PhUSE 2015

Paper PD09

Industry Standard for Good Programming Practice: Benefits and Implementation

Mark Foxwell, PRA, Reading, UK
Kate Peacock, Quintiles, Reading, UK

ABSTRACT

Programming in the Pharmaceutical Industry is subject to growing pressures to deliver to tighter timelines and to increase efficiency. Two consequences of this are a need for programmers to get up to speed quickly and to understand and reuse code developed for other studies or outputs. An industry standard Good Programming Practices guidance will ensure that all programmers follow a standard that is easy to read and maintain over time and develop code that can be shared between companies and regulators with ease.

The PhUSE GPP steering board has developed a GPP guideline targeting these aims, and with the commitment from the industry to implement them and share best practices, we can all benefit and focus on the important, complex and interesting tasks.

INTRODUCTION

Ensuring that Good Programming Practice (GPP) is followed is more important now than ever before. Increased focus on cost efficiency and ever more stringent timelines mean that the ability of a team to produce logical, easily transportable code is paramount. Fostering an environment where staff are encouraged and supported to follow GPP in all their coding will potentially lead to a more efficient, content team with far greater flexibility in resourcing projects.

The PhUSE GPP steering board have done some research into the different methods that companies use to implement GPP guidelines and this paper will discuss some of those methods and look at the benefits of implementing GPP.

A BRIEF OVERVIEW OF GPP

So when we talk about GPP what do we mean? GPP is essentially a set of rules which when followed lead to more robust, logical programs which are easier and more efficient to maintain. Some of the key rules are:

- Include a program header
    - This should provide basic documentation for the program including its purpose, file location, details of any output and author. It should also include a revision history
- Include comments
    - Only include informative comments, e.g. those which help clarify the aim of certain sections of code or those which help document how data is being processed
- Flow of code should be logical
    - Read data in (try to avoid repetitive sorting of the same data)
    - Process (e.g. derive variables or create summary statistics)
    - Create output (e.g. dataset, table)
    - Tidy up – e.g. delete any temporary datasets created, reset options
- Follow consistent naming conventions
    - Use industry standards (eg CDISC, ADaM) where applicable
    - Program naming conventions should make it possible to identify groups of related programs such as analysis dataset programs vs. output programs
- Follow consistent coding conventions, e.g.
    - Do not overwrite existing datasets or files

    - Use meaningful temporary dataset names and do not consistently overwrite the same temporary dataset
    - Use `"data=dataset"` option in procedure statements so that the dataset being used is explicitly stated to ensure that the statement will work if it is moved to another location.
    - Write one statement per line
    - Use consistent indentation, blank rows and comments to arrange the program in a clear structure.
    - Use `RUN;` or `QUIT;` at the end of each data/proc step
    - When converting character variables to numeric or vice versa, use the `PUT` and `INPUT` functions to explicitly convert the variable to ensure that it is done in the way intended
    - Program defensively – make sure that all options are accounted for to avoid later updates to the program
- Check log files
    - Not only for ERROR and WARNING messages but other NOTE's that can indicate a problem (e.g. Merge with repeats of BY variables)

This is not an extensive list however these and more are covered in more detail in the Good Programming Practice Guideline put together by the PhUSE GPP Board.

IMPLEMENTATION REQUIREMENTS

When considering how GPP will be implemented what needs to be taken into account?

GUIDANCE

Prior to implementation of GPP, guidance needs to be set up describing the rules which will be followed. Each organization may have their own version of the guidance in order to be able to include specific information in the guidance (for example GPP may include references to standard macros that may be in place or standard processes that are used to perform certain tasks). At a minimum, the guidance should cover the topics listed in the brief overview section above.

If no guidance currently exists the GPP Guideline created by the PhUSE GPP Board can be implemented or used as a starting point for creating organization-specific guidance. If guidance is already in place we would recommend checking it against the PhUSE GPP Board guideline to make sure all guidance within this document is covered. Using the PhUSE GPP Board guideline as a minimum standard will help to ensure different organizations are following the same baseline set of rules which will make transfer of code between organizations (e.g. CRO/Pharma/Regulatory) easier and in an expected format.

FRAMEWORK

Once guidance is in place, a framework needs to be set up which allows the guidance to be implemented and compliance monitored. Various methods are being used in different organizations and these are discussed below

<u>TRAINING</u>

Once guidance is in place, teams can be trained on how to apply this to their day to day coding and have access to it for reference moving forward. Training should be the cornerstone of any implementation method and should be structured to allow maximum benefit including sufficient time for questions. Incorporating examples of "good" and "bad" programs can help highlight the rules being discussed and show them in more practical terms. More informal training sessions at a local level may encourage more discussion than large global sessions. For ongoing support consider having GPP subject matter experts.

Ensuring that everyone attends training is the foundation for implementing GPP but does not allow compliance to be monitored. In order to gain most benefit, training should always be used in conjunction with one of the methods below

<u>SOPS</u>

SOPs describe how we should be performing the work we do and generally everyone would need to read and formally acknowledge each applicable SOP for their role. Including reference to following GPP in SOPs creates a more official process as rather than being a best practice to be followed after receiving training, it is now a practice that is mandated by the guidelines that govern how we do business.

Compliance would need to be monitored by some form of code review (see below) or by undertaking regular internal audits of studies.

CODE REVIEW

There are several benefits of undertaking code review to both help implement and assess compliance with GPP:

- It will help individuals develop their programming abilities through structured feedback
- It allows an assessment to be made of whether programs comply with the GPP guidance in place
    - If not, why not? Are there common themes? Can more targeted training be given to focus on these?
- It provides support - regular feedback and follow-up will help the general understanding of GPP and improve implementation

When should code review be performed?

- At regular intervals throughout the year to assess whether previous feedback is being implemented and determine any further training needs. Experience should not affect whether code review is needed – bad habits can be picked up at any time!
- Consider the work that individuals are performing – if assigned to a long-term study there may be less opportunity to produce new code regularly
- When new tasks are undertaken, e.g. beginning dataset programming (generally more applicable to less experienced staff)

Who is best placed to perform the review?

- Managers – will depend on programming ability, managers who are no longer involved in day-to-day programming tasks may not be able to provide the support needed
- Programming team leads – will have easy access to the project code but may struggle to fit code review in alongside their project responsibilities. At busy times, code review will likely be seen as less of a priority
- A "Code Review" team – creating a team of suitably qualified programmers may help to ensure that there is always some availability although they may not necessarily have access to all the code
- In all cases, ensure that people are involved who are keen help others develop their code as this will motivate them to fit code review in alongside their other responsibilities

Other considerations

- How to choose code for review
    - Ask people to submit code – easiest method as code is then supplied and avoids any issues with the reviewer having access to code in project folders. However people would always be inclined to submit their "best" code for review
    - Select code at random – would require input from the project team leads to help determine what each individual worked on. This method would reinforce GPP being applied across all programming as individuals would not know which code would be selected for review
    - Try to review different code (e.g. dataset and output) to ensure GPP is being applied consistently and ensure that when the code is supplied any background documentation (e.g. dataset specification, output shell, actual output and log files) are also supplied as well
    - Take into account whether projects are part of a series and code has been copied from one to another. Bad habits may be copied throughout although the guidance in place should encourage people to take ownership of programs, including programs they did not create initially
- How will feedback be given
    - Just to the individual – more informal focus with the aim being to support people in taking responsibility for implementing GPP themselves
    - To the individual and individual's manager – more formal, may not engage people in the process as well but could help with serial offenders

GOALS AND OBJECTIVES

The most formal process is to include following GPP within individuals goals and objectives for the year meaning that compliance or non-compliance can affect an individual's overall performance review. If followed correctly this method can achieve greater overall compliance but consider the following:

- This will most likely need to be assessed by code review. In addition to the considerations in the section above, take into account the following:
    - It's important that time is allowed for initial code review and subsequent review to allow the individual to demonstrate they have taken onboard feedback. If code review is only performed once and the individual is non-compliant at that stage this should not feed into an end of year assessment
    - Who does the code review is even more important- they need to be objective with no hidden agendas

- Code should be selected at random
- Consider encouraging people to feedback on code they have assumed responsibility for but which was originally written by someone else (however bear in mind objectivity)

If following this method it's crucial to ensure everyone buys in to the process and managers will have an important role to play.

BENEFITS

What benefits can be expected with the successful implementation of GPP? To highlight this consider the example code below included as the original version and updated version once GPP had been implemented

Original Code

```
data alb_tmp ;
set &derived_target..alb;
if saffl = 'Y' ;
if aval is missing then do;
tempflag1='x';
tempflag2='y';
end;
keep subjid usubjid  avisit avisitn paramcd aval chg armcd;
run;

data alb_tmp( where = (avisitn is not missing));
set alb_tmp;
where (aval is not missing or chg is not missing);
run;

proc sql noprint;
select armcd,count(distinct subjid) into :arm1 - :arm2 , :trt1 -:trt2
from &derived_target..asl
where saffl = 'Y'
group by armcd;

create table comb_alb as
select subjid,avisit,avisitn,"Actual" as groupby,paramcd,armcd,aval as val from
alb_tmp
where aval ne .
union all
select subjid,avisit,avisitn,"Change_baseline" as groupby,
paramcd,armcd,chg as val from alb_tmp
where chg ne . ;
quit;

proc sort
data = comb_alb;
by groupby paramcd armcd avisitn subjid;
run;

proc summary nway data = comb_alb;
class groupby paramcd armcd avisit;
var val;
output out=dean_out(drop = _type_ _freq_) median = Median;
run;
```

This is not GPP compliant for several reasons:

- Lack of comments – this will make it more difficult for a new programmer to pick the code up and assess where modifications are needed
- Lack of indentation makes the code less readable
- Temporary datasets are overwritten which can lead to problems debugging
- Temporary dataset names are not meaningful, also not helping understanding of the program

Updated Code

```
  ************************************************************************;
  *** Get external data                                                  ***;
  ************************************************************************;

DATA alb(KEEP=subjid usubjid avisit avisitn paramcd aval chg armcd);
  SET &derived_target..alb;
  WHERE saffl = 'Y';
  IF aval IS MISSING THEN DO;
    tempflag1='x';
    tempflag2='y';
  END;
RUN;

PROC SQL NOPRINT;
  SELECT armcd,
         COUNT(DISTINCT subjid)
    INTO :arm1 - :arm2 ,
         :trt1 -:trt2
  FROM &derived_target..asl
  WHERE saffl = 'Y'
  GROUP BY armcd;
QUIT;

  ************************************************************************;
  *** Remove missing values                                              ***;
  *** Create standard structure for summarising                          ***;
  ************************************************************************;

DATA alb_nomiss;
  SET alb;
  WHERE (aval IS NOT MISSING or chg IS NOT MISSING) AND (avisitn IS NOT MISSING);
RUN;

PROC SQL NOPRINT;
  CREATE TABLE comb_alb AS
  SELECT subjid,avisit,avisitn,"Actual" AS groupby,paramcd,armcd,aval AS val
  FROM alb_nomiss
  WHERE aval NE .
  UNION all
  SELECT subjid,avisit,avisitn,"Change_baseline" AS groupby,
         paramcd,armcd,chg AS val
  FROM alb_nomiss
  WHERE chg NE .;
QUIT;

  ************************************************************************;
  *** Sort and summarise the data                                        ***;
  ************************************************************************;

PROC SORT DATA = comb_alb
          OUT = sort_Alb;
  BY groupby paramcd armcd avisitn subjid;
RUN;

PROC SUMMARY NWAY DATA = sort_alb;
  CLASS groupby paramcd armcd avisit;
  VAR val;
  OUTPUT OUT=dean_out(DROP=_type_ _freq_) MEDIAN=median;
RUN;
```

The updated code which follows GPP is far easier to read and pick-up this leads to efficiencies in several ways:

- Less time spent understanding the program and making updates
- More portable code – doesn't require an experienced programmer to pick it up

- Flexibility – when code follows GPP it is far easier to assign resource to help with any modifications or re-running

This code only highlights a few aspects of GPP, consider some other benefits listed below

- Using defensive programming from initial program set-up can make programs more robust and lead to fewer updates needed later down the line saving time and resource
- Programs developed using GPP are far more likely to be re-used leading to time and cost savings on future studies

CONCLUSION

The benefits of implementing a good programming practice guideline speak for themselves. Developing programs which follow consistent rules will make these more portable and reduce the excess time that people can spend trying to understand and debug code written by someone else. In order to engage teams in adopting GPP a framework needs to be in place which encourages and provides support for implementation, including a GPP guideline. In the absence of any company guidance, the guideline developed by the PhUSE GPP board highlights the main standards to be followed and we would encourage companies to include these in their own guidance to help achieve an industry standard.

REFERENCES

- PhUSE Good Programming Practice Guideline: http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice_Guidance

ACKNOWLEDGMENTS

Thanks go to the PhUSE GPP Board for input and review.

RECOMMENDED READING

- PhUSE Good Programming Practice Guideline: http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice_Guidance

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Author Name	Kate Peacock
Company	Quintiles
Address	500 Brook Drive, Green Park
City / Postcode	Reading, RG2 6UU
Work Phone:	+44 (118) 4508502
Email:	Kate.Peacock@quintiles.com
Web:	