Reproducibility: Reports and Informative Outputs

Code should be managed in Git (see separate document) to allow someone to go back to the point at which it was run, but this doesn't always show what happened. To aid transparency and reproducibiity, a log, or report, that captures input data, options used, the code pathway, and outputs can be useful. A report can provide quality assurance, sense checks and an understanding of how data was processed.

We will look at creating PDF reports, but first there are some simple tips for getting informative output.

Simple Tips for Informative Output

Date and time when the code was run.

```
% Near the start of your processing, get the date and time
dtnow    = datetime ;
time_str = string(dtnow) ;
disp("Processing started at: " + time_str)
```

```
Processing started at: 07-Dec-2023 17:52:43
```

Name of a file used as input.

```
% Assume name of data file is in the variable: dfilename
% Often the full path is very long so separate file and path to aid
readability

dfilename = tempname ;  % use temporary name for this example

[pn, fn, ext] = fileparts(dfilename) ; % split into path, filename and
extension
if ~isempty(ext)
    fn_wth_ext = [fn, '.', ext] ;
else
    fn_with_ext = fn ;
end

disp("Data filename: " + fn_with_ext)
```

```
Data filename: tp3075f6c9_ac6f_4c70_92ce_ef86b454f534
```

```
disp(" Data path: " + pn)
```

```
 Data path: /private/var/folders/cq/0w0rbw6j62l2r589_bgbd0l00000gn/T
```

Optional parameters values (user-supplied and default values)

```
% Name, Value optional parameters and their defaults can be output
% easily in a Table

% This calls the function string_modify at the end of this LiveScript
```

```
% with one parameter set and the other using its default value.
%
% Within the function, a summary table is output
outstr = string_modify("middle",leading="First_")
```

```
Values of optional parameters used
optsTable = 1×2 table
```

	leading	trailing
1	"First_"	"_post"

```
outstr =
"First_middle_post"
```

Use of the Report Generator

The ReportGenerator allows for creation of PDF reports.

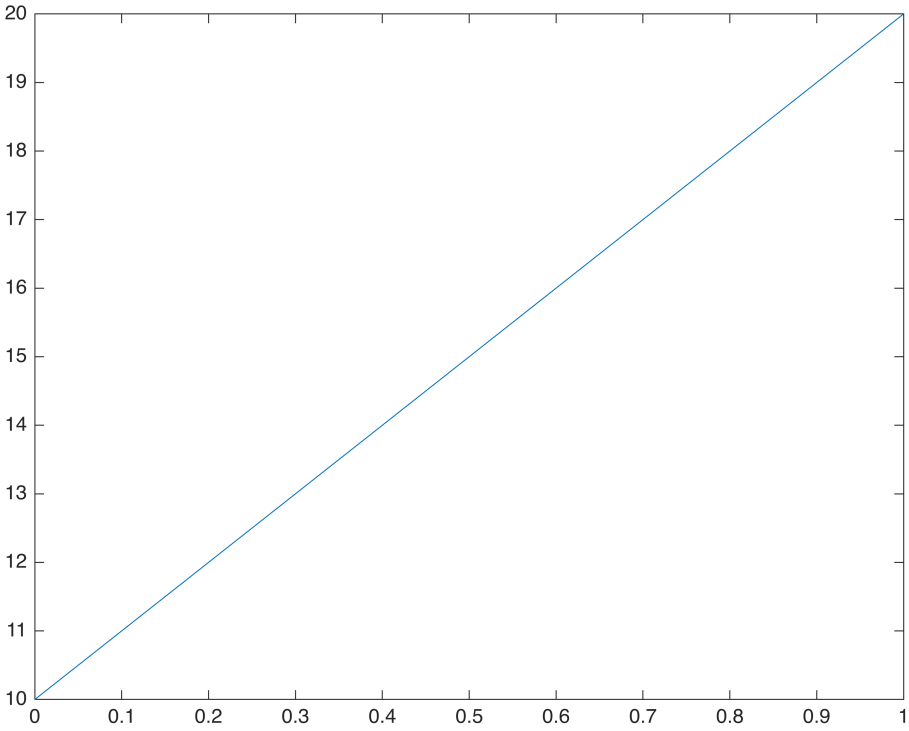
First set a name for the results folder based on the date and time, and create that folder.

```
% To prevent overwriting, create a folder using the date and time
% but avoid spaces in the name
dtnow = datetime ;
dtnow.Format='yyyy-MM-dd''T''HHmmss' ; % (no weird characters or spaces)
time_name_char = char(dtnow);

resultsFolder = fullfile(tempdir, ['res-',time_name_char]) ;

[status,msg,msgID] = mkdir(resultsFolder) ;  % make the folder (directory),
check for errors
if status == 0
    warning(msgID, msg)
end
```

Now open a report file, here called `rptresults.pdf`. From there add text, tables and figures as desired, before closing. The finished report can be viewed using `rptview`.

```
rptFileName = 'rptresults.pdf' ;
rptFFN = fullfile(resultsFolder, rptFileName) ;

import mlreportgen.report.*
import mlreportgen.dom.*

rpt = Report(rptFFN,'pdf');
rpt.Layout.Landscape = true;

% Add text
append(rpt, Paragraph(['This report was generated: ', time_name_char]) )
append(rpt, Paragraph(' '))

% Add a Table
opts.Age = 38;
opts.Height = 70 ;
```

```
pTObj = MATLABTable(struct2table(opts, 'AsArray',true)) ;
pTObj.Style = [pTObj.Style { FontSize('8pt')} ] ;
append(rpt, pTObj)

% Add a figure
hf = figure ;
plot([0 1],[10 20])
```




```
figrpt = Figure(hf);
figrpt.Snapshot.Caption = 'Output data figure' ;
figrpt.Snapshot.ScaleToFit = true ;
append(rpt,figrpt);

% close Report
close(rpt)

% Open Report View to see PDF that has been created
rptview(rpt)
```

GIT source control information

Here is some code that can be used to record information about the Git version of the code when it was run.

```
% Function gitrepo errors if there is no Git repository so use try-catch
try
    repo = gitrepo;
catch
    disp("Git repository not found")
    return
end

disp("Branch: " + repo.CurrentBranch.Name)
```

```
Branch: main
```

```
disp("Last Commit ID: " + repo.LastCommit.ID)
```

```
Last Commit ID: 3f65cf063a38ded6a8ed6b9570ec997099a1b12a
```

```
disp("Last Commit: " + string(repo.LastCommit.CommitterDate))
```

```
Last Commit: 07-Dec-2023 17:48:18 +0000
```

```
if ~isempty(repo.ModifiedFiles)
    disp("There are modified files")
end
```

```
There are modified files
```

```
if ~isempty(repo.UntrackedFiles)
    disp("There are untracked files")
end
```

```
There are untracked files
```

```

```

Function used earlier to demonstrate using Name,Value pairs as optional inputs

```
function outstr = string_modify(str, opts)
%
% outstr = string_modify(str)
% outstr = string_modify(str, Name, Value ...)
%
% Example
% outstr = string_modify("centre", trailing="_end part")

arguments
    str  {mustBeText}
    opts.leading  = "pre"
    opts.trailing = "_post"
end

outstr = opts.leading + str + opts.trailing ;
```

```
disp("Values of optional parameters used")
optsTable = struct2table(opts,'AsArray',true)

end
```