Make and gnuplot

Objective: You want to create an automated build, test and reporting system for a multi-threaded application. Consider there is a setup that contains:

(1) An application source which is compiled to build the application,	(App)
(2) The configuration scripts to generate output (logs) using the built application,	(runtest)
(3) An analysis phase to analyse the log and produce processed input for plotting graphs,	(analyse)
(4) A plotting phase to plot the graphs, and	(plot)
(5) A report written in latex which presents the plots as an executive summary.	(report)

This has to be done using GNU Make and gnuplot utilities performing **minimal** required operations when any of the relevant files change.

Setup:

C source files **prog.c thread_function.c common.h** which are used to build the application (say App).
The application takes two inputs:
(1) number of elements is specified in param.txt and,
(2) number of threads in threads.txt
This application is similar to the class example for pthreads where time taken is printed as the output of the program.
Configuration files: **threads.txt**, **params.txt**
threads.txt ==> Specifies number of threads the built program should be executed with
params.txt ==> num_of_elements: Specifies number of elements to be passed as
parameter

Steps:

1. Building the application from source
2. Execute tests and generate reports as follows:
   Your statistics collections should perform the following steps (you can write this using bash or python etc.)

```
for each <num_elements> in params.txt
    for each <num_threads> in threads.txt
        execute App 100 times with <num_elements> <num_threads>
        and collect results (by redirecting the op)
```

3. Analyse the generated logs and plot the results
   You are required to analyse the results in order to produce the processed data to plot the following using gnuplot (using scripts for each),
   (a) One point (scatter) graph for each thread configuration where X-axis is <num-of-elements> and Y-axis corresponds to execution time for each sample.
   (b) A **single** line graph for each thread configuration where X-axis is <num-of-elements> and Y-axis is average execution time over 100 samples, (*the line for each thread should be plotted in a single plot with proper legends for each thread*)
   (c) One bar graph with X-axis as <number of elements> and Y-axis as average speedup (as you have executed the same configuration for 100 times) w.r.t. one thread execution. Plot the bars representing <num of threads> for every point in X-axis (see the app_speedup.eps plot from the class examples).
   (d) Plot the same graph as (c) but with error bars. Where error bars represent the variance calculated over 100 samples for a particular configuration.
4. A latex document that embeds the above plots to produce a pdf report. Note that each of this plot should be embedded as a separate figure with labels and some explanation of results (MAX 3 sentences).

Notes:

1) Assume that threads.txt is fixed (*It will not be changed during evaluation*).
2) You are **not allowed** to change the C program
3) Your Makefile should support the following,
   (a) Build the latex report, which is eventually required to perform all the steps (if required)
   (b) Build each step separately: **$make App**, **$make runtest**, **$make analyse**, **$make plot** and **$make report**.
   (c) Incremental build. Example: if .tex file is modified only reporting step should be performed, if one single gnuplot file changes, only that plot should be recreated.
   (d) Clean the generated files