Using SPRINT and parallelised functions for analysis of large data on multi-core Mac and HPC platforms

Eilidh Troup[1*], Thorsten Forster[2], Luis Cebamanos[1], Terence Sloan[1], Peter Ghazal[2]

1. Edinburgh Parallel Computing Centre, University of Edinburgh, Edinburgh, UK
2. Division of Pathway Medicine, University of Edinburgh Medical School, Edinburgh, UK

*Contact author: e.troup@epcc.ed.ac.uk

Overview

Motivation

How to use SPRINT

SPRINT Implementation

SPRINT Functions

Performance

Case study

Overview

Motivation

How to use SPRINT

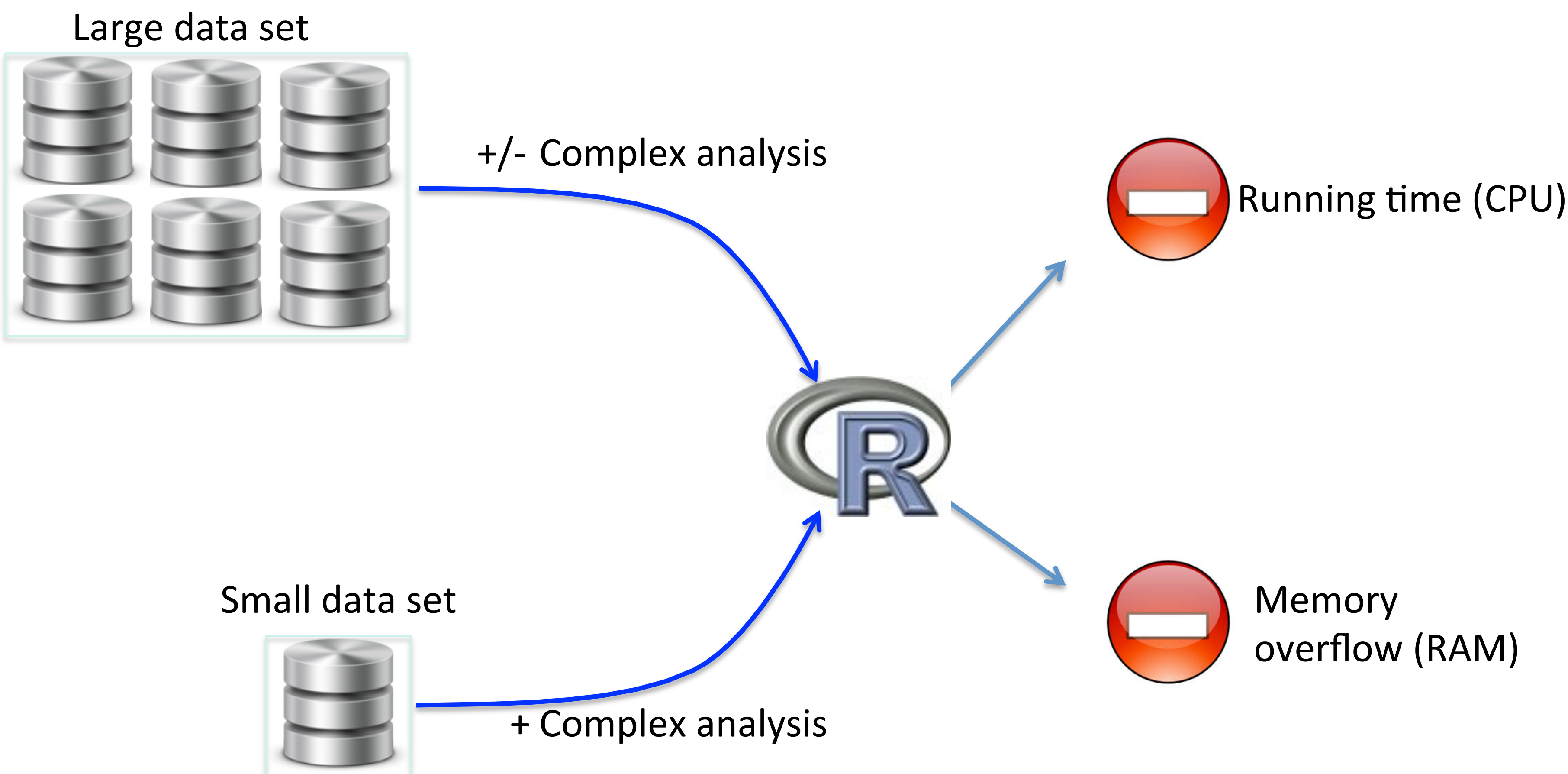
SPRINT Implementation

SPRINT Functions

Performance

Case study

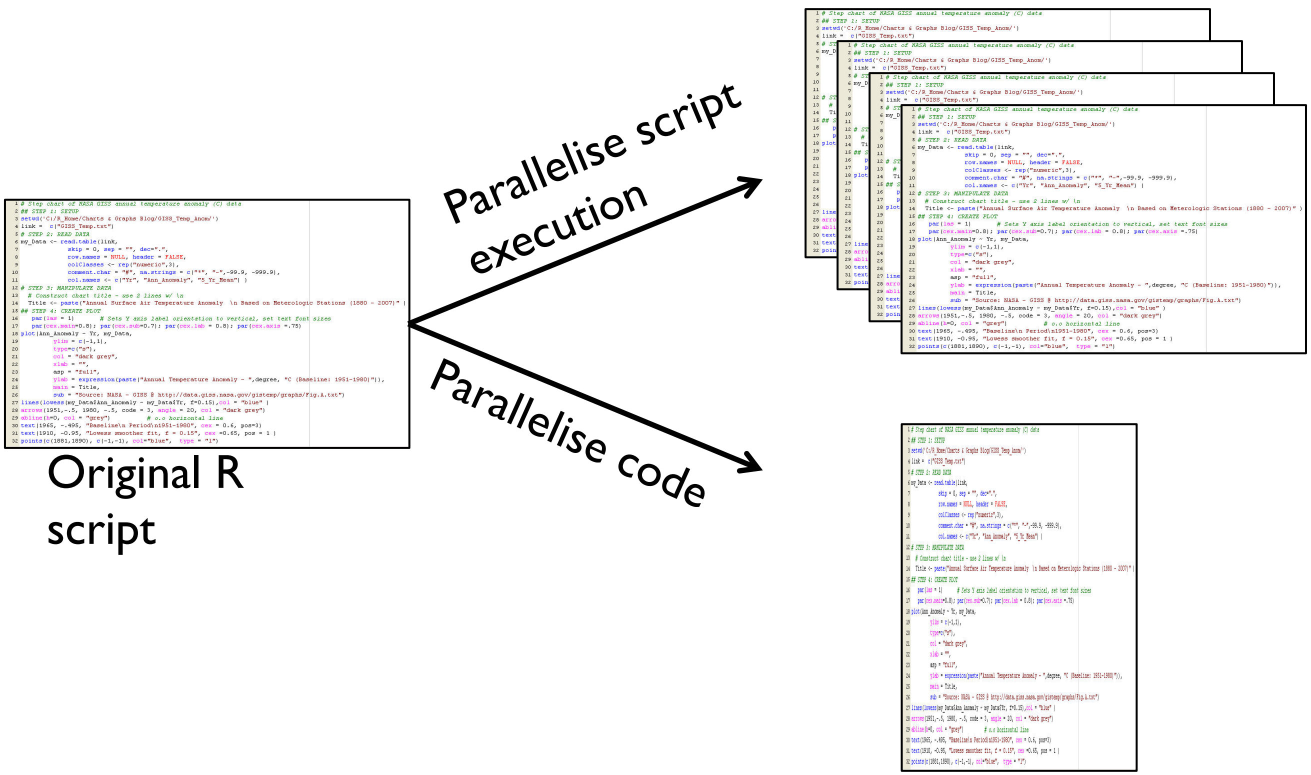
R performance bottlenecks



R solutions for parallelisation and memory usage

1. Simplify analysis, reduce data set, batch process

2. Use R functionality to parallelise code and extend memory (on supercomputers, clusters, multicore machines)



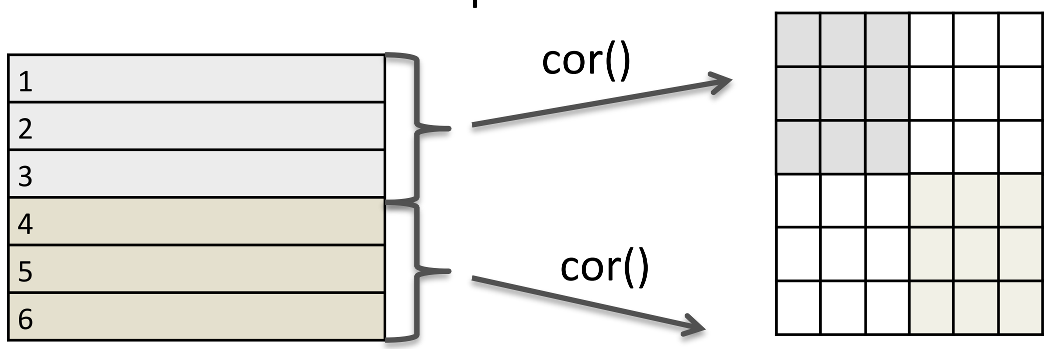
OS command line parallel script submission

parallel()
rmpi()
snow()
ff()
bigmemory()
…

For extensive resources, see Dirk Eddelbuettel's HPC task view:
http://cran.r-project.org/web/views/HighPerformanceComputing.html

Not all functions are easy to parallelise.

- Correlation for example.



- Clustering is another example where the data cannot be considered separately.
- Other SPRINT functions provide optimised implementations, or handle larger datasets.

SPRINT approach

Overcomes limitations on **data size** and **analysis time** and by providing easy access to HPC for all R users

Photo: Mark Sadowski

Simple Parallel R INTerface (www.r-sprint.org)

“SPRINT: A new parallel framework for R”, J Hill et al, BMC Bioinformatics, Dec 2008.

Overview

Motivation

How to use SPRINT

SPRINT Implementation

SPRINT Functions

Performance

Case study

Example

```
my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- cor( t(my.matrix) )


quit(save="no”)
```

Example

```
library("sprint")

my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- cor( t(my.matrix) )


quit(save="no")
```

Example

```
library("sprint")

my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- pcor( t(my.matrix) )


quit(save="no")
```

Example

```
library("sprint")

my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- pcor( t(my.matrix) )

pterminate()

quit(save="no")
```

How to run

sprint_script.R

```
library("sprint”)
my.matrix <- matrix(rnorm(500000,9,1.7), nrow=20000, ncol=25)
genecor <- pcor( t(my.matrix) )
pterminate()
quit(save="no”)
```

```
$ mpiexec -n 4 R -f sprint_script.R
```

Overview

Motivation

How to use SPRINT

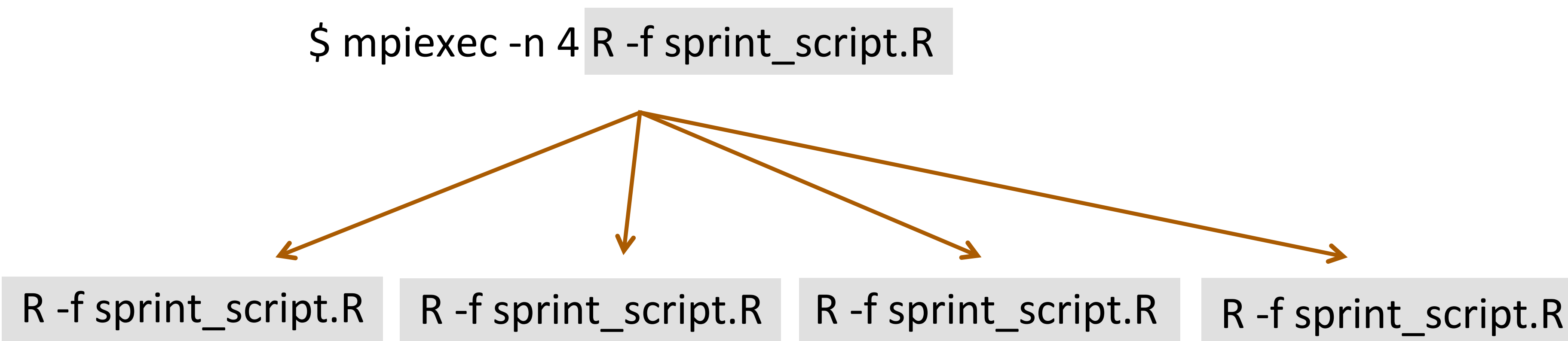
SPRINT Implementation

SPRINT Functions

Performance

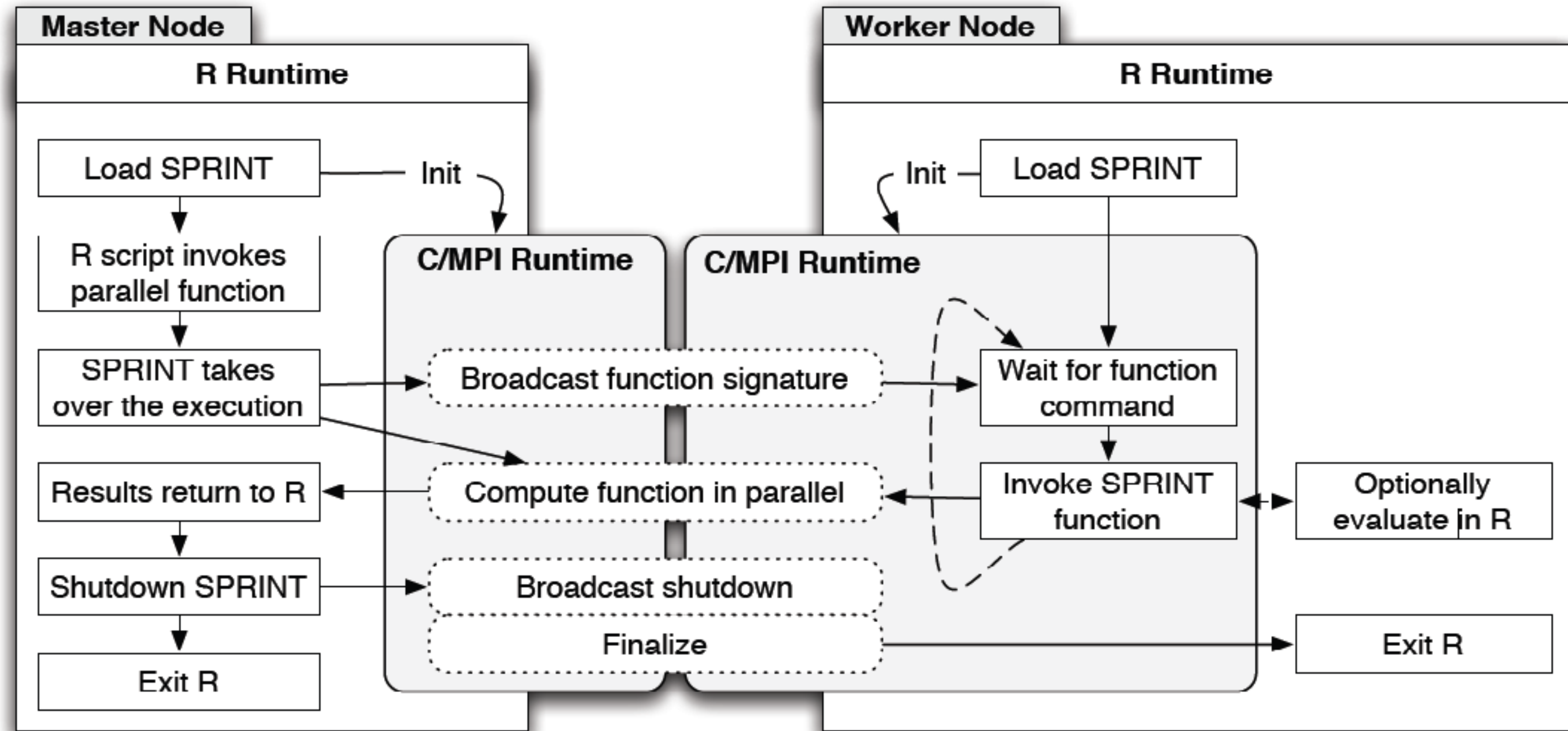
Case study

mpiexec


$ mpiexec -n 4 R -f sprint_script.R
R -f sprint_script.R
R -f sprint_script.R
R -f sprint_script.R
R -f sprint_script.R

SPRINT architecture

SPRINT overview schema, for those interested in implementation
(not important to using SPRINT)



Overview

Motivation

How to use SPRINT

SPRINT Implementation

SPRINT Functions

Performance

Case study

Parallelised SPRINT functions

***p*cor()**	Pearson correlation for pairs of numeric variables
***p*pam()**	Partitioning-Around-Medoids clustering
***p*randomForest()**	Random Forest classification algorithm
pmaxt()	Permutation-adjusted p-values
***p*RP()**	Rank-Product non-parametric statistical permutation-based test
***p*svm()**	Support-Vector-Machine classification algorithm
***p*stringdistmatrix()**	Hamming distance for pairs of character strings
***p*apply()**	Apply any function to each row/column in a matrix
***p*boot()**	Bootstrap estimates for any given statistic/function
p*dist()	A variety of distance metric to compute (dis)similarity of data vectors

pcor()

Pearson correlation for pairs of numeric variables

	Obs1	Obs2	Obs3	Obs4	Obs5	Obs6	Obs7	ObsN
Var1								
Var2								
Var3								
Var4								
Var5								
Var6								
Var7								
Var8								
Var9								
Var10								
Var11								
Var12								
Var13								
Var14								
VarP								

Input data
(*N* rows)

Perform correlation on all possible pairs of variables.

Output – Correlation matrix
(“adjacency matrix”, “similarity matrix”)

	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1
Var1	1														
Var2		1													
Var3			1												
Var4				1											
Var5					1										
Var6						1									
Var7							1								
Var8								1							
Var9									1						
Var10										1					
Var11											1				
Var12												1			
Var13													1		
Var14														1	
VarP															1

(1-correlation matrix = distance matrix)

(N^2 correlation coefficients)

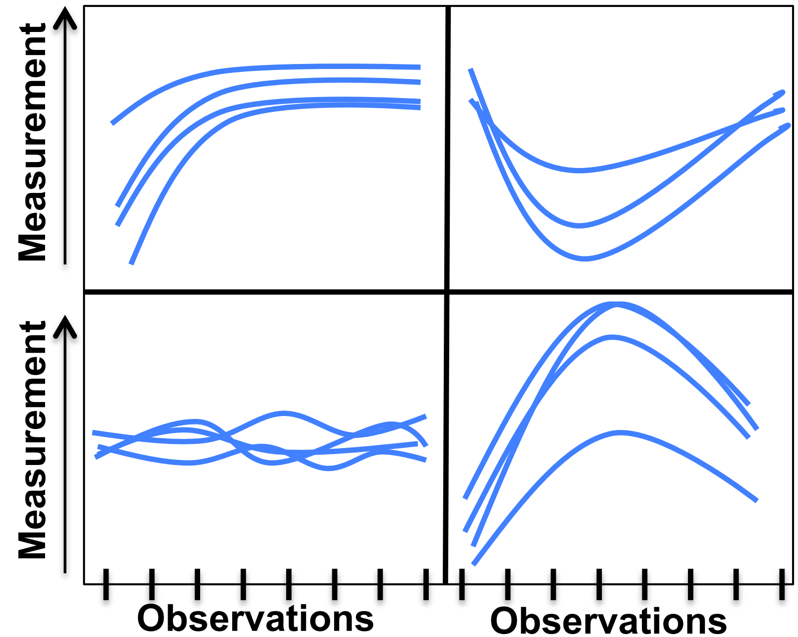
ppam()

Partitioning-Around-Medoids clustering

Input data

	Obs1	Obs2	Obs3	Obs4	Obs5	Obs6	Obs7	ObsN
Var1								
Var2								
Var3								
Var4								
Var5								
Var6								
Var7								
Var8								
Var9								
Var10								
Var11								
Var12								
Var13								
Var14								
VarP								

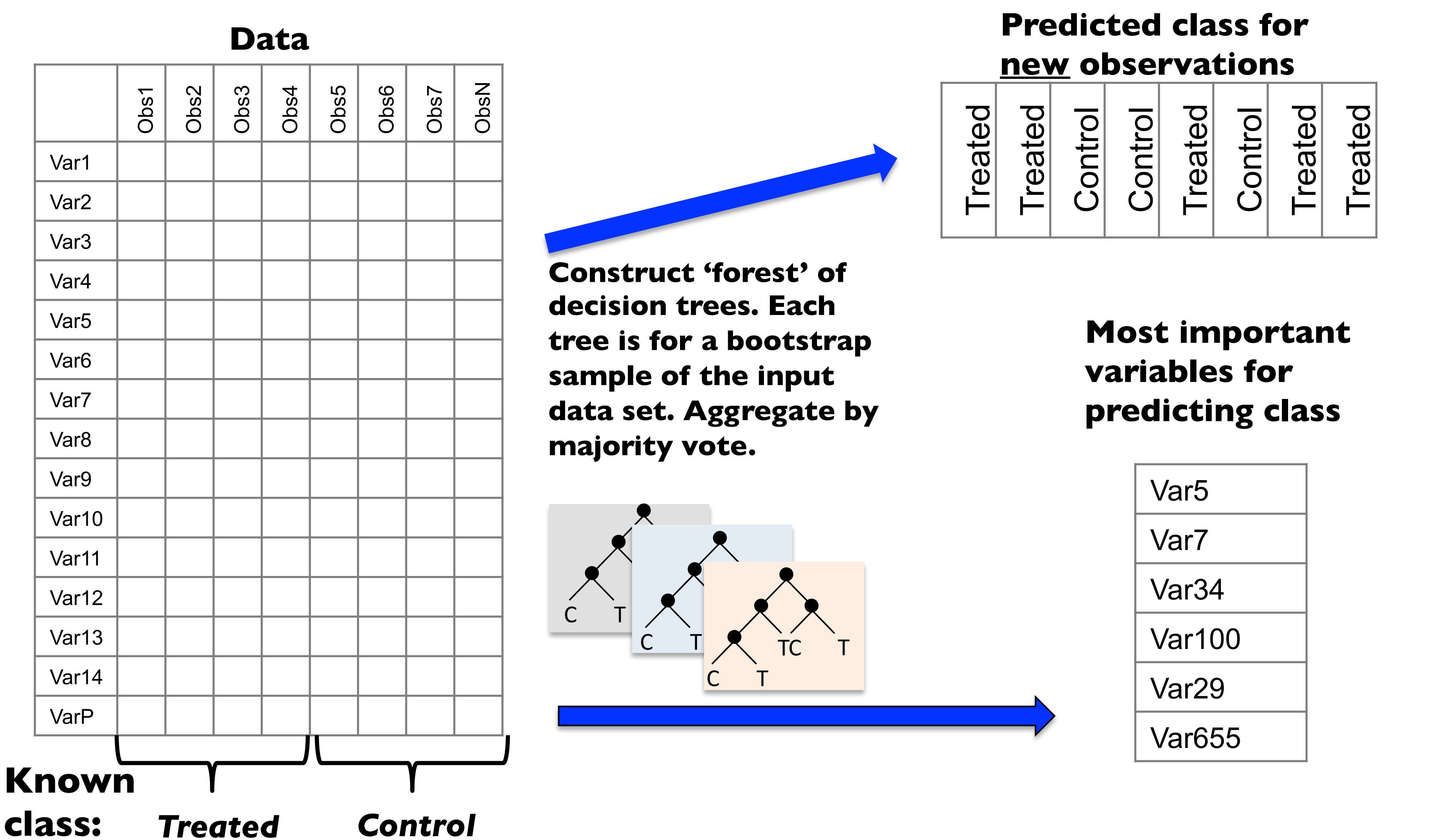
Compute *distance* between all possible pairs of variables, then *partition* into sets of maximal distinction



4 clusters

Optimisation and parallelisation of the partitioning around medoids function in R.
Piotrowksi M. et al. BILIS 2011, Jul 2011.

prandomForest() Random Forest classification algorithm

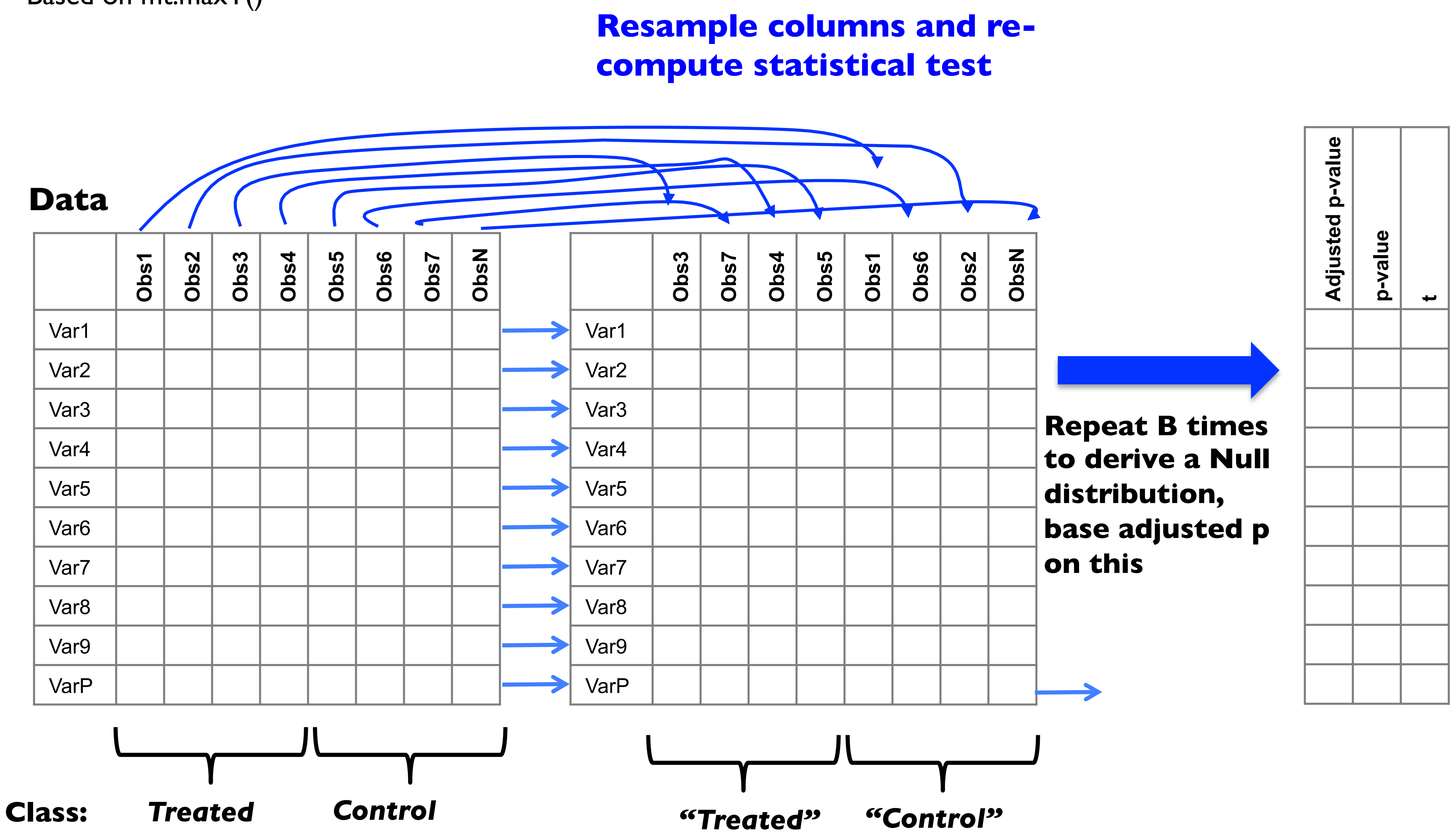


A parallel random forest classifier for R, L. Mitchell et al, HPDC 2011, Jun 2011.

pmaxT()

Based on mt.maxT()

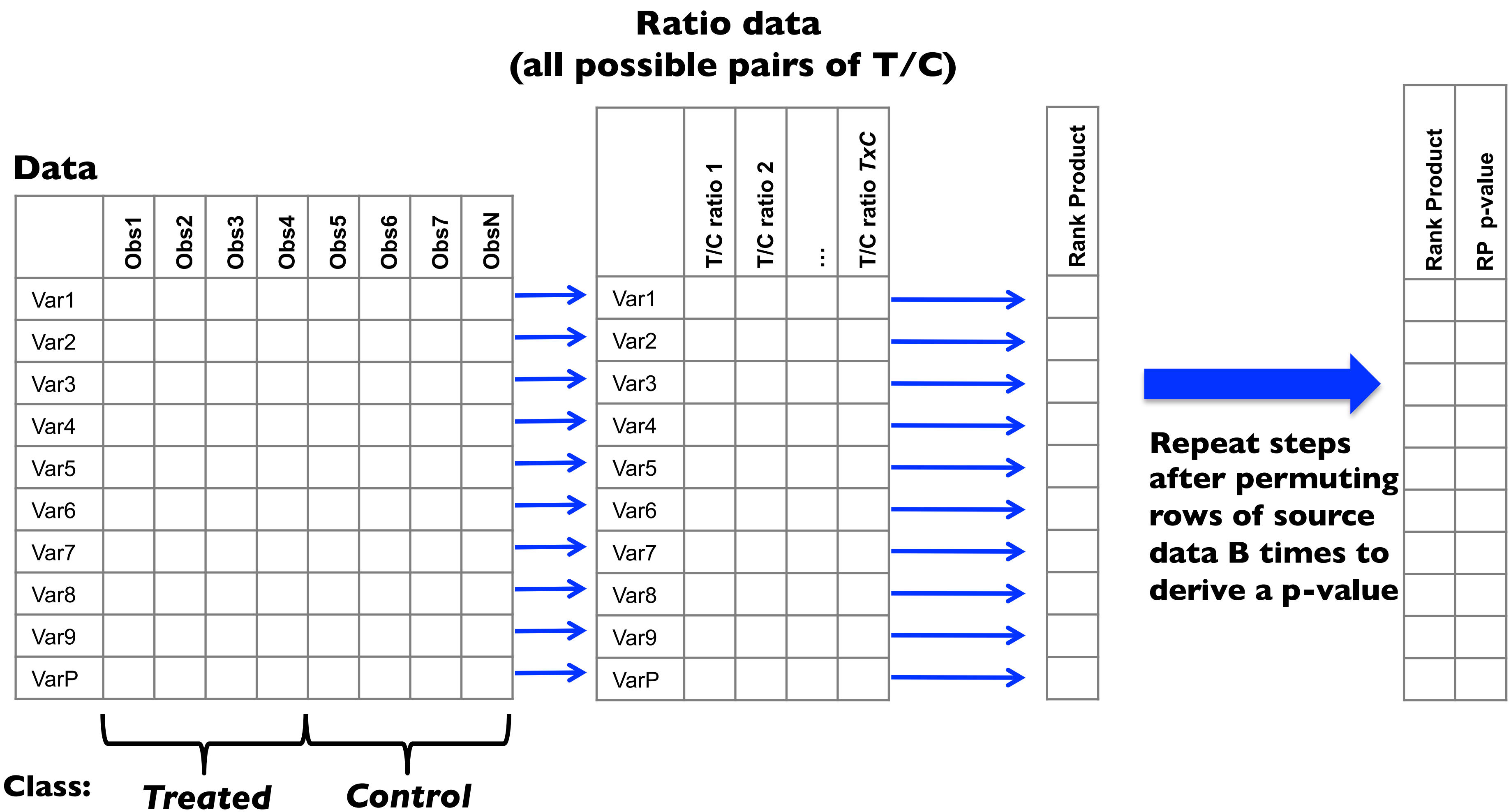
Permutation-adjusted p-values



Optimization of a parallel permutation testing function for the SPRINT R package, S. Petrou et al, Concurrency and Computation: Practice and Experience, Jun 2011.

pRP()

Rank-Product non-parametric statistical permutation-based test



Parallel classification and feature selection in microarray data using SPRINT.
Mitchell L. et al. 2012. Concurrency and Computation: Practice and Experience.

pstringdistmatrix()

Hamming distance for pairs of character strings

Input data= string vector

abc	acd	abc	…	bcd	ada

(N strings)

Calculate string alignment on all possible string pairs.

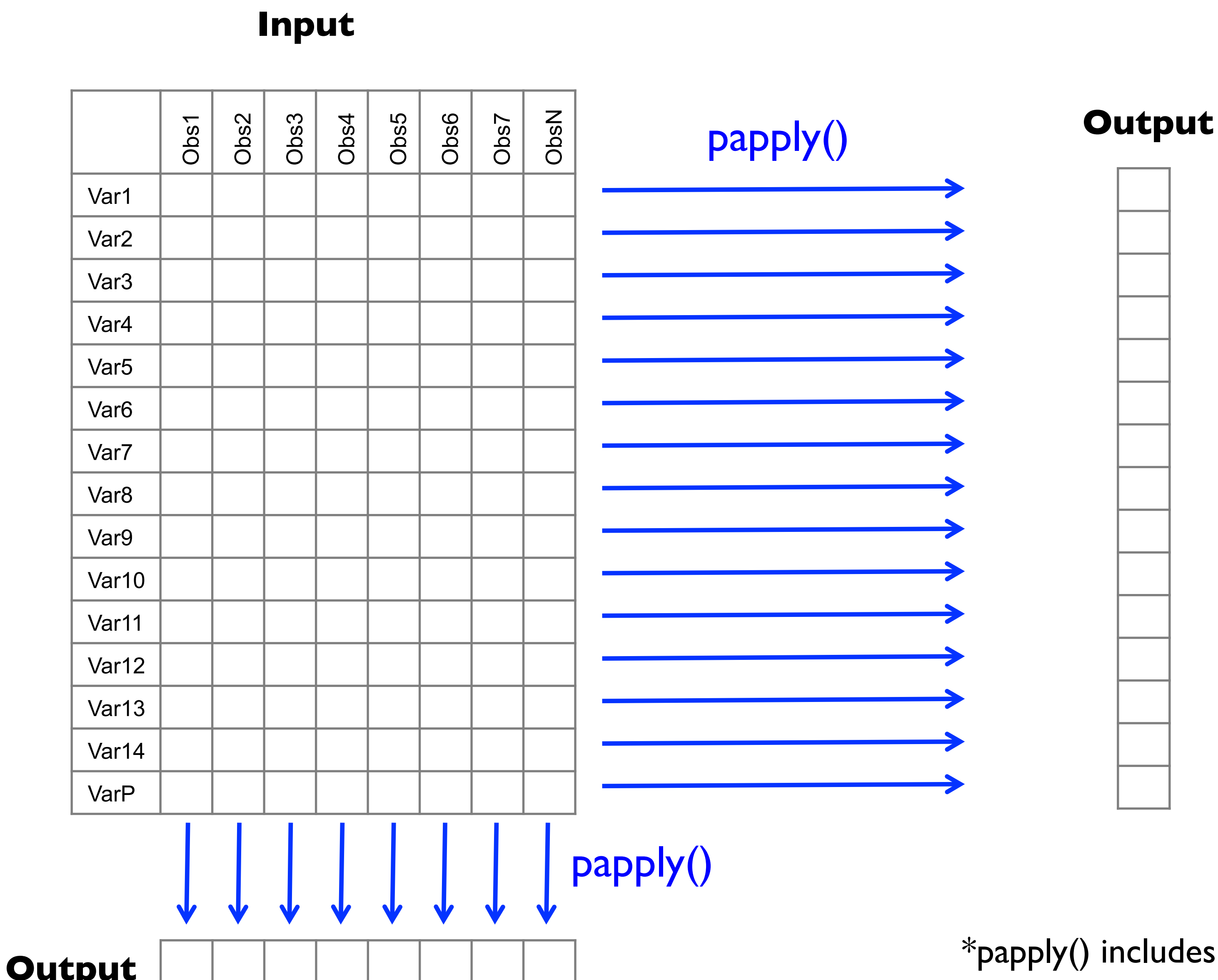
Distance matrix

	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1	Var1
Var1	0														
Var2		0													
Var3			0												
Var4				0											
Var5					0										
Var6						0									
Var7							0								
Var8								0							
Var9									0						
Var10										0					
Var11											0				
Var12												0			
Var13													0		
Var14														0	
VarP															0

(*N^2* alignment scores)

papply()

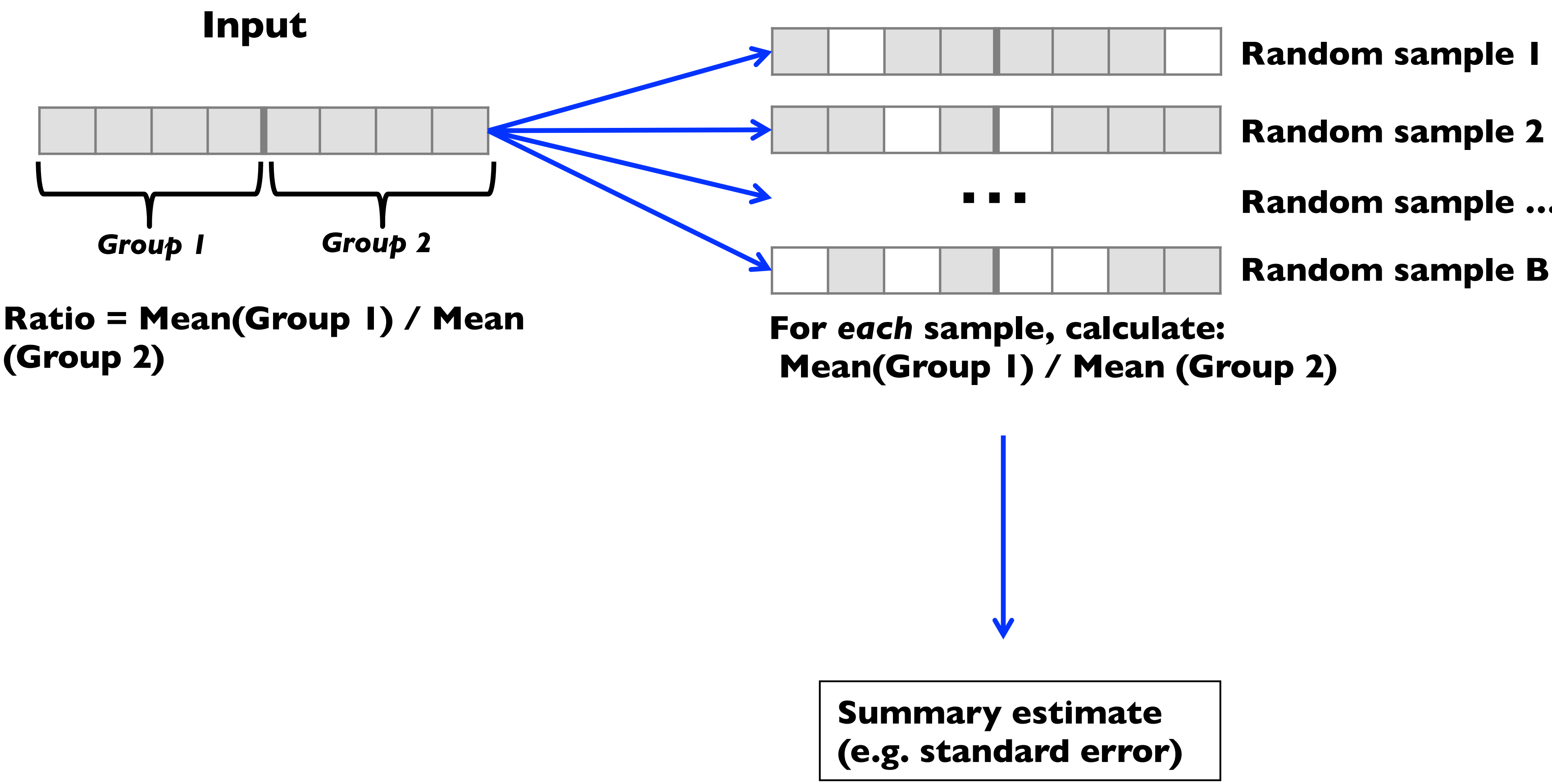
Apply any function to each row or column in a matrix*



*papply() includes lapply() functionality

pboot()

Bootstrap estimates for any given statistic/function



Parallel Optimisation of Bootstrapping in R. Sloan TM, Piotrowski M, Forster T, Ghazal P. arXiv.org pre-publication January 2014.

Overview

Motivation

How to use SPRINT

SPRINT Implementation

SPRINT Functions

Performance

Case study

SPRINT and Data Size

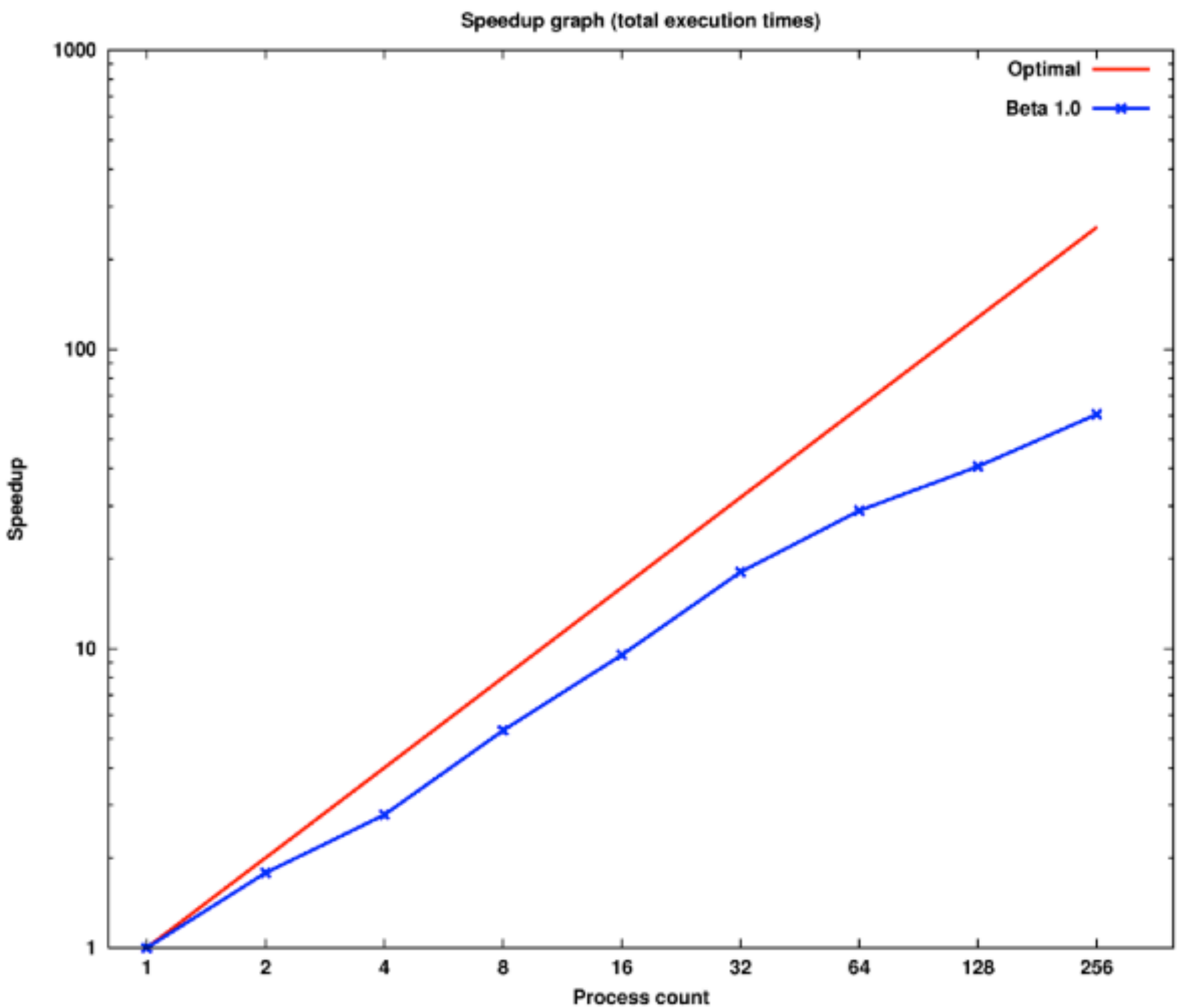
<u>Overcome limitations on data size</u> and analysis time by providing easy access to High Performance Computing for all R users

Input Matrix Size	Output Matrix Size	Serial Run Time	Parallel Run Time
11,000 x 320 **26.85 MB**	**0.9 GB**	**63.18 secs**	**4.76 secs**
22,000 x 320 **53.7 MB**	**3.6 GB**	**Insufficient memory**	**13.87 secs**
35,000 x 320 **85.44 MB**	**9.12 GB**	**Crashed**	**36.64 secs**
45,000 x 320 **109.86 MB**	**15.08 GB**	**Crashed**	**42.18 secs**

Benchmark on HECToR - UK National Supercomputing Service on 256 cores.
S. Petrou et al, dCSE NAG Report, www.r-sprint.org.

For example, Pearson's correlation, pcor() enables processing of datasets where the output does not fit in physical memory using R ff package.

Performance increase – pcor()



The pcor() function scales well. (Shown here up to 256 cores).

SPRINT and Analysis Time

__Overcome limitations on__ data size and *__analysis time__* by providing easy access to High Performance Computing for all R users

Input Matrix Size	# Permutations	Serial Run Time (estimated)	Parallel Run Time
36,612 x 76	**500,000**	**6 hrs**	**73.18 secs**
36,612 x 76	**1,000,000**	**12 hrs**	**146.64 secs**
36,612 x 76	**2,000,000**	**23 hrs**	**290.22 secs**
73,224 x 76	**500,000**	**10 hrs**	**148.46 secs**
73,224 x 76	**1,000,000**	**20 hrs**	**294.61 secs**
73,224 x 76	**2,000,000**	**39 hrs**	**591.48 secs**

Benchmark on HECToR - UK National Supercomputing Service on 256 cores.
S. Petrou et al, HPDC 2010 & CCPE, 2011.

For example, permutation testing, pmaxT() is a parallel implementation of mt.maxT() from multtest package (available from CRAN)

SPRINT Data Size and Analysis Time

<u>Overcome limitations on data size and analysis time</u> by providing easy access to High Performance Computing for all R users

Input Data Size	# Clusters	Serial Run Time Pam()	Parallel Run Time Ppam()
10 000	**24**	**99 mins**	**1.2 mins**
22 374	**24**	**Insufficient memory**	**4.5 mins**

Benchmark on a shared memory cluster with 8 dual-core 2.6GHz AMD Opteron processors with 2GB of RAM per core.
M. Piotrowski et al, BILIS 2011.

For example, clustering with partitioning around medoids, ppam()

- Parallel implementation of pam() from cluster package (available from CRAN)
- Optimisation of serial version through memory and data storage management
- Increased capacity by using external memory (i.e. ff objects)

Overview

Motivation

How to use SPRINT

SPRINT Implementation

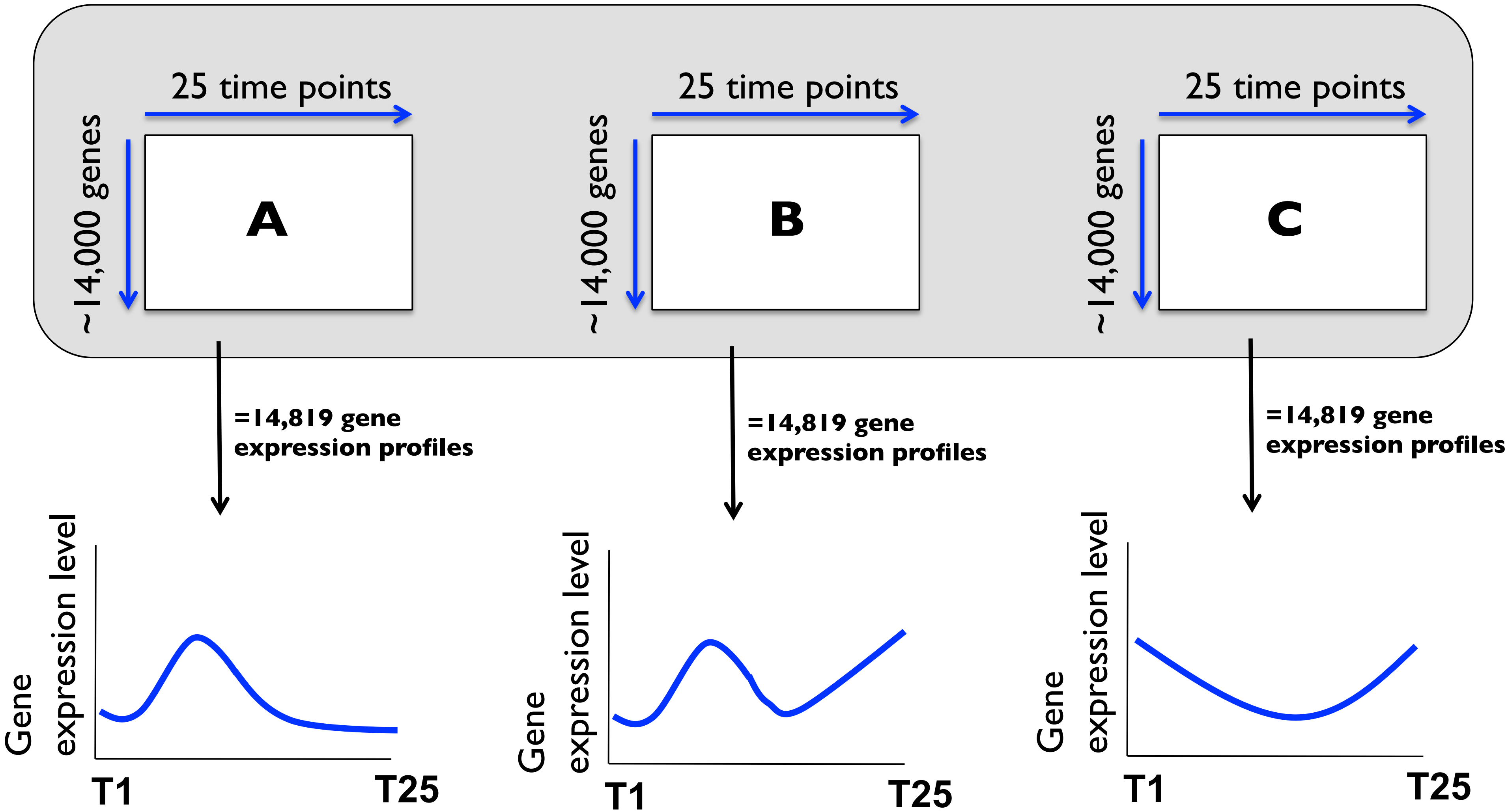
SPRINT Functions

Performance

Case study

Case study

3 microarray gene expression time courses
(each a data matrix of 14,819 rows x 25 columns)



Usual approach

To measure correlation of gene expression profiles within each of the data matrices OR between the 3 possible pairs of data matrices:

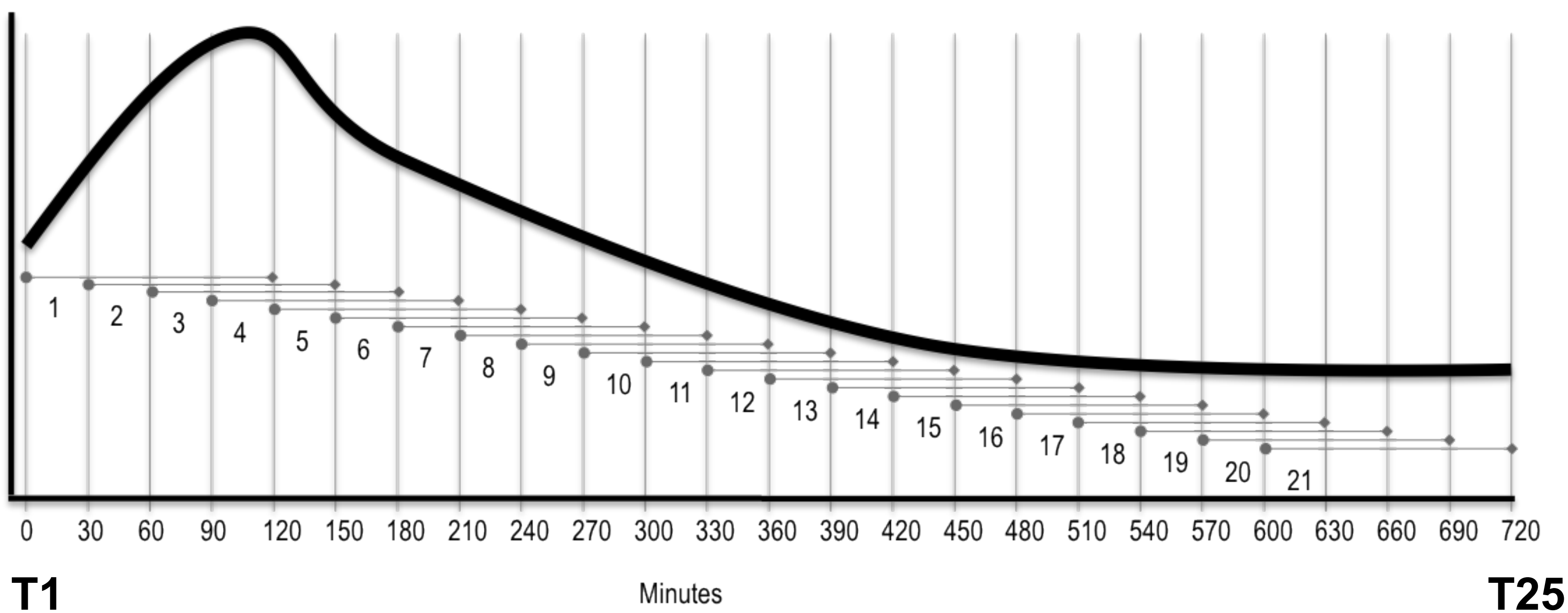
$N = 3 \times 14{,}819^2 \approx$ **659 million** correlation computations

BUT…

But we wanted to expand on this

We want to look at **time-shifted** correlations, where part of each gene's expression profile in one of the data sets could match a part of another gene's in another of the data sets.

We split each gene's expression profile into 21 overlapping time windows of length 5 (= 2 hours)



pcor() use case

Serial cor(), computing all correlations of 2-hour time windows **between** 2 data sets

(14819 x 21 time windows)^2 ≈ **97 billion** calculations

Fails

Serial cor() with reduced number of genes (5561)

(5561x 21 time windows)^2 ≈ **14 billion** calculations

~3 hours

Same computation in parallel (and using 'ff' package to exceed RAM constraints) on full gene set with SPRINT pcor()

~10 min

SPRINT future

Biomedical research projects will drive parallelisation of R functionality

- Ensembl learning (multiple classification algorithms and multiple classification parameter values) with clinical microarray data sets to diagnose/prognose disease

- Data fusion of clinical and biological data sets

…but we're open to collaborations if there are specific problems to solve

SPRINT

EPCC

Eilidh Troup
Luis Cebamanos
Terence Sloan (PI)

DPM

Thorsten Forster
Peter Ghazal (PI)

Former Contributors and Funders

Muriel Mewissen
Savaas Petrou
Michal Piotrowski
Jon Hill
Florian Scharinger
Laurence Baldwin
Bartek Dobrzelecki
Lawrence Mitchell
Kevin Robertson
Andy Turner

BBSRC bioscience for the future

wellcome trust

www.software.ac.uk

nag

edikt e-science data information & knowledge transformation

www.r-sprint.org