<u>Important notes to the tissue microarray analysis tool 'TMAinspiration'</u>

Index

Installation of the programs

The operating system for all subsequent descriptions is assumed to be Linux.
The binaries are compiled on Debian/Ubuntu 12.04.2 with the flag 'static', so should include (nearly) all dependencies. This version should also run on newer operating system versions and related systems (tested up to Ubuntu 14.04.3). For the MPI version of the program (core/system number >1) a functional MPI environment like 'mpich' is necessary (tested up to 3.1.4).

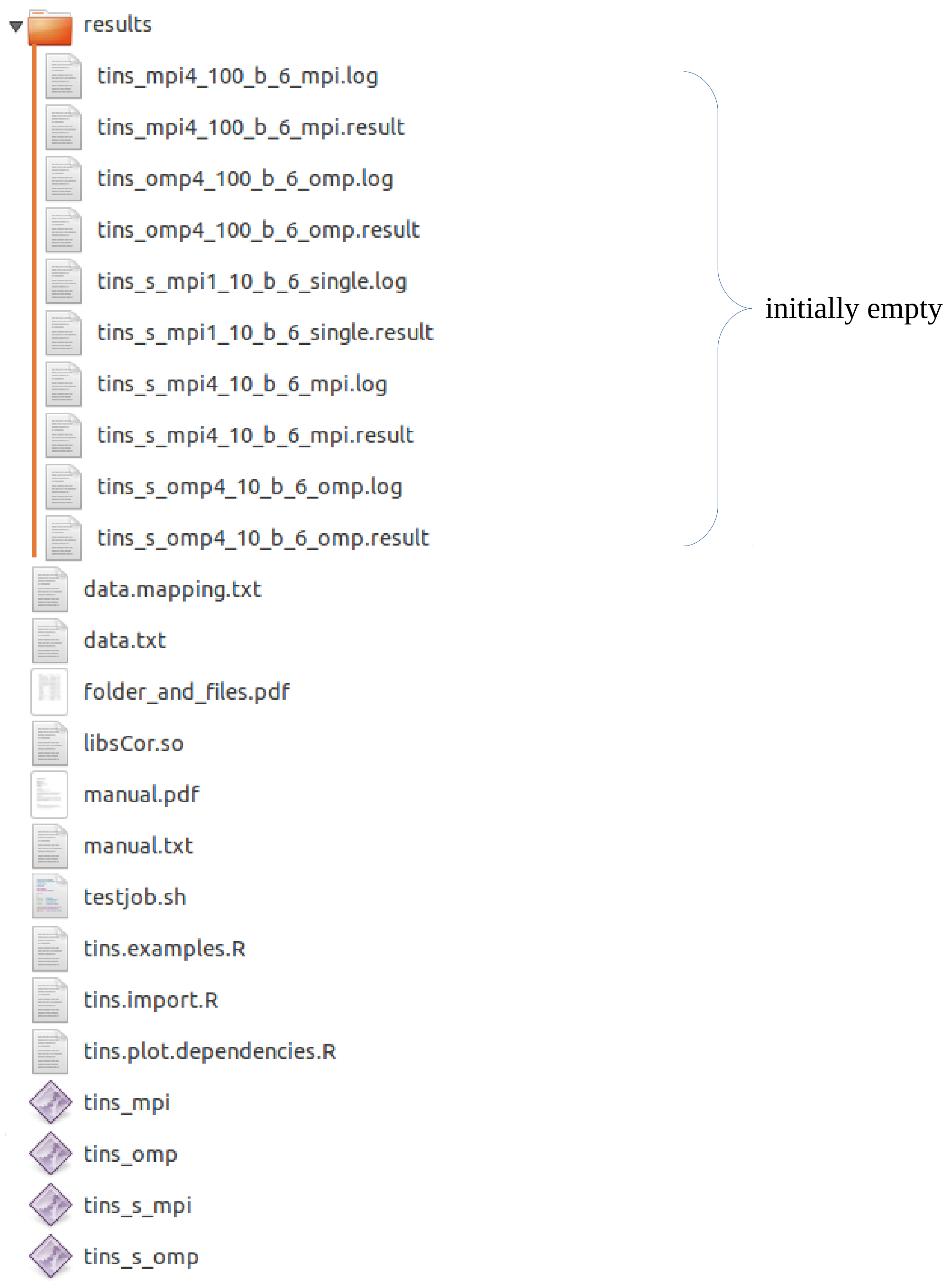
You should download all the following files and place it in a dedicated folder of your choice in your home folder:

TMAinspiration readme.pdf :	description of formats and usage (this file)
testjob.sh :	master file - collection of command lines for the test data
data.txt :	test data
data.mapping.txt :	scheme denoting which of the columns from test data will be used
tins_s_mpi :	calculate the best order of a certain reference to test selection and verify the result by sampling; using the Message Passing Interface (MPI)
tins_s_omp :	the same - but using Open Multi-Processing (OMP)
tins_mpi :	the same as **tins_s_xxx** but calculates the best order for all combinations of a certain group size (using MPI)
tins_omp :	the same - but using OMP

Additionally to that we provide some R source files to import and handle the result files (see section '**R script files**').

Running the test data

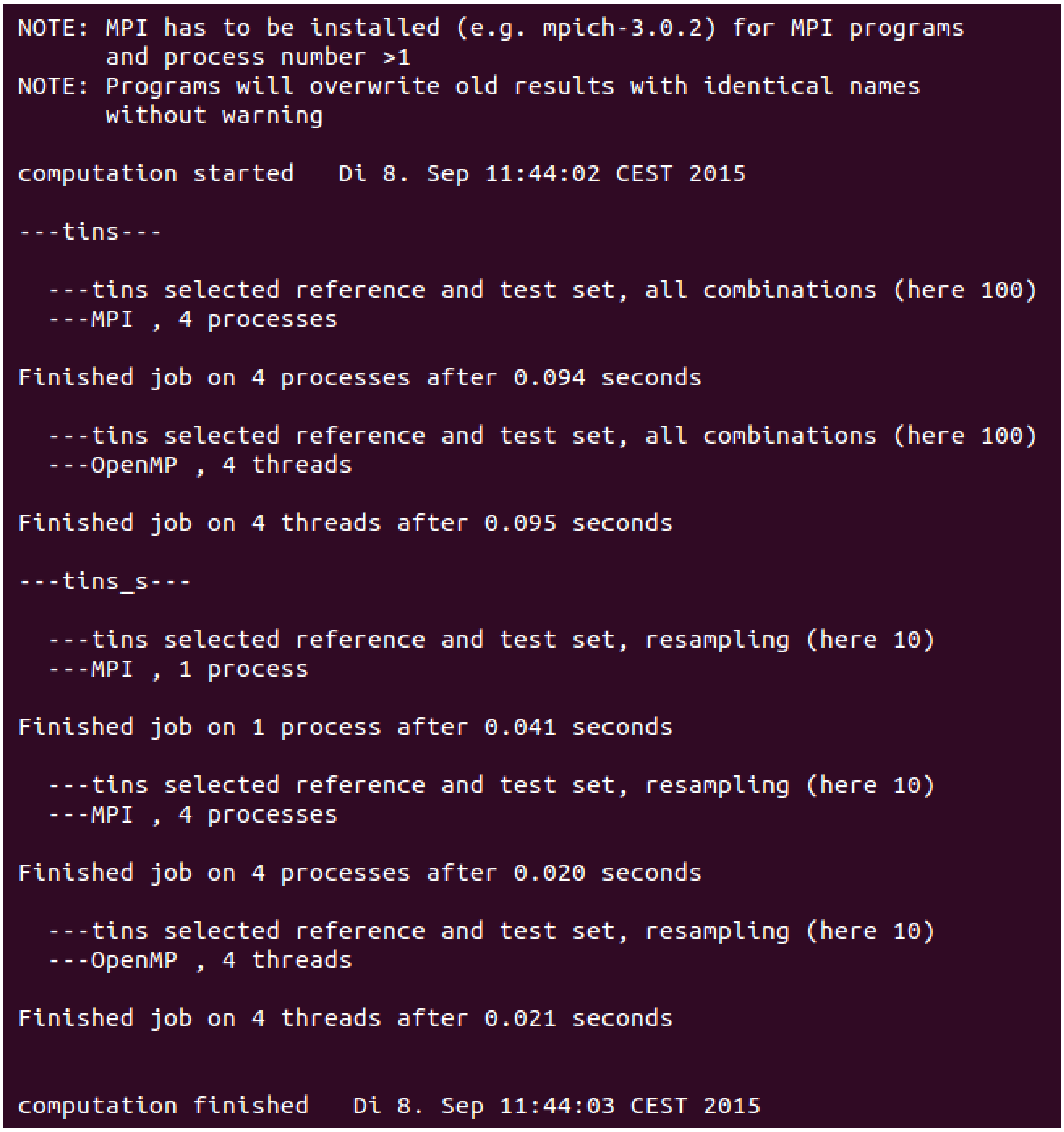
Create a folder and download all files to that folder.
If not already done open a terminal.
Switch with 'cd' into this folder.
Create here a subfolder called 'results'.



```
results
    tins_mpi4_100_b_6_mpi.log
    tins_mpi4_100_b_6_mpi.result
    tins_omp4_100_b_6_omp.log
    tins_omp4_100_b_6_omp.result
    tins_s_mpi1_10_b_6_single.log
    tins_s_mpi1_10_b_6_single.result
    tins_s_mpi4_10_b_6_mpi.log
    tins_s_mpi4_10_b_6_mpi.result
    tins_s_omp4_10_b_6_omp.log
    tins_s_omp4_10_b_6_omp.result
data.mapping.txt
data.txt
folder_and_files.pdf
libsCor.so
manual.pdf
manual.txt
testjob.sh
tins.examples.R
tins.import.R
tins.plot.dependencies.R
tins_mpi
tins_omp
tins_s_mpi
tins_s_omp
```

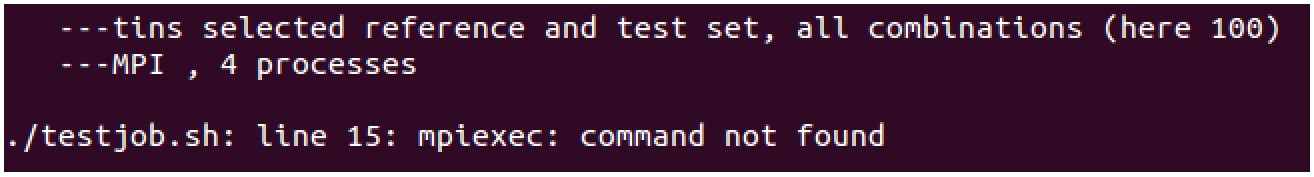
Figure 1 -A typical folder structure intended for the test sample.

Run the master file 'testjob.sh'
  (errors: check if the script and the binaries are executable: terminal: ls -al, flag: x).
You see some 'echo' lines of responses in your terminal and
after about 1 second (obviously depending on your processor power)
all jobs should have been finished without additional error messages
  (we assume here a standard 4 core PC).



```
NOTE: MPI has to be installed (e.g. mpich-3.0.2) for MPI programs
      and process number >1
NOTE: Programs will overwrite old results with identical names
      without warning

computation started   Di 8. Sep 11:44:02 CEST 2015

---tins---

  ---tins selected reference and test set, all combinations (here 100)
  ---MPI , 4 processes

Finished job on 4 processes after 0.094 seconds

  ---tins selected reference and test set, all combinations (here 100)
  ---OpenMP , 4 threads

Finished job on 4 threads after 0.095 seconds

---tins_s---

  ---tins selected reference and test set, resampling (here 10)
  ---MPI , 1 process

Finished job on 1 process after 0.041 seconds

  ---tins selected reference and test set, resampling (here 10)
  ---MPI , 4 processes

Finished job on 4 processes after 0.020 seconds

  ---tins selected reference and test set, resampling (here 10)
  ---OpenMP , 4 threads

Finished job on 4 threads after 0.021 seconds


computation finished   Di 8. Sep 11:44:03 CEST 2015
```

Figure 2 - A successful run of the provided sample.



```
  ---tins selected reference and test set, all combinations (here 100)
  ---MPI , 4 processes

./testjob.sh: line 15: mpiexec: command not found
```

Figure 3 - A run where the multi-core MPI parts are not working due to a missing MPI installation.
Install MPICH from your software repositories or download MPICH and install it.
Note: MPI is not necessary for the OpenMP binaries.

If all went well the results folder is now filled with several pairs of result files, generated from the test data.

The assignment of the result files to the binaries can be figured out by
comparing the result names with the command lines in the bash script 'testjob.sh'.

The 'log' files will report some important runtime information and sometimes warnings.

The result files have a 'tab' separated data format which is described below.
They can easily be imported in e.g. a 'R' environment for further processing
and visualization (see section: '**R script files'**).

So the bash file 'testjob.sh' can be seen as a template for your own job specific command lines.

Input formats

The programs expect 'tab' separated *data* in a plain text file and
a single 'tab' separated line in the case of the *mapping* file.

The *data* file host as many protein marker columns as necessary or even more.
The *mapping* file defines which columns will be used and
the options in programs:
tins_s_xxx : 7. - number of reference protein markers
tins_xxx : 6. - number of reference protein markers
will define how many protein markers in the <u>beginning</u> of the *mapping* file belong to the reference partition.
Every number in the *mapping* file is a pointer to the column position in the *data* file.

<u>Data file format</u>

ID	proteinmarker1	proteinmarker2	...	proteinmarker20
abc1	2	3	...	2
...	...	...	...	...
abc85	1	3	...	2

Header on top: immunohistochemically measured protein markers
description column: samples/cases
See also section 'Mapping file'.

So the first column is the sample description column. The naming scheme
should be simple (standard characters and numbers) and no blanks.
This holds true also for the header row.
The table structure is filled by positive integers or numbers (excluding zero)
with up to 6 decimals. So scale the data appropriately before you start.
Missing values are not allowed. So try to find a reasonable strategy
to fill these gaps (but check if this is appropriate for your data;
this is especially of importance if you own data sets with less than 200 rows).

<u>Mapping file format</u>

Y	Y	4	5	6	...	20

Contains only one row. First and second position: a character denoting
if a header row respectively a first identifier column is present [Y,y]
or not [N,n].

We **strongly recommend** to use a header row and identifier column (despite it will work without),
because otherwise the log file will not contain the initial order, data management becomes

therefore complicated, and the provided R scripts will not work.

The following positions denote the position of the columns in the data file
which should act as *reference* partition markers (the number of reference markers is given as a parameter to the program).
The length of this block is defined in the program options (see page 7, 8).

The next following block denote the position of the columns in the data file
which should act as *test* partition markers.

Mapping file example:

cf. file: 'data.mapping.txt'

We have *6 reference* and *10 test protein markers* out of 20 markers in total.

So position 1 and 2 : Y , Y

position 3 to 8: column position numbers of the selected reference protein markers

position 9 to 19: column position numbers of the selected test protein markers

So 4 columns were not selected from the raw data.

Programs -- tins_s_mpi & tins_s_omp --

Purpose :
Find a dependency structure in between the protein markers of interest, which is optimal concerning the given data.

Usage :
Select one reference and one test set from the data columns and calculate the best
test set order. Verify the result by a sampling procedure.

tins_s_mpi : uses the MPI framework (e.g. 'mpich') while
tins_s_omp : uses the OpenMP framework (Intel, linked into the executable).

Description of the command line arguments

```
1.         - path and name of the data input file
2.         - path and name of the mapping input file
3.         - path and name of the result file
4.         - path and name of the log information output file

5.         - number of resamplings (e.g. 1000 or 10000,  1: no resampling)

6.         - resampling method
                     's' or 'S' for shuffling (*)
                     'b' or 'B' for bootstrapping (+)
                     default: shuffling

7.         - number of reference protein markers
                     default = 6

8. omp only  - number of threads OpenMP should use for parallel computation
               default = number of all available processors (pipelines)


(*): shuffling means randomly rearrange the complete (used) data matrix
(+): bootstrapping is performed per protein marker column
```

Note: `Arguments cannot be left out - order matters.`

Programs -- tins_mpi & tins_omp --

Purpose :
Create an overview on all combinations of a certain group size.

Usage :
Select one reference set and one test set as starting point and calculate the best test set order for each possible combination of that configuration.

Example: If we pick 16 protein markers and partition in 6 reference and 10 test protein markers, we can calculate all 6 of 16 group combinations and their best results.
So we look in the complete combinatorial space and analyse the distribution of solutions. This defines the position of the selected combination in relation to all combinations.

tins_mpi : uses the MPI framework (e.g. 'mpich') while
tins_omp : uses the OpenMP framework (Intel, linked into the executable).

Description of the command line arguments

```
1.        - path and name of the data input file
2.        - path and name of the mapping input file
3.        - path and name of the result file
4.        - path and name of log information output file

5.        - number of combinations (if the number is not specified
                    or higher than the maximum number of possible
                    combinations within the reference set, all possible
                    combinations are calculated)
                    [R language: number of combinations: e.g. choose(n=16, k=6)]

6.        - number of reference protein markers
                    default = 6

7.        - number of protein markers differing from the original combination
                    default = number of reference protein markers
                    so also sub-partions can be calculated

8. omp only  - number of threads OpenMP should use for parallel computation
               default = number of all available processors (pipelines)
```

Note: Arguments cannot be left out - order matters.

Output formats

The programs returns a 'tab' separated plain text file holding
a table of data.
Additionally a *log* file will be returned.

<u>Result file for -- tins_s_mpi & tins_s_omp --</u>

The order by starting the calculations can be found in the *log* file.
In that order the first n protein markers will be the n reference protein markers.
The following markers will be the m test markers.
So this line has length of n+m.

Based on the given order in the *log* file the interpretation of the result file can be done:
First position: sum of sum of squares error (ssqg for the optimal order).
Following that the m test markers: denoting the optimal order of test markers.

The first row is the optimal order of the given selection while all
the following lines are optimal orders from sampling tests.
A good selection is characterized by a ssqg which is far on the lower
end of the sampling distribution of ssqg.

<u>Result file for -- tins_mpi & tins_omp --</u>

The order by starting the calculations can be found in the *log* file.
In that order the first n markers will be the n reference markers.
The following markers will be the m test markers.
So this line has length of n+m.

Based on the given order in the *log* file the interpretation of the
result file can be done:
First position: sum of sum of squares error (ssqg for the optimal order).
Following that the selection of n reference markers.
Following that the selection of m test markers.
Following that the optimal order of the m test markers.

The first row denote the optimal order of the selection given to the program.
All the following lines are other combinations (all possible or a subset thereof).
The unique combinations will be constructed from the given selection and their
given group size (of reference and test group).

R script files

We provide three R script files (The R Project for Statistical Computing *https://cran.r-project.org/*) to facilitate the import and visualization of the data in R.
Of course this code needs some basic R skills, but not much.

Download the following files and make them part of your R workspace
(e.g. execute the command : 'source(/path/file_name)').

tins.import.R :	function(s) to import the result data and create a symbolic header line
tins.plot.dependencies.R :	function(s) to plot the results and some ssqg distributions

and

tins.examples.R	example code utilizing the provided R functions

The example code is based on the test data results and describes in an exemplary way how to make use of the generated results.

Additional usage information and code comments are given in the R scripts.
This code might also serve as a starting point for your own analysis scripts.

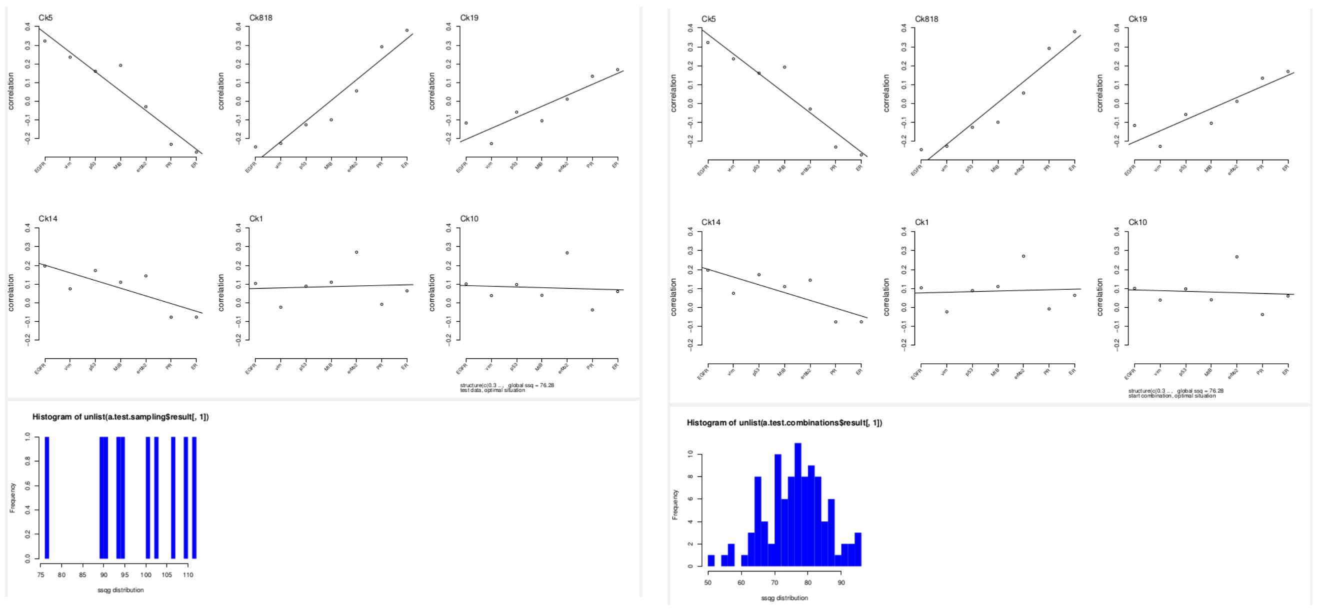


Figure 4 - Resulting R sample graphs

Troubleshooting

All our tests were performed on Debian and Ubuntu Linux systems.

If there are errors during the start of the binaries :

- check the arguments in the program call
- check if all folders exist
- use absolute path names
- check if the data and mapping file format is valid
- check if the number/character format in the data and mapping file is valid (OS dependencies!)

- inspect the data in LibreOffice and save them again under a different name as a 'tab' separated format

- use the small test data set and see if this example is running
- check if -our- default example is working with -your choice- of parameters (check if the number of columns is less or equal that of the given test sample)

- check if the corresponding OpenMP / MPI program is working (first choice is OpenMP)
- reread the complete documentation
- ask Linux experts in your group

If you assume that system dependencies are not fulfilled you might run
'ldd program_name' while program_name corresponds to one of our binaries.
ldd might provide hints on missing system libraries which should be installed first - despite this might be a very uncommon and strange situation.

If you are at the end, write all your tests and the respective results up
in a structured and logical form, restrict yourself to the important parts,
and send us an email containing this report and your question.

The anonymized discussion will be made public in a QA section on the web site for other users.