NHH

INSTITUTT FOR SAMFUNNSØKONOMI

DEPARTMENT OF ECONOMICS

SAM 25 2010

ISSN: 0804-6824

OCTOBER 2010

Discussion paper

Exploiting Parallelization in Spatial Statistics: an Applied Survey using R

BY

ROGER BIVAND

Norges Handelshøyskole

NORWEGIAN SCHOOL OF ECONOMICS AND BUSINESS ADMINISTRATION

Exploiting Parallelization in Spatial Statistics: an Applied Survey using R

Roger Bivand
Norwegian School of Economics and Business Administration

Abstract

Computing tasks may be parallelized top-down by splitting into per-node chunks when the tasks permit this kind of division, and particularly when there is little or no need for communication between the nodes. Another approach is to parallelize bottom-up, by the substitution of multi-threaded low-level functions for single-threaded ones in otherwise unchanged user-level functions. This survey examines the timings of typical spatial data analysis tasks across a range of data sizes and hardware under different combinations of these two approaches. Conclusions are drawn concerning choices of alternatives for parallelization, and attention is drawn to factors conditioning those choices.

Keywords: high performance computing, parallel computing, computer cluster, multi-core systems, spatial statistics, benchmark, R.

1. Introduction

Innovations in methods of statistical data analysis can be presented and tested using open source collaborative tools. Rey (2009, p. 197) has highlighted the important position of R contributed packages in spatial data analysis. When innovations in methods occur, the developers often only explore the performance of their software on one platform, and are unable to offer advice on how the implementations may perform on other platforms. While it would be tempting to conduct this survey across commonly available high performance computing platforms for a larger subset of techniques discussed in Bivand *et al.* (2008), attention will be restricted to functions recently introduced into the **spdep** and **spgwr** contributed packages.

Calling the procedures undertaken in this survey "high performance computing" is slightly impudent, as we will only be concerned with attempts to improve performance on standard hardware; the comparisons should, however, scale up on specialised hardware provided that similar conditions are matched. In general, more efficient and careful coding, building on the output of profiling in R, is capable of providing substantial enhancements to performance, as described in R Development Core Team (2010c, pp. 53–55). Once these avenues have been exhausted, it is possible to look to a better exploitation of the available resources on the platforms at hand. In this survey, we will consider two approaches, and what happens when they are applied separately and jointly.

Using a range of hardware under different operating systems, running the same version of R, we will compare run times for a selection of tasks encountered in spatial data analysis for the chosen approaches. The timings were taken once for each combination of conditions on

otherwise lightly loaded machines. The results are used as a basis for recommending which choices may be appropriate under given conditions, for both developers and users.

2. Using R for spatial data analysis

Methods innovations are now often published in geography and GIS journals as formulae, with no (pseudo) code implementation or example data set with results. Only rarely do referees ask to see data or code, usually basing their recommendations on the "manuscript" alone. This leads to substantial barriers to reproducible research, as, not infrequently, no reference implementation is available. In the examples used below, collaboration with the researchers developing new methods of analysis has been crucial to establishing that the R implementations are reasonable representations of those methods.

The first key facility for reproducible research provided by the R project is the full availability of source code, and of details of changes in the code over time. The second facility is full journalling and scripting, permitting all operations to be reconstructed (including RNG seeds). Thirdly, R provides through `Sweave` (Leisch and Rossini 2003) for the writing of code and text in the same document, arguably helping to keep results and text synchronised. Finally, because the R project is a successful collaborative forum, with over 2500 contributed packages on CRAN and high-traffic mailing lists, there are plenty of critical voices and alternatives — for insight into the R communities, see Fox (2009).

Spatial data anaysis using R now covers the needs of students, researchers and practitioners across a range of disciplines; for an overview, see the R contributed packages referenced and used in Bivand *et al.* (2008).[1] The volume of spatial data is highlighted by some as a major challenge, and requires care in data representation and management. However, as the **raster** package shows effectively (Hijmans and van Etten 2010), one can achieve much by forethough and careful coding, here using tiling of image data to permit large objects to be processed chunk by chunk. In fact, challenges come just as much from innovative computational methods, the focus of attention here. Rossini *et al.* (2007) use several spatial data analysis examples in their presentation of the benefits of parallel computing, so going spatial is not just a choice geographers might make.

3. Using multiple cores: principles

Since most contemporary computing hardware is furnished with one or more multi-core CPU, it is tempting to try to use more than one computing thread at a time to get work done faster. Scheduling the allocation of processes and threads to cores is hard, and modern operating systems typically try to balance an appearance of high responsiveness with throughput. Very often at least one core will be at least partly occupied with event loops tracking possible input activity, both from input hardware and from network origins. It is really only on dedicated clusters that the numerical computation process will be given the full attention of the node. Elsewhere, competition for resources will take place, and is unavoidable, and probably even undesirable. Even so, much better use can still be made of the computing resources of standard hardware than is typically the case at present.

[1] See also the Spatial task view: `http://cran.r-project.org/view=Spatial`.

The R engine itself is single-threaded and vectorised (R Development Core Team 2010a), for a recent review, see Tierney (2009). Luke Tierney has been central in advancing both of the two approaches to be considered in this survey. For want of better terminology, we will describe the approaches as bottom-up and top-down. In bottom-up parallelization, the changes are made within the functions that the developer would use, so that operations utilize available resources without the need for a user to start and stop a cluster. Top-down parallelization on the other hand requires the availability of a cluster, among the nodes of which chunks of work are distributed.

The distinction between the bottom-up and top-down approaches is different from the established definitions of fine-grained, coarse-grained, and embarrassingly parallel applications, which range from high-frequency communication between tasks on compute nodes, through low-frequency communication to no communication at all between tasks on nodes apart from initialization from the administrator node to workers, and from workers to adminstrator on completion. In spatial data analysis, it is often the case that computing tasks are embarrassingly parallelizable.

In Tierney (2009, pp. 220–222), the question of how to parallelize R functions is raised — this corresponds to our bottom-up approach. Some may be attacked through the use of a parallelized BLAS library that will be used on multiple threads instead of the default single-threaded linear algebra subroutines provided on most platforms. The use of alternative enhanced BLAS is described in detail in R Development Core Team (2010b, pp. 33–36). Only R binaries for OSX are built by default for the `vecLib` enhanced BLAS, which is threaded; for details see point 12.5 of the R for Mac OSX FAQ.[2] In addition to enhanced BLAS, work is in progress on parallelizing the underlying R mathematical functions library in the **pnmath** package, as described by Schmidberger *et al.* (2009, p. 10).

The top-down approach is represented here by the **snow** package, discussed by Rossini *et al.* (2007) and Tierney *et al.* (2009), and provides simple mechanisms for parallel computing in R. A recent comparison by Schmidberger *et al.* (2009) indicates that the socket communication mechanism in **snow** is efficient, even compared to PVM and MPI; it works without extra software under Windows. Using additional packages depending on **snow**, it is possible to offer an apparently simpler interface and some fault tolerance (Knaus *et al.* 2009; Schmidberger *et al.* 2009). For our purposes, however, **snow** using sockets provides the most general mechanism for administering a cluster, for splitting tasks between nodes, and for executing the split task across available nodes. It is crucial in simulation tasks to ensure that the worker nodes do not use closely correlated streams of random numbers. Using **snow**, this may be secured by the use of the **rlecuyer** package interfacing the random number generator streams described in L'Ecuyer *et al.* (2002).

As we have seen, if work can be chunked into independent parts, then it can be profitably distributed to worker nodes when the splitting/collation overhead is not too large. Typically, bootstrap and Monte Carlo simulations can be readily chunked up for worker nodes. The same applies to predictions from the same fitted model for splittable new data, and similar operations. MCMC computations in chains are time-series dependent, but the chains themselves are independent, so can be distributed to worker nodes. The potential usefulness of the top-down approach will depend on the balance between the extra overhead incurred in the splitting of tasks, the transfer of objects to nodes, and the collation of results from nodes. In

[2] http://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html

some cases, the overhead consumes any benefit from parallel execution, in others, the set-up and collation timings are insignificant, and parallel computation delivers time savings that are directly proportional to the number of nodes in the cluster.

Returning to the bottom-up approach, very detailed structuring of code within BLAS can enhance performance. It is important to note that performance may be enhanced, but may alternatively suffer degradation if the hardware and configuration analysis made by the enhanced BLAS no longer match. It is wise to note that: "using a multi-threaded BLAS can result in taking more CPU time and even more elapsed time (occasionally dramatically so) than using a similar single-threaded BLAS" (R Development Core Team 2010b, p. 33).

The R installation manual (R Development Core Team 2010b) mentions a number of enhanced BLAS options. ATLAS BLAS (Whaley *et al.* 2001; Whaley and Petitet 2005) may be built from source, and the number of threads to be used is set at compile time. It is also available as a binary dynamic-link library for 32-bit Windows for a number of processor architectures.[3] The Core2Duo build of the ATLAS BLAS DLL does not appear to use more than one core under Windows, although execution seems to jump from core to core less than with the BLAS DLL distributed as part of the R Windows binary. As mentioned above, the R OSX binaries by default use the Apple vecLib BLAS,[4] which is also used by many applications, including the main GUI, to speed up operations. It appears that the number of threads used is the maximum available.

GotoBLAS2 is built from source following individual download;[5] it is a further development of GotoBLAS. The analyses reported in Goto and van de Geijn (2008b,a) show that even on cores of equal specification speed, the built GotoBLAS binaries will also differ because they are optimised to the size of the translation look-aside buffer and L2 cache for each core. This is because, with different TLB and L2 cache sizes, matrix chunks of different sizes can be kept close to the CPU registers. The TLB is a local store of recently used pages, avoiding the need to go out to RAM when a reference can be found. The analysis made at compile time takes into account the size configuration of the host machine, customizing the choices to match its architecture as closely as possible. As built, GotoBLAS by default uses the number of cores visible, but by setting an environment variable `GOTO_NUM_THREADS` before starting the application that will load BLAS, the number of threads used can be controlled, albeit not dynamically.

4. Using multiple cores: practice

We will try out a number of spatial tasks, using data sets of varying size, several different computers, operating systems, and BLAS versions. Comparative wall-clock timings will be used to compare runs of the tasks; the timings match manual stopwatch measurements adequately. Because the timings are only taken once for each combination of conditioning factors, small differences are not to be interpreted as real differences, as they may have arisen from other processes on the computers.

Four (five) machines were used for trials, an HP DC7900 8GB, 4 core Q9400 2.66GHz computer (cache 3072 KB per core); an HP DC7800 4GB, 2 core E8500 3.16GHz computer (cache 6144

[3] http://cran.r-project.org/bin/windows/contrib/ATLAS/.
[4] http://developer.apple.com/hardwaredrivers/ve/vector_libraries.html.
[5] http://www.tacc.utexas.edu/tacc-projects/gotoblas2/.

KB per core); a MacBook 4GB, 2 core T8300 2.4GHz computer (cache 3072 KB per core); and a Lenovo T60 Thinkpad 1GB, 2 core T5600 1.83GHz (cache 2048 KB per core). The DC7900 and DC7800 computers were running Red Hat Enterprise Linux, kernel 2.6.18, 64 bits, the MacBook was running OSX 10.5.8, 64 bits. The Thinkpad was running Fedora 12, kernel 2.6.32, 32 bits; it was also running Microsoft Windows XP SP3, 5.1.2600, 32 bits. All were running R 2.10.1, **snow** 0.3-3, **spdep** 0.5-4 and **spgwr** 0.6-6.

R ships with an unthreaded, non-optimised BLAS, which can serve as a baseline for comparison with fast BLAS versions. The four fast BLAS variants used in these trials were the vecLib BLAS provided as standard on the OSX platform, non-threaded ATLAS BLAS for Windows Core 2, downloadable from CRAN as a binary replacement for `Rblas.dll`, and two versions of GotoBLAS built from source on Linux platforms. As discussed above, GotoBLAS interrogates the host system at build time to customise the built dynamically loadable library (shared object); the number of threads used can be set by an environment variable ranging from 1 to the number of cores/processors on the host — here we report one, two (all GotoBLAS2 platforms) and four threads (DC7900 only).

Since two source versions of GotoBLAS were avaliable (here termed GotoBLAS (1.26) and GotoBLAS2 (1.10)), they were both used as they showed differing performance. Results for GotoBLAS are not reported, as trials failed to complete when the number of **snow** workers were equal to the maximum number of BLAS threads. It appears that GotoBLAS2 guards against such competition for resources. Of course, users should be conscious of the dangers of trying to parallelize top-down and bottom-up at the same time, but it is useful to establish how robust parallelization options are to lack of user caution.

4.1. Local Moran's I_i under the alternative

The first task is to reproduce the local Moran's I_i exact calculations under the alternative with no spatial autocorrelation from Bivand *et al.* (2009); details of the statistic may be found in that paper, and the function is `localmoran.exact.alt` in **spdep**. It is run on the Upper Austria data set of 445 communities provided as supplementary material to the original paper. In a footnote on p. 2866, it was noted that the use of a fast threaded BLAS (GotoBLAS) had speeded up the computation of local Moran's I_i exact calculations under the alternative. Exact local Moran's I_i under the alternative involves the calculation of the eigenvalues of an $N \times N$ matrix for each $i, i = 1, \ldots, N$, in addition to some $N \times N$ dense matrix multiplications, and numerical integration using `integrate`. The matrix multiplications and solution of the eigenproblem may benefit from an enhanced BLAS.

Figure 1 shows an ordering the timings of running this task, grouping on whether **snow** is used or not, and on the number of workers if used. In the footnote in the paper referred to above, which provoked Michael Tiefelsdorf to ask for this survey of parallelization for spatial statistics, **snow** was not used.[6] The computer used in that paper is no longer available, but is most similar to the Thinkpad under Fedora 12 without **snow** with GotoBLAS2 using two threads (131s) and with RBLAS (205s). When **snow** is not used, the enhanced BLAS using the maximum number of threads is always dominant.

We can see that increasing the number of **snow** workers helps, and increasing the number of threads in GotoBLAS2 then hinders improvements in elapsed time when using **snow**. On

[6] (Tiefelsdorf 2002) provides an alternative approach to inference from local Moran's I_i using a Saddlepoint approximation

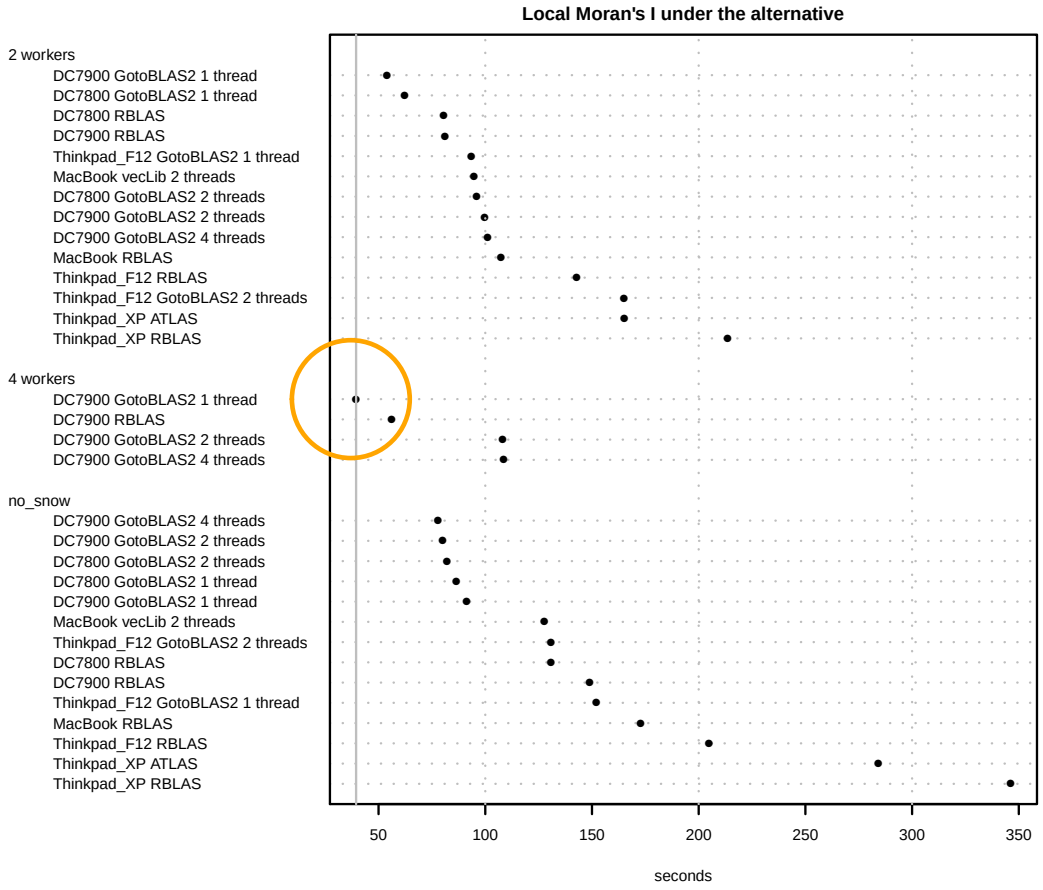

Figure 1: Local Moran's I_i under the alternative; timing comparisons.

all machines, enhanced BLAS on one thread with the maximum number of workers wins, a clear case of an embarrassingly parallel problem. Note that the `vecLib` BLAS with two **snow** workers (95s) is faster than without **snow** (128s), despite competition for resources; this suggests that OSX does a reasonable job of moderating between running processes. The fastest performance was on the DC7900 with four **snow** workers (39s) with a single threaded GotoBLAS2, and RBLAS (56s). Very similar execution times on the same machine with two and four GotoBLAS2 threads (108s) show the degradation caused by competition for performance, here not moderated by the operating system, and that GotoBLAS2 appears to limit the degradation if the user has made unfortunate choices.

4.2. Monte Carlo tests by permutation bootstrap

The second task is to run Monte Carlo tests on the approximate profile-likelihood estimator of spatial autocorrelation in detrended data (APLE, Li *et al.* 2007), using permutation bootstrap; the function used is `aple.mc` in **spdep**, with the exaggeratedly large number of 5000 simulations. The Mercer and Hall wheat yield data ($n = 500$) as presented by Cressie (1993) and used in Li *et al.* (2007) is provided in **spdep**, and is used here too. Each simulation involves two multiplication of sparse matrices of size $N \times N$ by vectors of size N, using methods

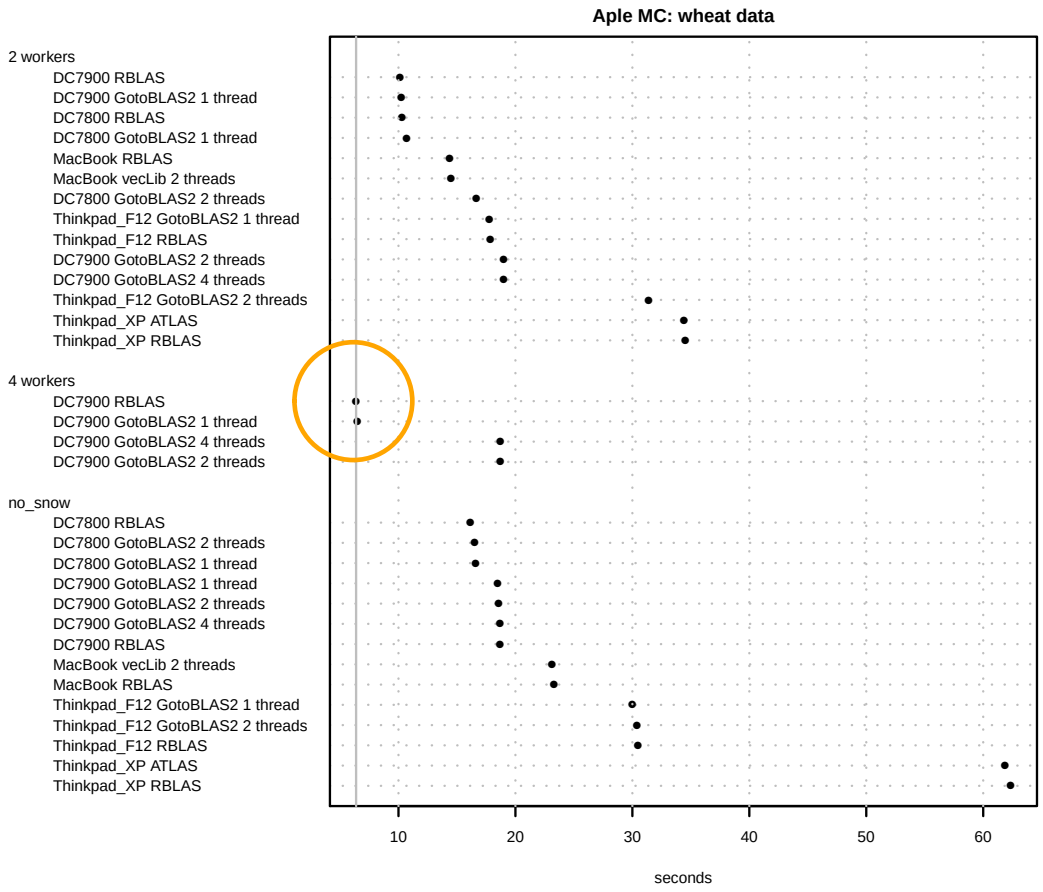

Figure 2: APLE Monte Carlo tests by permutation bootstrap; timing comparisons.

in the **Matrix** package, which are equivalent to level 2 BLAS operations, and so less likely to be speeded by enhanced BLAS for this moderate N.

For some functions in the **spdep** package, a **snow** cluster may be set in an option, which by default is `NULL`, for `ncl` the desired number of nodes:

```
> library(snow)
> cl <- makeCluster(ncl)
> set.ClusterOption(cl)
```

Functions such as Monte Carlo tests may then test internally to see whether a cluster is available, and use it if this is the case. The internal set-up code will then make sure that `clusterSetupRNGstream` is called, and that the required packages are loaded on the worker nodes. The number of simulations will be divided between the nodes, and `parLapply` used to parallelize the call to `boot` in the **boot** package. A final wrapper is used to re-assemble the output object, so that the shapes of objects with and without cluster use agree. Because of the different random number streams, the cluster and non-cluster objects are not identical, although either can be replicated by setting seeds.

Figure 2 shows that Monte Carlo tests by permutation bootstrap are also embarrassingly parallel, and indeed in all three worker variants, the unoptimised R BLAS comes out fastest.

The overall fastest execution uses four workers, with degradation setting in when the BLAS tries to use multiple threads, again because of competition for resources. When **snow** is not used, the platforms display flat responses to changes in BLAS, so in this function for this N, there is no benefit from using an enhanced BLAS, but also no disadvantage.

4.3. Spatial Durbin impact measures

The third task is to carry out Monte Carlo tests on impact measures for a spatial Durbin model, using the Lucas County OH house price data set ($n = 25357$), from the Matlab spatial econometrics toolbox, and provided as a data set in **spdep**. Monte Carlo tests based on simulations of impact measures from fitted spatial lag or spatial Durbin models should be used instead of interpreting their coefficients. The spatial lag model (LeSage and Pace 2009) is the most frequently encountered specification in spatial econometrics:

$$\mathbf{y} = \rho\mathbf{W}\mathbf{y} + \mathbf{X}\beta + \varepsilon,$$

where $\mathbf{y}$ is an $(N \times 1)$ vector of observations on a dependent variable taken at each of N locations, $\mathbf{X}$ is an $(N \times k)$ matrix of exogenous variables, β is an $(k \times 1)$ vector of parameters, ε is an $(N \times 1)$ vector of independent and identically distributed disturbances and ρ is a scalar spatial lag parameter.

In the spatial Durbin model, the spatially lagged exogenous variables are added to the model:

$$\mathbf{y} = \rho\mathbf{W}\mathbf{y} + \mathbf{X}\beta + \mathbf{W}\mathbf{X}\gamma + \varepsilon,$$

where γ is an $((k-1) \times 1)$ vector of parameters where $\mathbf{W}$ is row-standardised, and a $(k \times 1)$ vector otherwise. It is clear that these two models may be estimated in the same way, for example by maximum likelihood. The specific problem lies in the fact that the spatial dependence in the parameter ρ feeds back, obliging analysts to base interpretation not on the fitted parameters β, and γ where appropriate, but rather on correctly formulated impact measures (LeSage and Pace 2009, p. 33–42). For the spatial Durbin model, the impact measures for $\mathbf{X}$ variable r is defined using $S_r(\mathbf{W}) = (\mathbf{I} - \rho\mathbf{W})^{-1}(\beta_r\mathbf{I} + \gamma_r\mathbf{W})$. The direct impact for variable r is taken as the average of the diagonal elements of $S_r(\mathbf{W})$, the total impact the average of all elements of $S_r(\mathbf{W})$, and the indirect impact the difference between these two (LeSage and Pace 2009; Elhorst 2010).

Impact measures for a fitted model may be fitted readily by expressing the intractible $(\mathbf{I} - \rho\mathbf{W})^{-1}$ term as a series of traces of powers of $\mathbf{W}$, which may be approximated by Monte Carlo methods (LeSage and Pace 2009, p. 114–115). It is more difficult to infer from these measures, as they are not simple functions of the estimated coefficient standard errors. The prefered approach is to draw samples from the multivariate distribution described by the fitted coefficient values and their variance-covariance matrix, and conduct a Monte Carlo test on the impact measures. In **spdep** `impacts` methods, the multivariate samples are drawn in one operation, rather than in bootstrap fashion or by MCMC-style random walk — this is to avoid repeated decompositions of the variance-covariance matrix in `mvrnorm` from the **MASS** package (Ripley 1987, p. 98). From then on, it appears that running the sequence of impact measures calculations by row on the matrix of draws of coefficient values could be susceptible to parallelization, by splitting the sequence into chunks.

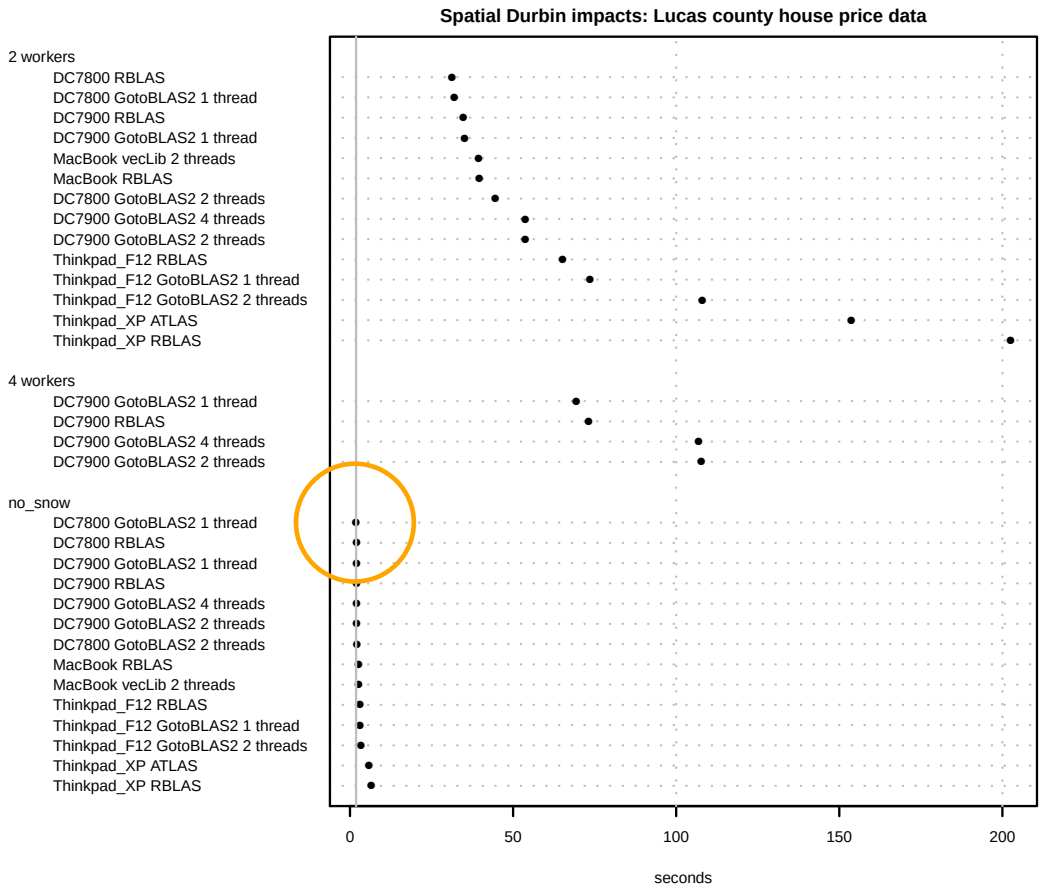

Figure 3: Monte Carlo tests on impact measures for a spatial Durbin model; timing comparisons.

In fact, as Figure 3 shows clearly, the reverse is the case, and **snow** workers just get in the way. Examining only the no-worker variants, we see that threaded and optimised BLAS do not speed up the computation. It appears that the overhead involved in spreading matrix multiplication across threads for quite small matrices is too high, and is detected as such by GotoBLAS2. Of course, the initial setup and the collation of results does not involve linear algebra. It remains possible that these serialization and communication operations between the administrator node and the worker nodes are unnecessarily burdensome, if the objects moved become larger than needed through scoping; this will be investigated.

4.4. Spatial filtering

Spatial filtering approaches the accommodation of spatial autocorrelation in regression residuals by including chosen eigenvectors of a doubly-centred spatial weights matrix in the regression model, to "whiten out" the spatial patterns. The eigenvectors represent a decomposition of spatial patterns encoded in the spatial weights matrix, and their chosen subset constitutes a "model" of the autocorrelation. (Tiefelsdorf and Griffith 2007) describe how the eigenvectors for inclusion may be selected from among the N candidates by brute force, reducing the

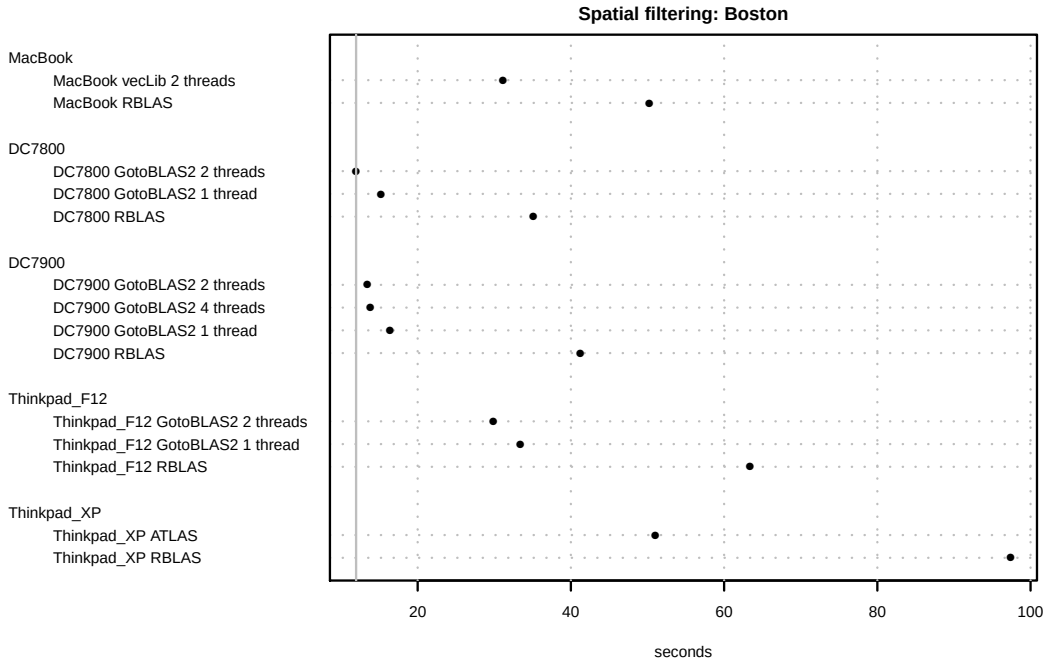

Figure 4: Spatial filtering; timing comparisons not using **snow**.

residual autocorrelation step-by-step. Each step involves many matrix multiplications, but the brute force search does not appear to offer any opportunities for top-down parallelization.

Consequently, the only source of faster execution are threaded and optimised BLAS, to the extent that these subroutines are called. Here we see that the best choice is the faster DC7800 with a double-threaded GotoBLAS2 on cores with larger L2 cache, about three times faster than R BLAS, on the moderately sized Boston hedonic house prices data set $N = 506$. Interestingly, the GotoBLAS2 two and four thread timings on the four-core platform with smaller L2 cache are very similar, despite all four cores being engaged when four threads were used. Perhaps this reflects extra overhead when moderate sized matrices are being reorganized for computing on multiple cores for the L2 cache resources available. Trying out different sized data sets might show when this overhead becomes worthwhile.

4.5. Geographically weighted regression

Geographically weighted regression as introduced by Brunsdon *et al.* (1998) involves first choosing a bandwidth (for example by cross-validation) for a two-dimensional kernel placed on the points at which data was collected for fitting weighted linear models, then using the kernel with this bandwidth to weight the distances between data points and fit points. These weights are then used to fit a weighted linear model at each fit point using the same data and bandwidth, but obviously dfferent weights at each fit point. The intention is to examine any spatial patterns that may emerge from mapping the local coefficients at the fit points, which may be the same as the data points (Fotheringham *et al.* 2002); these patterns may be helful in identifying missing variables.

Harris *et al.* (2010) show that both steps may be parallelized on a Grid: in the cross-validation

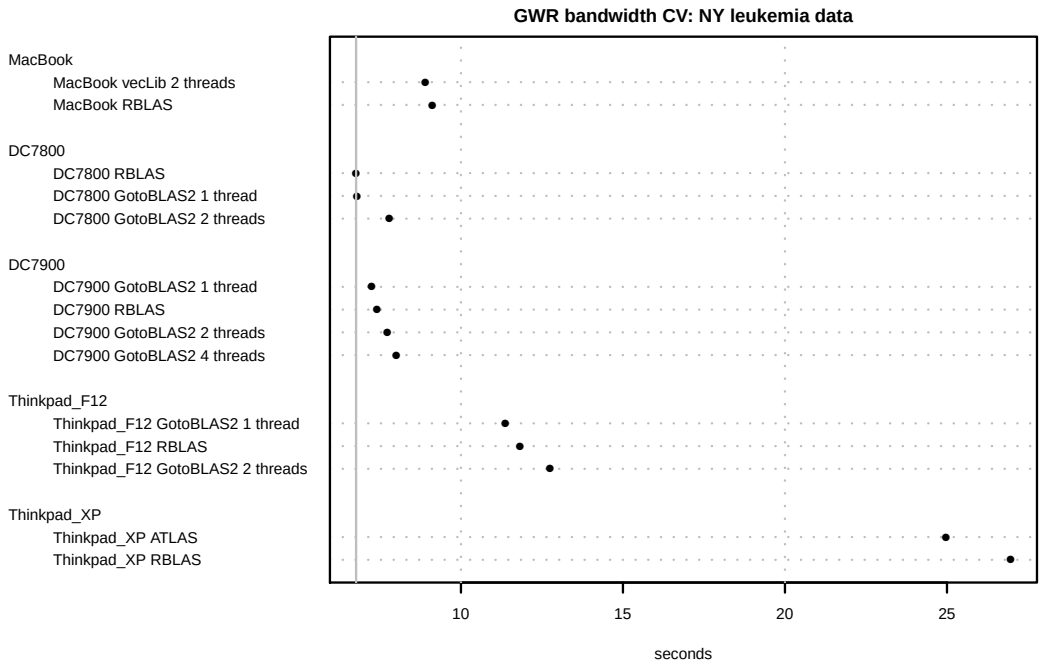

Figure 5: Geographically weighted regression bandwidth cross-validation at N data points; timing comparisons not using **snow**.

step over the data points, the distances between blocks of data points can be computed once and recycled if the regressions are distributed across nodes. If the nodes are separate processing entities, each with plenty of memory, the splitting of the $N \times N$ distance matrix by row makes sense, but on a single multi-core computer, this is not the case. In the computation of the geographically weighted regressions at the fit points, it is clear that spreading the fit points across nodes will lead to savings in execution time.

Using the NY leukemia data ($N = 281$) presented in Waller and Gotway (2004), we first examine the performance of standard `gwr.sel` in the **spgwr** package for cross-validation bandwidth selection without **snow** — no parallelization of this step was attempted. Figure 5 shows little variation in performance between the BLAS variants examined by platform. It is a little worrying that GotoBLAS2 using a number of threads equal to the number of cores always comes out slower than single-threaded alternatives. It is possible that the repeated execution of `lm.wfit` for this N involves a mixture of matrix operations that misleads the attempts of GotoBLAS2 to optimize its work across the cores.

Figure 6 shows clearly that splitting the fit points for which geographically weighted regressions are to be estimated is worthwhile for this number of fit points. As we have seen before, when top-down parallelization is being used, permitting an optimized threaded BLAS to compete for resources leads to serious degradation of performance. The best timings of 4.1s using the fore-core machine with an unenhanced BLAS and 4.3s with a single-threaded GotoBLAS2 on the same platform can be compared on the same machine with no **snow** parallelization of 10.9s and 10.5s repectively. When four cores are used by **snow**, the four core GotoBLAS2 (11.4s) is slower than when **snow** is not used. The same degree of performance degradation is

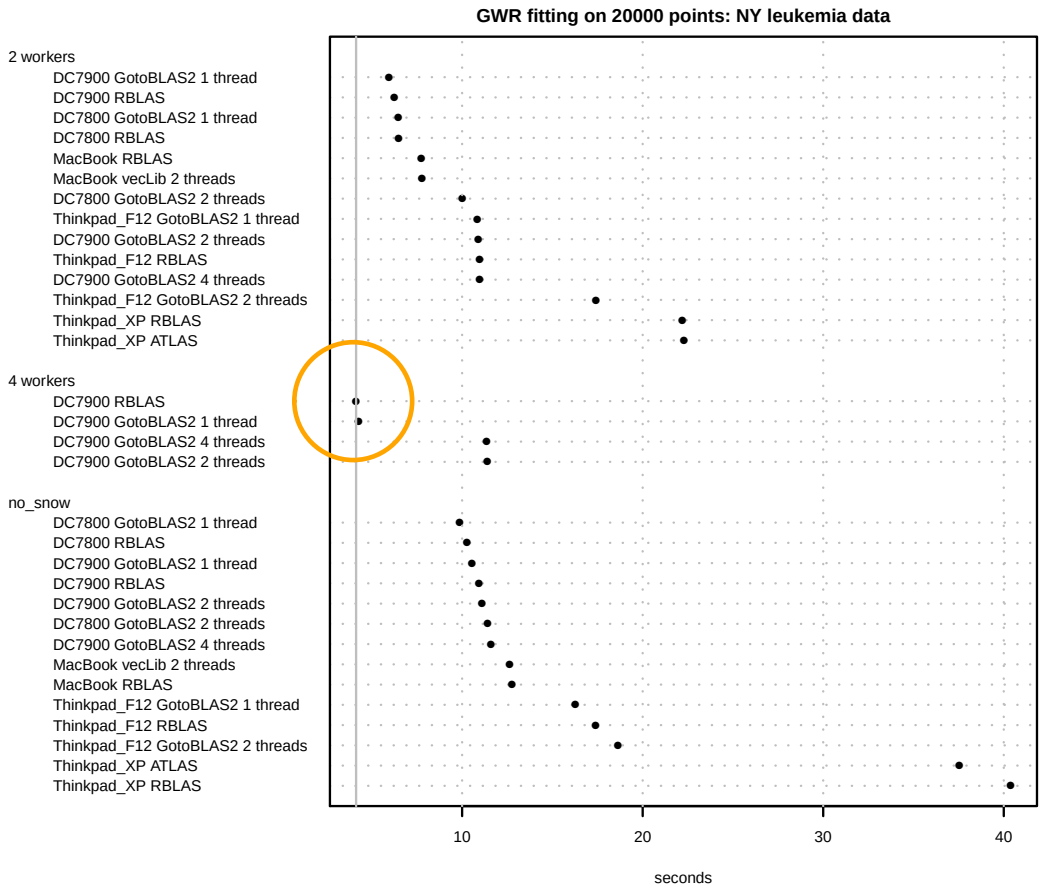

Figure 6: Geographically weighted regression — fitting at a grid of 20,000 fit points; timing comparisons.

not observed for vecLib BLAS, which is less aggressively optimized for matrix multiplication than GotoBLAS2. Using **snow** for top-down parallelization on the OSX platform reduces execution times to 7.7s for both BLAS from 12.7s for both BLAS.

5. Conclusions

As we have seen, understanding how the interpreted languages used for coding "view" the hardware resources is important. Balancing coding effort with flexibility and choice offered to users is also important, so that users have the tools needed to get their work done in a customised way. In some cases this means advocating the use of multicore architectures as top-down clusters, in others as resources for bottom-up threaded and optimised computation, and in yet others as not directly relevant. In addition, the advice one might give for a given method might change depending on the size of the data being analysed, both the number of spatial entities, and the number of variables involved.

Only the R binary for OSX comes with a fast BLAS library enabled by default; its performance when competing for resources with snow workers is acceptable. On some tasks the vecLib

BLAS clearly dominates the standard R BLAS (local Moran's I_i and spatial filtering), on others they perform similarly, but in none of these cases does the use of the vecLib BLAS lead to degraded performance. One can conclude that the analyst may rely on the vecLib BLAS to give reasonable performance on tasks similar in nature and size to those run here, also when top-down parallelization is used, but subject to the reservations made in the the R for Mac OSX FAQ.

The Windows R ATLAS BLAS is only available for some processor architectures; building one's own is possible under CygWin, but demands considerable skill. Again, in some cases the ATLAS BLAS used here performed better than the standard R BLAS (local Moran's I_i, spatial filtering, GWR bandwidth cross-validation and GWR fitting); again, in no case is standard BLAS very much better than ATLAS BLAS. If an ATLAS BLAS binary is available that matches the platform in use, it may lead to better performance in some cases, such as the spatial filtering example, which involved many matrix and vector operations.

On the same hardware, but running under Fedora 12, the standard R BLAS is sometimes faster than ATLAS BLAS under Windows, sometimes a little slower than it, but always faster than the Windows standard R BLAS. Under Fedora 12, GotoBLAS2 offers further opportunities for enhanced performance. All of the platforms compared here are running other software at the same time, and a proportion of the "attention" of the operating system, and of the available capacity of the processors, is absorbed in "housekeeping", such as running event loops on input devices, and watching for incoming network traffic. Although extraneous programs were not executed during these trials, it does appear that different operating systems absorb different levels of processing resources; naturally, this can be overcome by upgrading the hardware rather than by switching operating system.

Finally, it is important to recognise that top-down and bottom-up parallelization approaches do not mix well. Although threaded GotoBLAS2 recognizes resource race situations, its performance is seriously degraded when being run multi-threaded on multiple cluster nodes causing resource competition. For tasks for which top-down parallelization is chosen, it is important to restrict the number of threads in enhanced BLAS such as GotoBLAS to one only.[7]

In the context of **snow** socket clusters, it appears to be possible to start each node using the appropriate environment variable to ensure that the enhanced BLAS on each node only uses one thread, reducing possible competition to that between any multi-threaded BLAS on the administrator node and the R processes on the worker nodes:

```
> library(snow)
> rscript <- paste("GOTO_NUM_THREADS=1", paste(R.home("bin"), "Rscript",
+     sep = .Platform$file.sep))
> cl <- makeCluster(ncl, type = "SOCK", rscript = rscript)
```

As we have seen, parallelization may be exploited in spatial statistics. It is much harder to recommend how it should be exploited, whether by top-down or bottom-up approaches, or a combination of the two approaches varying with the kinds and sizes of the computational tasks being undertaken. Indeed, the effort involved in trying to parallelize a task may be better spent in improving the efficiency of existing code, or resources may rather be channelled

[7] See Kazushige Goto's comment "If you use your multi-threaded code which calls BLAS, you should use single threaded version", https://lists.tacc.utexas.edu/pipermail/gotoblas/2009-March/000177.html; the comment applies to compiled and linked applications, but is relevant here.

to hardware upgrades. However, when much linear algebra on sufficiently large matrices is involved, an enhanced BLAS may be a sensible choice, and when a task really is embarrassingly parallelizable, it probably should be parallelized if the available hardware permits.

References

Bivand RS, Müller W, Reder M (2009). "Power Calculations for Global and Local Moran's *I*." *Computational Statistics and Data Analysis*, **53**, 2859–2872.

Bivand RS, Pebesma EJ, Gómez-Rubio V (2008). *Applied Spatial Data Analysis with R*. Springer, New York.

Brunsdon C, Fotheringham S, Charlton M (1998). "Geographically weighted regression — modelling spatial non-stationarity." *The Statistician*, **47**(3), 431–443.

Cressie N (1993). *Statistics for Spatial Data, Revised Edition*. John Wiley & Sons, New York.

Elhorst JP (2010). "Applied Spatial Econometrics: Raising the Bar." *Spatial Economic Analysis*, **5**(1), 9–28.

Fotheringham AS, Brunsdon C, Charlton ME (2002). *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. Wiley, Chichester.

Fox J (2009). "Aspects of the Social Organization and Trajectory of the R Project." *The R Journal*, **1**(2), 5–13. URL http://journal.r-project.org/archive/2009-2/RJournal_2009-2_Fox.pdf.

Goto K, van de Geijn R (2008a). "High-performance implementation of the level-3 BLAS." *ACM Transactions on Mathematical Software*, **35**(1 (4)).

Goto K, van de Geijn RA (2008b). "Anatomy of high-performance matrix multiplication." *ACM Transactions on Mathematical Software*, **34**(3 (12)).

Harris R, Singleton A, Grose D, Brunsdon C, Longley P (2010). "Grid-enabling Geographically Weighted Regression: A Case Study of Participation in Higher Education in England." *Transactions in GIS*, **14**(1), 43–61.

Hijmans RJ, van Etten J (2010). *raster: Geographic analysis and modeling with raster data*. R package version 1.4-10, URL http://CRAN.R-project.org/package=raster.

Knaus J, Porzelius C, Binder H, Schwarzer G (2009). "Easier Parallel Computing in R with snowfall and sfCluster." *The R Journal*, **1**(1), 54–59. URL http://journal.r-project.org/archive/2009-1/RJournal_2009-1_Knaus+et+al.pdf.

L'Ecuyer P, Simard R, Chen E, Kelton W (2002). "An object-oriented random-number package with many long streams and substreams." *Operations Research*, **50**(6), 1073–1075.

Leisch F, Rossini AJ (2003). "Reproducible Statistical Research." *Chance*, **16**, 46–50.

LeSage J, Pace R (2009). *Introduction to Spatial Econometrics*. CRC Press, Boca Raton, FL.

Li H, Calder CA, Cressie N (2007). "Beyond Moran's I: Testing for spatial dependence based on the spatial autoregressive model." *Geographical Analysis*, **39**, 357–375.

R Development Core Team (2010a). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

R Development Core Team (2010b). *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, Austria.

R Development Core Team (2010c). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria.

Rey S (2009). "Show me the code: spatial analysis and open source." *Journal of Geographical Systems*, **11**, 191–207.

Ripley BD (1987). *Stochastic Simulation*. John Wiley & Sons, New York.

Rossini A, Tierney L, Li N (2007). "Simple parallel statistical computing in R." *Journal of Computational and Graphical Statistics*, **16**, 399–420.

Schmidberger M, Morgan M, Eddelbuettel D, Yu H, Tierney L, Mansmann U (2009). "State of the Art in Parallel Computing with R." *Journal of Statistical Software*, **31**(1), 1–27. URL http://www.jstatsoft.org/v31/i01.

Tiefelsdorf M (2002). "The Saddlepoint approximation of Moran's I and local Moran's I_i reference distributions and their numerical evaluation." *Geographical Analysis*, **34**(3), 187–206.

Tiefelsdorf M, Griffith DA (2007). "Semiparametric Filtering of Spatial Autocorrelation: The Eigenvector Approach." *Environment and Planning A*, **39**(5), 1193–1221.

Tierney L (2009). "Code analysis and parallelizing vector operations in R." *Computational Statistics*, **24**(2), 217–223.

Tierney L, Rossini A, Li N (2009). "Snow: A Parallel Computing Framework for the R System." *International Journal of Parallel Programming*, **37**, 78–90.

Waller LA, Gotway CA (2004). *Applied Spatial Statistics for Public Health Data*. John Wiley & Sons, Hoboken, NJ.

Whaley RC, Petitet A (2005). "Minimizing development and maintenance costs in supporting persistently optimized BLAS." *Software: Practice and Experience*, **35**(2), 101–121.

Whaley RC, Petitet A, Dongarra JJ (2001). "Automated Empirical Optimization of Software and the ATLAS Project." *Parallel Computing*, **27**(1–2), 3–35.

Affiliation:

Roger Bivand
Department of Economics

Norwegian School of Economics and Business Administration
Helleveien 30, 5045 Bergen, Norway
E-mail: Roger.Bivand@nhh.no

NHH

Norges Handelshøyskole

Norwegian School of Economics
and Business Administration

NHH
Helleveien 30
NO-5045 Bergen
Norway

Tlf/Tel: +47 55 95 90 00
Faks/Fax: +47 55 95 91 00
nhh.postmottak@nhh.no
www.nhh.no