANGED MIND

Conclusion

While PROC COMPARE is a strong time-saving procedure with many options that can be included in the code, it is important to use a variety of validation tools and follow a risk-based approach:

Higher risk → More safeguards put in place
Lower risk (e.g. not complex derivation) → Less safeguards put in place