Compiling pcaPP for Matlab

Heinrich Fritz

September 12, 2011

1 Introduction

The main functions of the **R**-package pcaPP are implemented in an environmentindependent manner, which allows the user to use this package beyond the scope of **R**. The package has also been prepared to be compiled and used with **Matlab**, which is summarized and demonstrated in this document. The following items are required for using pcaPP together with **Matlab**:

- The pcaPP package sources `pcaPP_1.9-44.tar.gz` (available at `http://CRAN.R-project.org/package=pcaPP`).
- **Matlab** (version $\geq$ 2010a).
- A compatible **C++** compiler (for currently supported compilers see `http://www.mathworks.com/support/compilers/current_release/`).

Section 2 helps to set up a suitable compiler together with **Matlab**, whereas Section 3 gives instructions on how to actually compile the package. Section 4 demonstrates some examples on the usage of the package and Section 5 concludes.

2 Setting up the Compiler

Assuming that **Matlab** has already been set up properly on the target system, the first step is to set up a suitable **C++** compiler, such that **Matlab** recognizes it. A list of compatible compilers can be obtained by typing

```
>> mex -setup
   n
```

into the **Matlab** console. Once a compiler from this list has been installed on the system, select it (by using the previous command) and make sure that **Matlab** locates it correctly. Note that after installing a compiler **Matlab** might have to be restarted for correctly recognizing it. Finally assure that the compiler has been set up properly by typing

```
>> mex.getCompilerConfigurations ('C++')
```

Matlab should now correctly display the chosen compiler's details. A more extensive introduction to the mex-interface and its configuration can be found at `http://www.mathworks.de/support/tech-notes/1600/1605.html`.

3 Compiling pcaPP

Extract the downloaded package sources (`pcaPP_1.9-44.tar.gz`) to a working directory, (e.g. *C:/work*), and set **Matlab**'s current directory to the *pcaPP/matlab* subfolder:

```
>> cd ('C:/work/pcaPP/matlab')
```

Now the package is ready to be compiled by calling pcaPP's `setup` routine:

```
>> setup
Changing the current directory to '../src' ... ok
Compiling the pcaPP package ... ok
Copying the 'pcaPP.mex*' file(s) to '../matlab' ... ok
Changing the current directory back to '../matlab' ... ok

  Successfully compiled the pcaPP package for Matlab!
```

Note that this **Matlab**-setup routine has been tested with Microsoft's Visual C++ 6.0 compiler. Other compilers supported by **Matlab** are very likely to work as well, but have not been tested in this context yet.

4 Using pcaPP

Once the preceding code has been executed successfully, the pcaPP package can be used almost the same way as in **R**. The following functions are available in **Matlab**: `l1median_HoCr`, `l1median_VaZh`, `PCAgrid`, `PCAproj`, `qn`, `sPCAgrid` and work as described in the **R** man pages:

```
>> rand('seed', 0) ;
>> X = rand (100, 5) ;
>> mHC = l1median_HoCr (X)

mHC =

    0.5261    0.5123    0.5171    0.4963    0.4635

>> mVZ = l1median_VaZh (X)

mVZ =

    0.5261    0.5123    0.5171    0.4963    0.4635

>> pc = PCAgrid (X)

pc =

        sdev: [0.4251 0.3939]
    loadings: [5x2 double]
           k: 2
         obj: [0.1807 0.1552]
       n_obs: 100
       scale: [1 1 1 1 1]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
    pc_order: [1 2]
      scores: [100x2 double]

>> sp = PCAproj (X, 2)

sp =
```

```
    loadings: [5x2 double]
        sdev: [0.4027 0.3835]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
       scale: [1 1 1 1 1]
       n_obs: 100
>> rand('seed', 0) ;
>> X = rand (100, 5) ;
>> mHC = l1median_HoCr (X)

mHC =

    0.5261    0.5123    0.5171    0.4963    0.4635

>> mVZ = l1median_VaZh (X)

mVZ =

    0.5261    0.5123    0.5171    0.4963    0.4635

>> pc = PCAgrid (X)

pc =

        sdev: [0.4251 0.3939]
    loadings: [5x2 double]
           k: 2
         obj: [0.1807 0.1552]
       n_obs: 100
       scale: [1 1 1 1 1]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
    pc_order: [1 2]
      scores: [100x2 double]

>> sp = PCAproj (X, 2)

sp =
```

```
    loadings: [5x2 double]
        sdev: [0.4027 0.3835]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
       scale: [1 1 1 1 1]
       n_obs: 100
      scores: [100x2 double]

>> sp = PCAproj (X, 5, 'mad', 'lincomb')

sp =

    loadings: [5x5 double]
        sdev: [2.0793 0.4027 0.3835 0.3474 0.3110]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
       scale: [1 1 1 1 1]
       n_obs: 100
      scores: [100x5 double]

>> sc = qn (X)

sc =

    0.2958     scores: [100x2 double]

>> sp = PCAproj (X, 5, 'mad', 'lincomb')

sp =

    loadings: [5x5 double]
        sdev: [2.0793 0.4027 0.3835 0.3474 0.3110]
      center: [0.5261 0.5123 0.5171 0.4963 0.4635]
       scale: [1 1 1 1 1]
       n_obs: 100
      scores: [100x5 double]

>> sc = qn (X)

sc =
```

```
0.2958
```

5 Conclusions

The configuration of a **C++** compiler in the context of **Matlab** has been discussed briefly, as well as how to compile the **R** package pcaPP in this environment. Further some examples on how to use the package in **Matlab** were given. Due to the package's architecture the same **C++** sources can be used in both environments, which increases the availability of this software beyond the scope of the **R** community.