Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

R, paralelización, datos masivos y aplicaciones web: ejemplos del uso de R en bioinformática

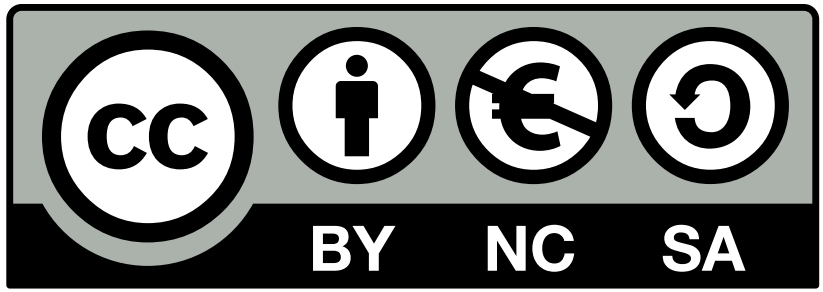
Ramón Díaz-Uriarte

Dept. Bioquímica
Universidad Autónoma de Madrid
Madrid, Spain
rdiaz02@gmail.com
http://ligarto.org/rdiaz

III Jornadas de Usuarios de R
17-Noviembre-2011

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

License and copyright



Outline

Context

Parallelizing code

Web applications

Large data sets and parallelization

R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Context

Biological context

Computational context

Parallelizing code

Web applications

Large data sets and parallelization

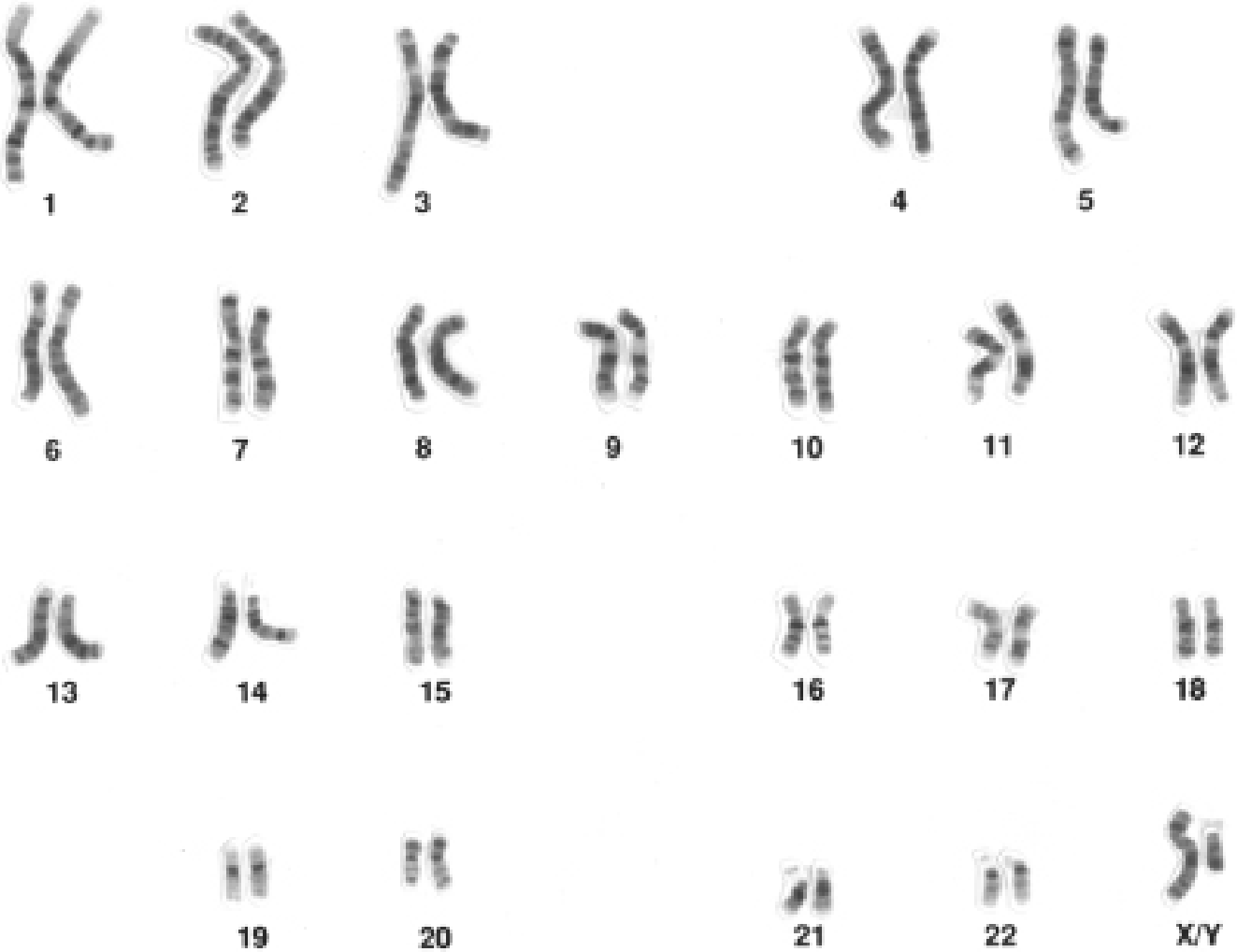
R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Chromosomes

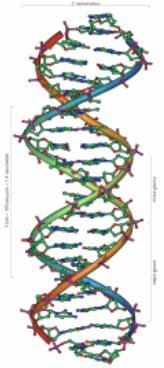


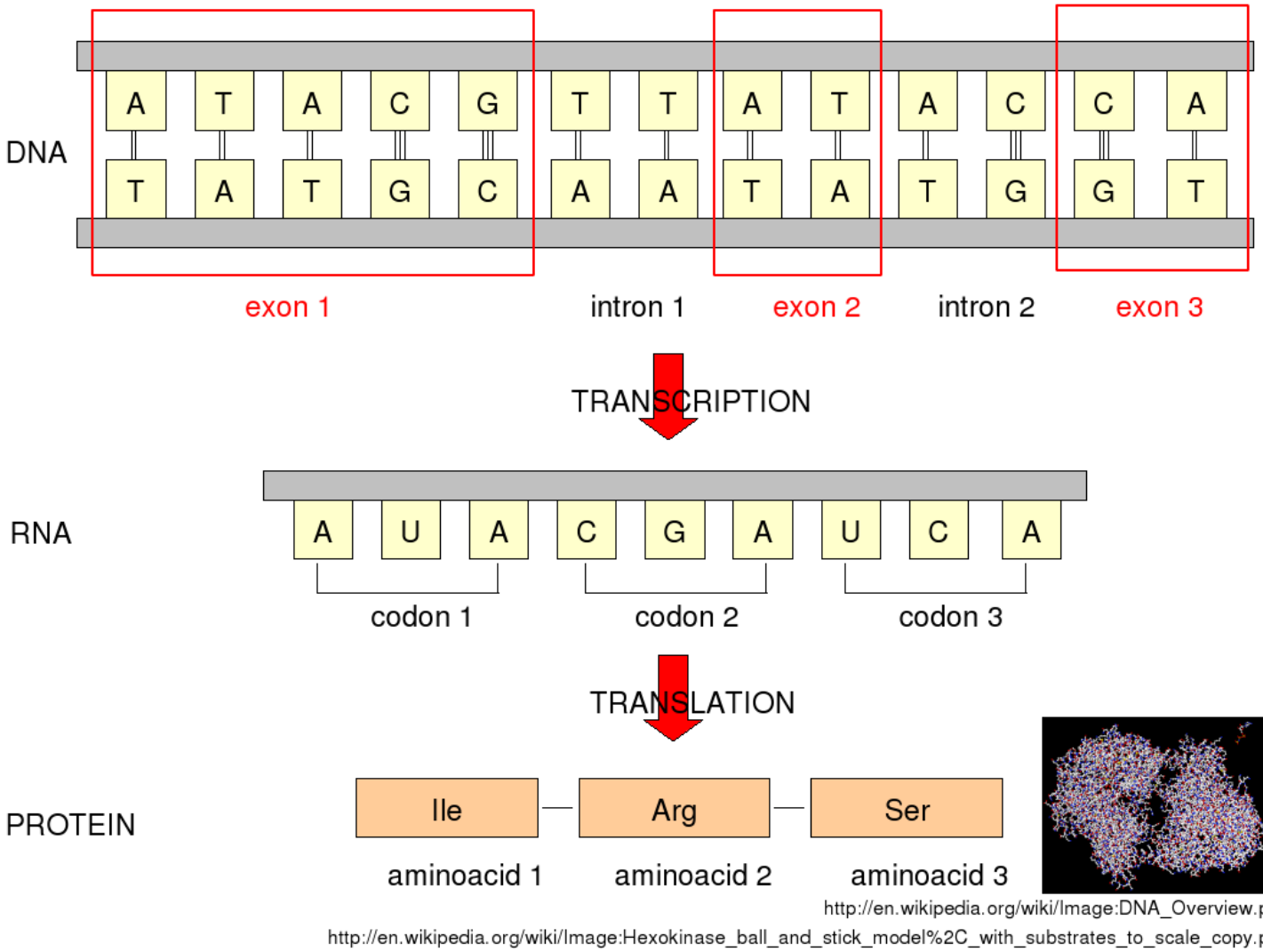
From the Wikipedia; original source

`http://www.genome.gov/Pages/Hyperion//DIR/VIP/Glossary/Illustration/karyotype.shtml`

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

DNA → protein



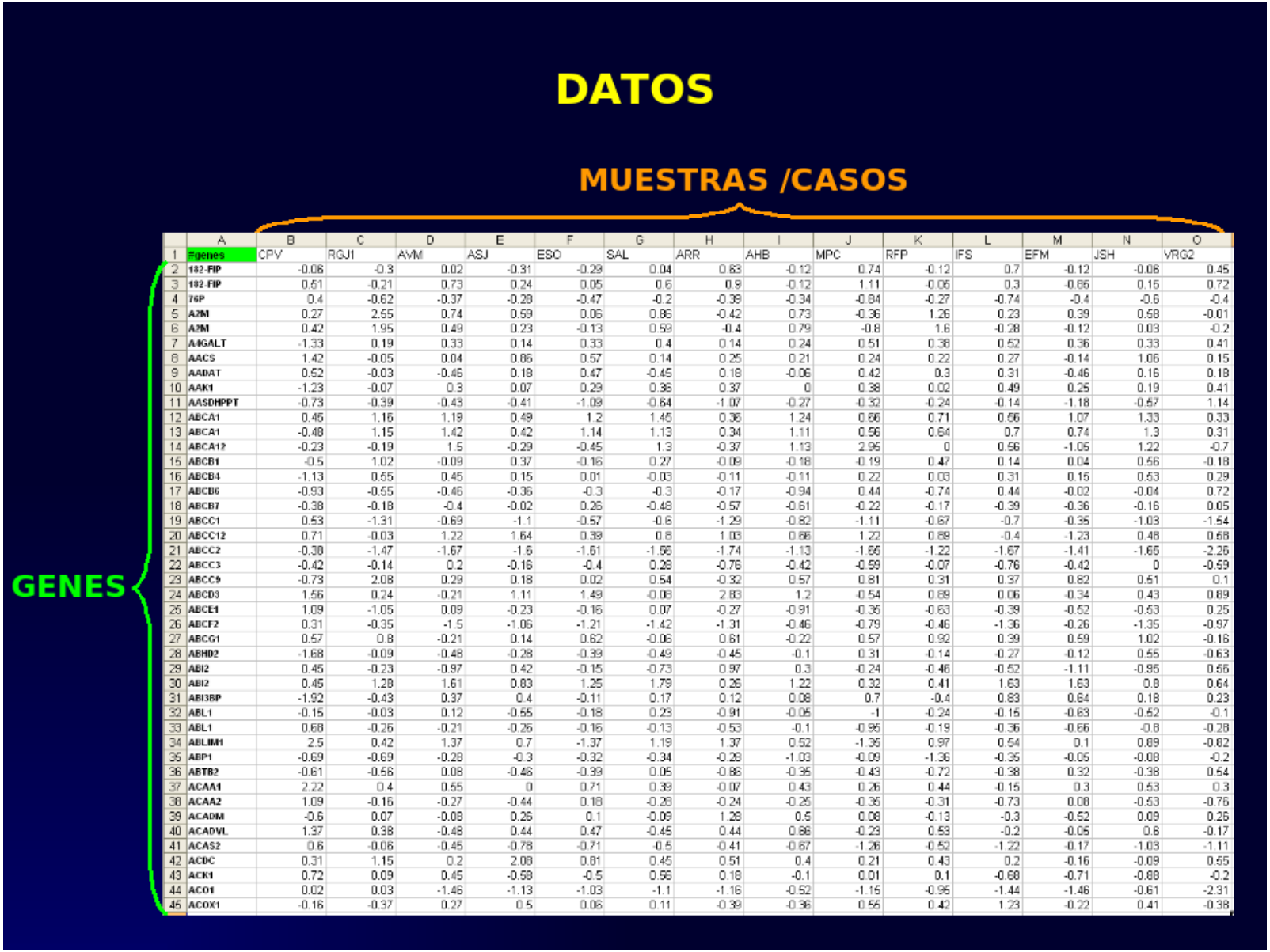


http://en.wikipedia.org/wiki/Image:DNA_Overview.png
http://en.wikipedia.org/wiki/Image:Hexokinase_ball_and_stick_model%2C_with_substrates_to_scale_copy.png

(From O. Rueda's PhD Thesis)

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Data from a microarray experiment

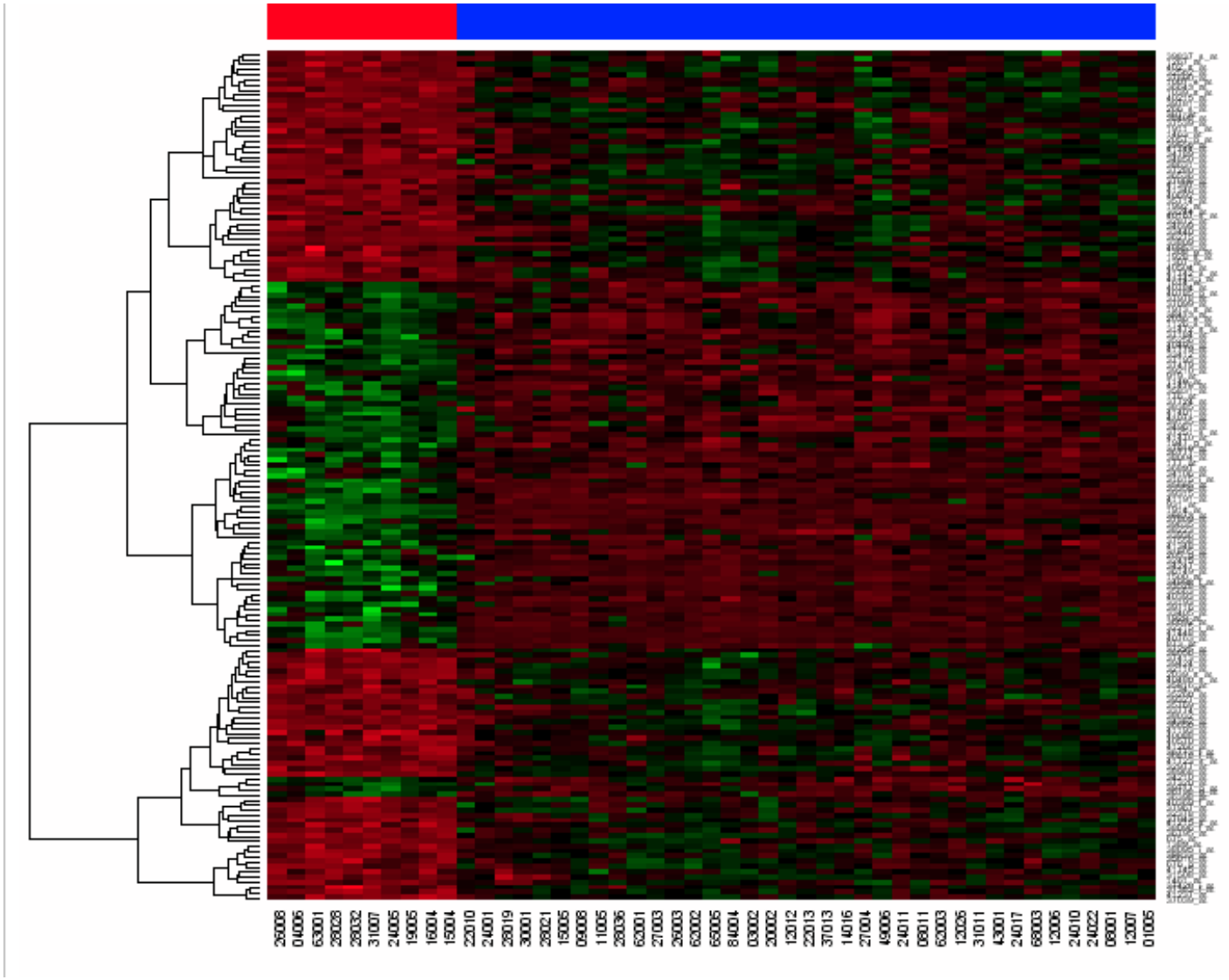


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	#genes	CPV	RGJ1	AVM	ASJ	ESO	SAL	ARR	AHB	MPC	RFP	IFS	EFM	JSH	VRG2
2	182-FIP	-0.06	-0.3	0.02	-0.31	-0.29	0.04	0.63	-0.12	0.74	-0.12	0.7	-0.12	-0.06	0.45
3	182-FIP	0.51	-0.21	0.73	0.24	0.05	0.6	0.9	-0.12	1.11	-0.05	0.3	-0.65	0.15	0.72
4	76P	0.4	-0.62	-0.37	-0.28	-0.47	-0.2	-0.39	-0.34	-0.84	-0.27	-0.74	-0.4	-0.6	-0.4
5	A2M	0.27	2.55	0.74	0.59	0.06	0.86	-0.42	0.73	-0.36	1.26	0.23	0.39	0.58	-0.01
6	A2M	0.42	1.95	0.49	0.23	-0.13	0.59	-0.4	0.79	-0.8	1.6	-0.28	-0.12	0.03	-0.2
7	A4GALT	-1.33	0.19	0.33	0.14	0.33	0.4	0.14	0.24	0.51	0.38	0.52	0.36	0.33	0.41
8	AACS	1.42	-0.05	0.04	0.86	0.57	0.14	0.25	0.21	0.24	0.22	0.27	-0.14	1.06	0.15
9	AADAT	0.52	-0.03	-0.46	0.18	0.47	-0.45	0.18	-0.06	0.42	0.3	0.31	-0.46	0.16	0.18
10	AAK1	-1.23	-0.07	0.3	0.07	0.29	0.36	0.37	0	0.38	0.02	0.49	0.25	0.19	0.41
11	AASDHPPT	-0.73	-0.39	-0.43	-0.41	-1.09	-0.64	-1.07	-0.27	-0.32	-0.24	-0.14	-1.18	-0.57	1.14
12	ABCA1	0.45	1.15	1.19	0.49	1.2	1.45	0.36	1.24	0.66	0.71	0.56	1.07	1.33	0.33
13	ABCA1	-0.48	1.15	1.42	0.42	1.14	1.13	0.34	1.11	0.56	0.64	0.7	0.74	1.3	0.31
14	ABCA12	-0.23	-0.19	1.5	-0.29	-0.45	1.3	-0.37	1.13	2.96	0	0.56	-1.05	1.22	-0.7
15	ABCB1	-0.5	1.02	-0.09	0.37	-0.16	0.27	-0.09	-0.18	-0.19	0.47	0.14	0.04	0.56	-0.18
16	ABCB4	-1.13	0.55	0.45	0.15	0.01	-0.03	-0.11	-0.11	0.22	0.03	0.31	0.16	0.53	0.29
17	ABCB6	-0.93	-0.55	-0.46	-0.36	-0.3	-0.3	-0.17	-0.94	0.44	-0.74	0.44	-0.02	-0.04	0.72
18	ABCB7	-0.38	-0.18	-0.4	-0.02	0.26	-0.48	-0.57	-0.61	-0.22	-0.17	-0.39	-0.36	-0.16	0.05
19	ABCC1	0.53	-1.31	-0.69	-1.1	-0.57	-0.6	-1.29	-0.82	-1.11	-0.67	-0.7	-0.35	-1.03	-1.54
20	ABCC12	0.71	-0.03	1.22	1.64	0.39	0.8	1.03	0.86	1.22	0.89	-0.4	-1.23	0.48	0.68
21	ABCC2	-0.38	-1.47	-1.67	-1.6	-1.61	-1.56	-1.74	-1.13	-1.65	-1.22	-1.67	-1.41	-1.65	-2.26
22	ABCC3	-0.42	-0.14	0.2	-0.16	-0.4	0.28	-0.76	-0.42	-0.59	-0.07	-0.76	-0.42	0	-0.59
23	ABCC9	-0.73	2.08	0.29	0.18	0.02	0.54	-0.32	0.57	0.81	0.31	0.37	0.82	0.51	0.1
24	ABCD3	1.56	0.24	-0.21	1.11	1.49	-0.08	2.83	1.2	-0.54	0.89	0.06	-0.34	0.43	0.89
25	ABCE1	1.09	-1.05	0.09	-0.23	-0.16	0.07	-0.27	-0.91	-0.35	-0.63	-0.39	-0.52	-0.53	0.25
26	ABCF2	0.31	-0.35	-1.5	-1.06	-1.21	-1.42	-1.31	-0.46	-0.79	-0.46	-1.36	-0.26	-1.35	-0.97
27	ABCG1	0.57	0.8	-0.21	0.14	0.62	-0.06	0.61	-0.22	0.57	0.92	0.39	0.59	1.02	-0.16
28	ABHD2	-1.68	-0.09	-0.48	-0.28	-0.39	-0.49	-0.45	-0.1	0.31	-0.14	-0.27	-0.12	0.55	-0.63
29	ABI2	0.45	-0.23	-0.97	0.42	-0.15	-0.73	0.97	0.3	-0.24	-0.46	-0.52	-1.11	-0.95	0.56
30	ABI2	0.45	1.28	1.61	0.83	1.25	1.79	0.26	1.22	0.32	0.41	1.63	1.63	0.8	0.64
31	ABI3BP	-1.92	-0.43	0.37	0.4	-0.11	0.17	0.12	0.08	0.7	-0.4	0.83	0.64	0.18	0.23
32	ABL1	-0.15	-0.03	0.12	-0.55	-0.18	0.23	-0.91	-0.05	-1	-0.24	-0.15	-0.63	-0.52	-0.1
33	ABL1	0.68	-0.26	-0.21	-0.26	-0.16	-0.13	-0.53	-0.1	-0.95	-0.19	-0.36	-0.66	-0.8	-0.28
34	ABLIM1	2.5	0.42	1.37	0.7	-1.37	1.19	1.37	0.52	-1.35	0.97	0.54	0.1	0.89	-0.82
35	ABP1	-0.69	-0.69	-0.28	-0.3	-0.32	-0.34	-0.28	-1.03	-0.09	-1.36	-0.35	-0.05	-0.08	-0.2
36	ABTB2	-0.61	-0.56	0.08	-0.46	-0.39	0.05	-0.86	-0.36	-0.43	-0.72	-0.38	0.32	-0.38	0.64
37	ACAA1	2.22	0.4	0.55	0	0.71	0.39	-0.07	0.43	0.26	0.44	-0.15	0.3	0.53	0.3
38	ACAA2	1.09	-0.16	-0.27	-0.44	0.18	-0.26	-0.24	-0.25	-0.36	-0.31	-0.73	0.08	-0.53	-0.76
39	ACADM	-0.6	0.07	-0.08	0.26	0.1	-0.09	1.29	0.5	0.08	-0.13	-0.3	-0.52	0.09	0.26
40	ACADVL	1.37	0.38	-0.48	0.44	0.47	-0.45	0.44	0.86	-0.29	0.53	-0.2	-0.05	0.6	-0.17
41	ACAS2	0.6	-0.06	-0.45	-0.78	-0.71	-0.5	-0.41	-0.67	-1.26	-0.52	-1.22	-0.17	-1.03	-1.11
42	ACDC	0.31	1.15	0.2	2.08	0.81	0.45	0.51	0.4	0.21	0.43	0.2	-0.16	-0.09	0.55
43	ACK1	0.72	0.09	0.45	-0.58	-0.5	0.56	0.18	-0.1	0.01	0.1	-0.68	-0.71	-0.88	-0.2
44	ACO1	0.02	0.03	-1.46	-1.13	-1.03	-1.1	-1.16	-0.52	-1.15	-0.95	-1.44	-1.46	-0.61	-2.31
45	ACOX1	-0.16	-0.37	0.27	0.5	0.06	0.11	-0.39	-0.36	0.56	0.42	1.23	-0.22	0.41	-0.38

Slide from Gema Moreno Bueno, Department of Biochemistry, UAM

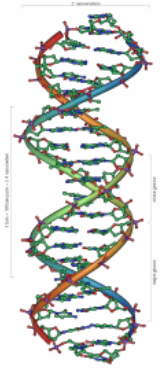
Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

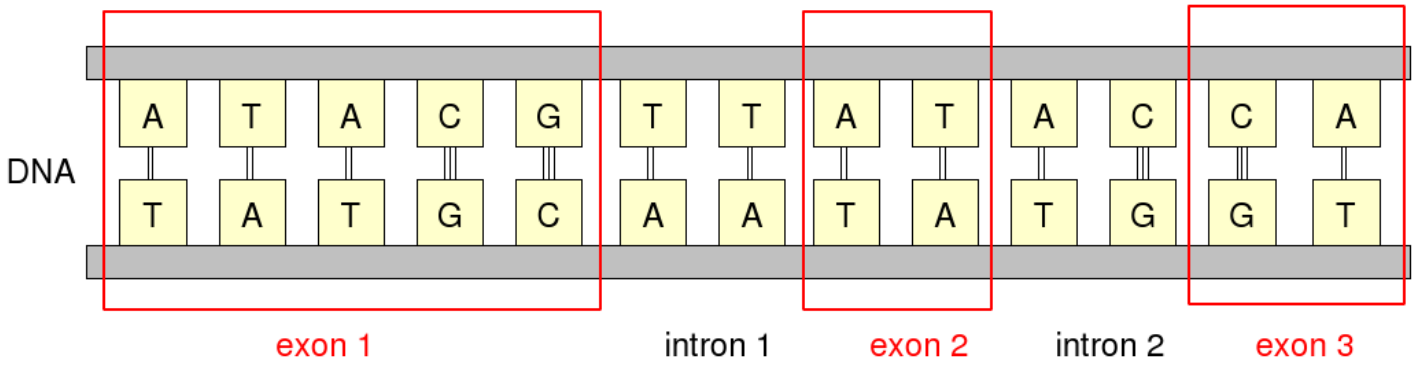
More microarray data

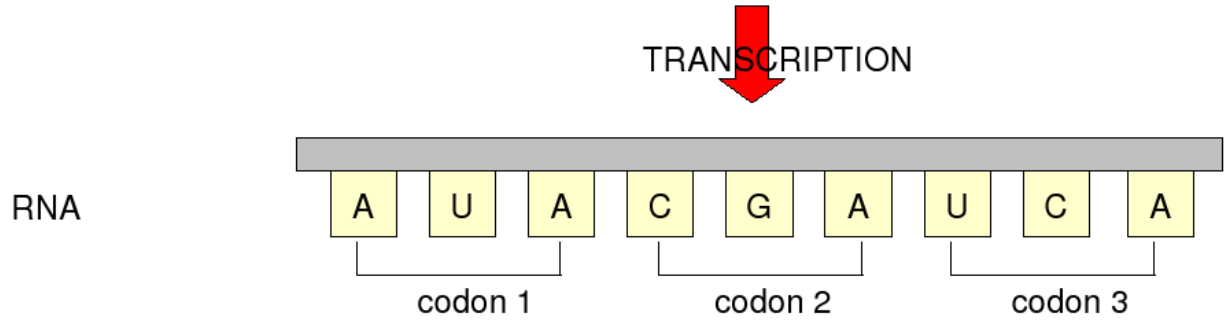


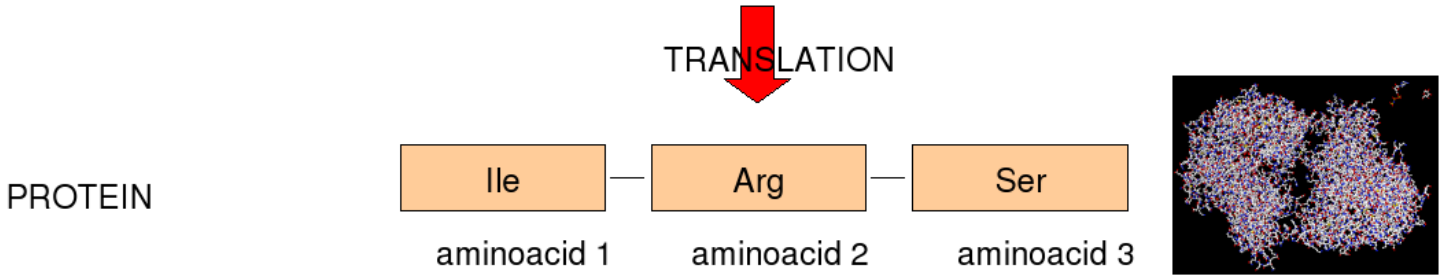
Modified from `http://www2.warwick.ac.uk/fac/sci/moac/students/peter_cock/r/heatmap/scaled_color_key.png`

DNA → protein







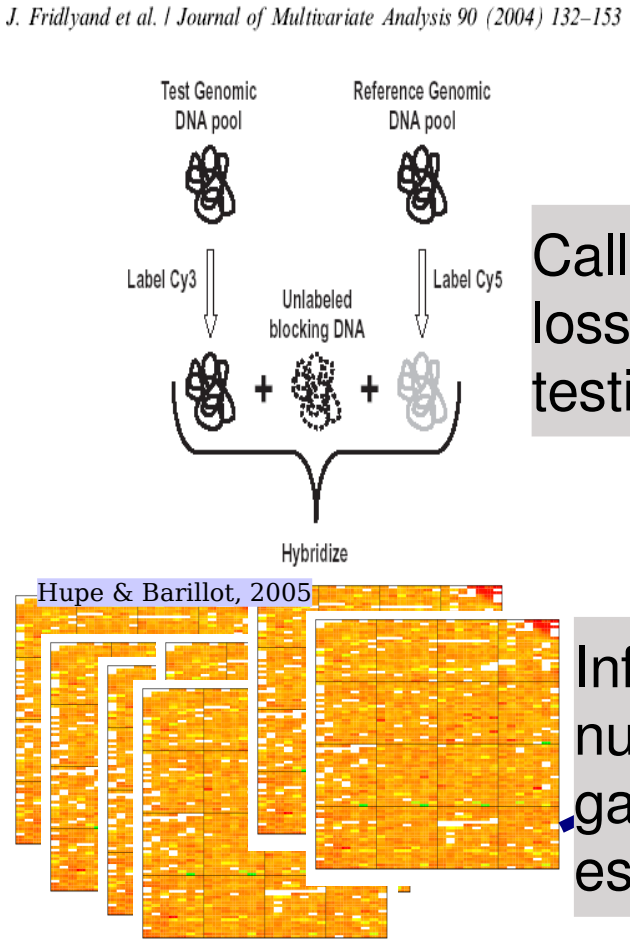


http://en.wikipedia.org/wiki/Image:DNA_Overview.png
http://en.wikipedia.org/wiki/Image:Hexokinase_ball_and_stick_model%2C_with_substrates_to_scale_copy.png

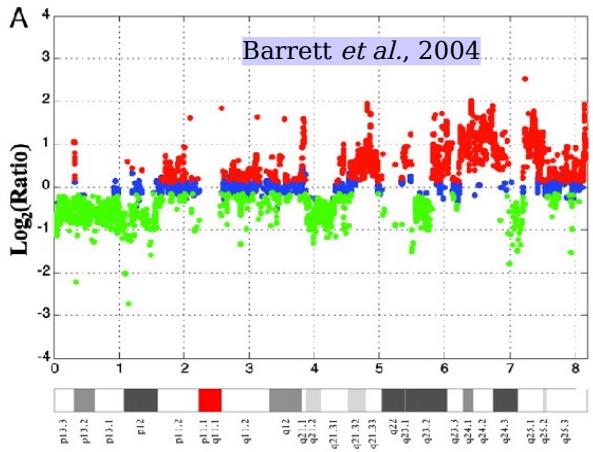
(From O. Rueda's PhD Thesis)

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

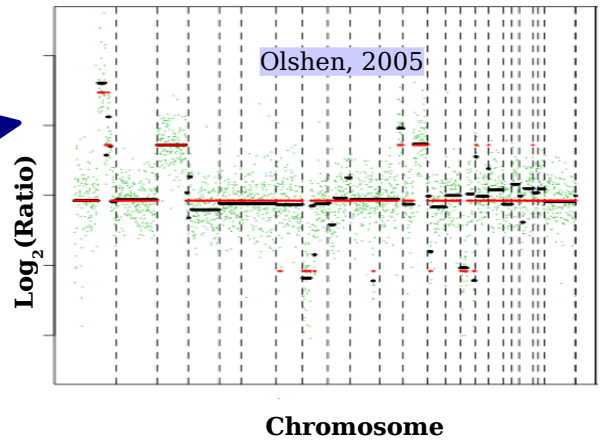
aCGH



Calling gains and losses: hypothesis testing



Inferring number of copy gains/losses: estimation



Arrays: a dot is a DNA fragment. Each array a sample. Each array all chromosomes.

(For analysis, location in chromosome matters)

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Data, data, data (in Gigabytes)

Expression arrays (mRNA) $>$ 40,000 probes

Copy number with aCGH $>$ 400,000 common;
some $>$ 4 x 10^6

... ...

Context
Biological context
Computational context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Multicores and computing clusters

- Increases in CPU speed slowed down (< 20% per year since 2002).
- Increase in the number of “cores”: 2, 4, 8. Next 10 years?
- Inexpensive computing clusters with off-the-shelf components.
- Must design our programs from the start: parallel programming

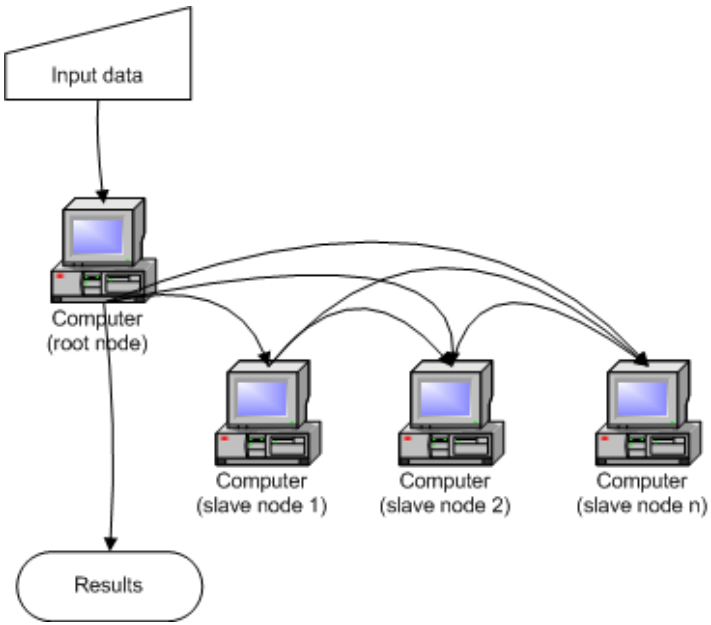


Image from `http://faq.distributed.net/`

Context

Parallelizing code

Web applications

Large data sets and parallelization

R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Context

Parallelizing code

Web applications

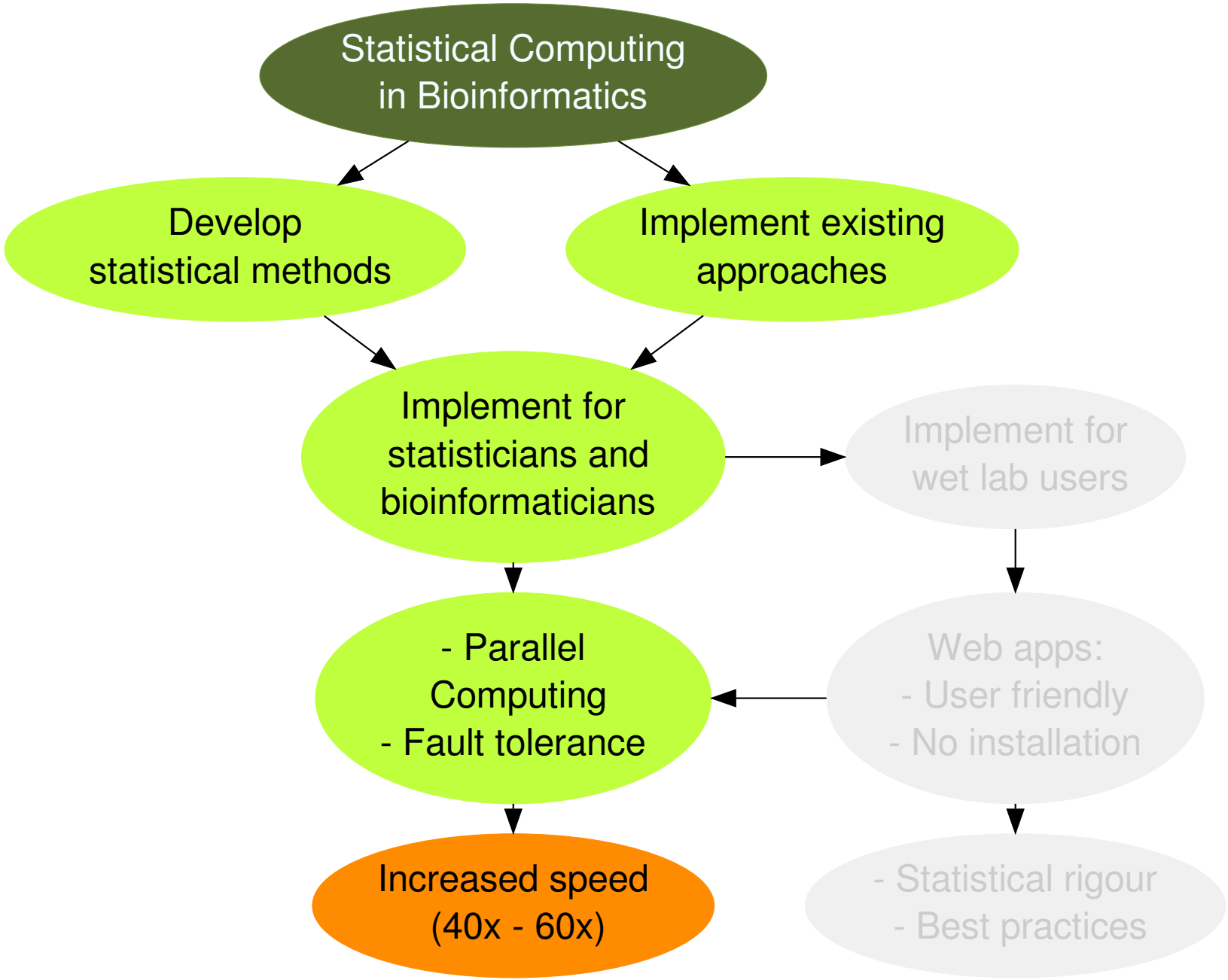
Large data sets and parallelization

R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Standalone


Statistical Computing in Bioinformatics
Develop statistical methods
Implement existing approaches
Implement for statisticians and bioinformaticians
Implement for wet lab users
- Parallel Computing
- Fault tolerance
Web apps:
- User friendly
- No installation
Increased speed (40x - 60x)
- Statistical rigour
- Best practices

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

R code

- Code available for many procedures (but a few years ago none parallelized!)
- Many computations embarrassingly parallelizable:
  - bootstrapping and cross-validation
  - arrays (or samples)
  - arrays by chromosomes
  - parallel chains in MCMC
- Figure production can be parallelized

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Parallelizing R code

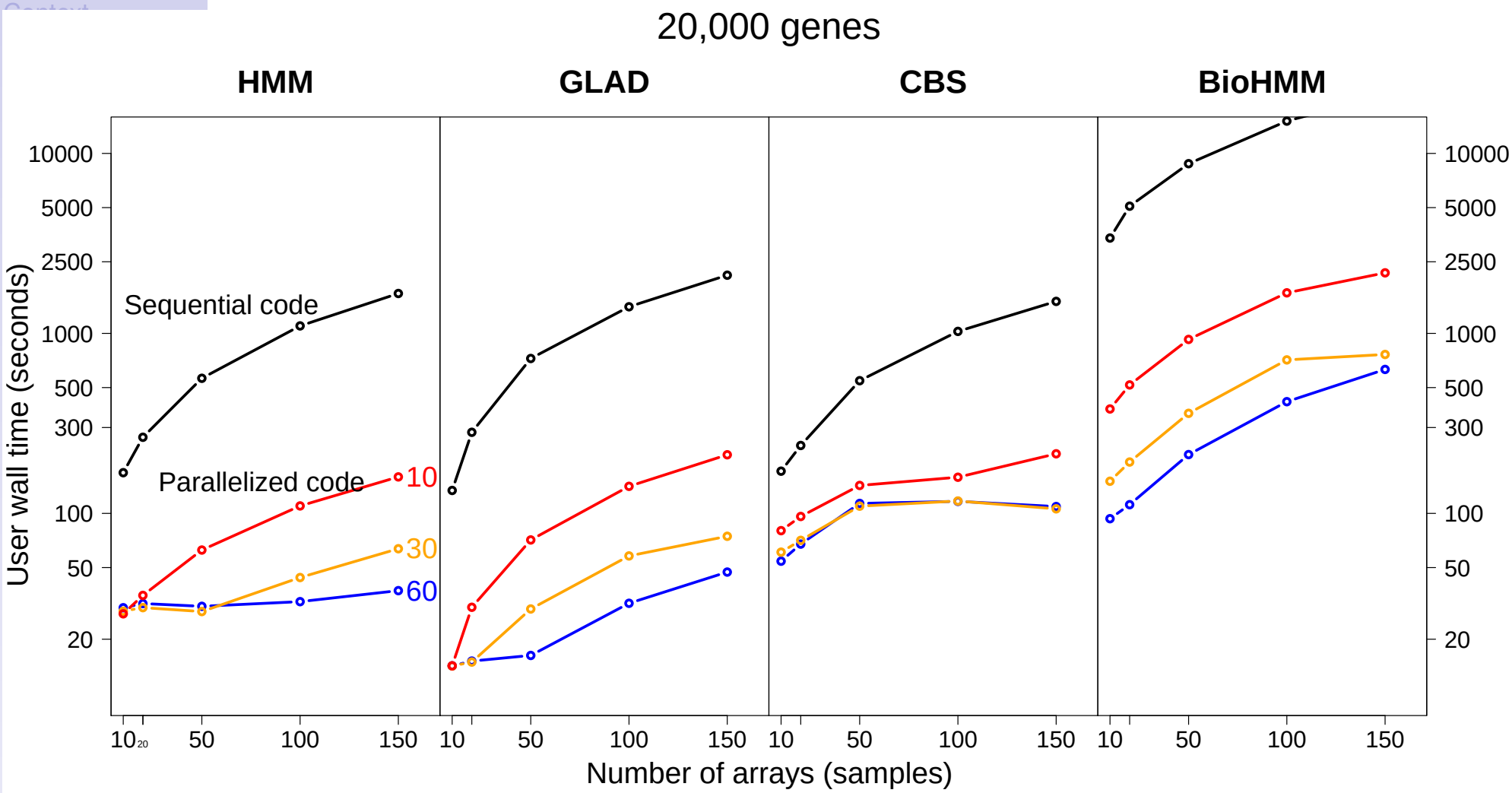
- (Implement missing functionality: R/C)
- MPI: R packages Rmpi, papply, snow, snowfall
- Load balanced
- Wrappers over “mid level” functions in package: ease updating
- Parallelize:
  - Bootstrap samples/Cross-val. runs.
  - arrays
  - arrays by chromosomes
  - (or a combination of both)

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Is it worth it?

- Are speed improvements really worth the effort?
- Over what range of problems do see improvements?
- With what hardware can we see improvements?

What do we gain?



Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

What do we gain?

Are speed improvements really worth the effort?

Your effort: “R CMD INSTALL ADaCGH2”.

Over what range of problems do see improvements?

10 to 10^3 arrays/samples;
10^4 to 10^6 spots/genes.

With what hardware can we see improvements?

2 cores to 120 cores.

- Smaller clusters: more cost effective
- Single node/multi-core: less communication overhead

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Where is this running?

- *varSelRF* (CRAN)
- *ADaCGH2* (BioConductor)
- *SignS* (launchpad: http://launchpad.net/signs)

Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Context
Parallelizing code

Web applications

Web apps: how

Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Applications for wet lab researchers

- Analyze data in a reasonably short time.
- User friendly access to methods that are statistically rigorous.

Web-based applications

- User-friendly interface.
- No hardware/software hassles for end users.
- Parallelization is transparent.
- Method selection can be partially transferred (to us).
- Short user wall time: use (hardware/software) resources rarely available to individual biomedical researchers
- Just type in a URL:
  `http://www.some-application`

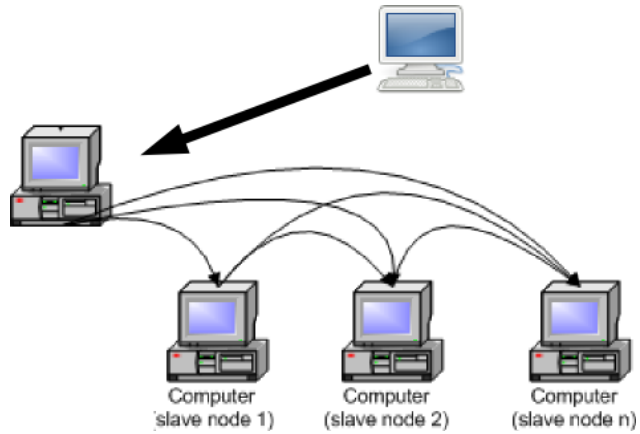
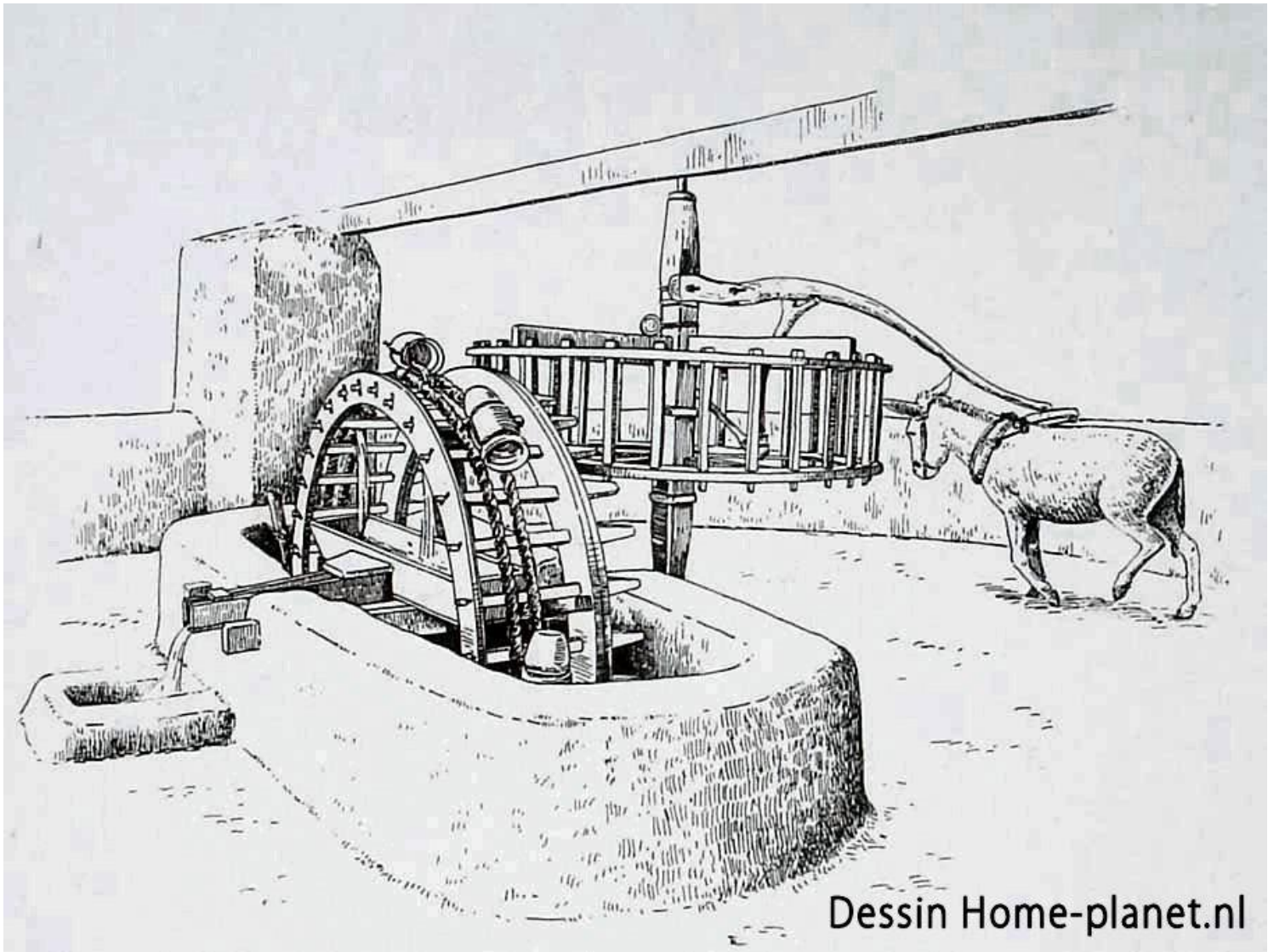


Image modified from `http://faq.distributed.net/`

Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

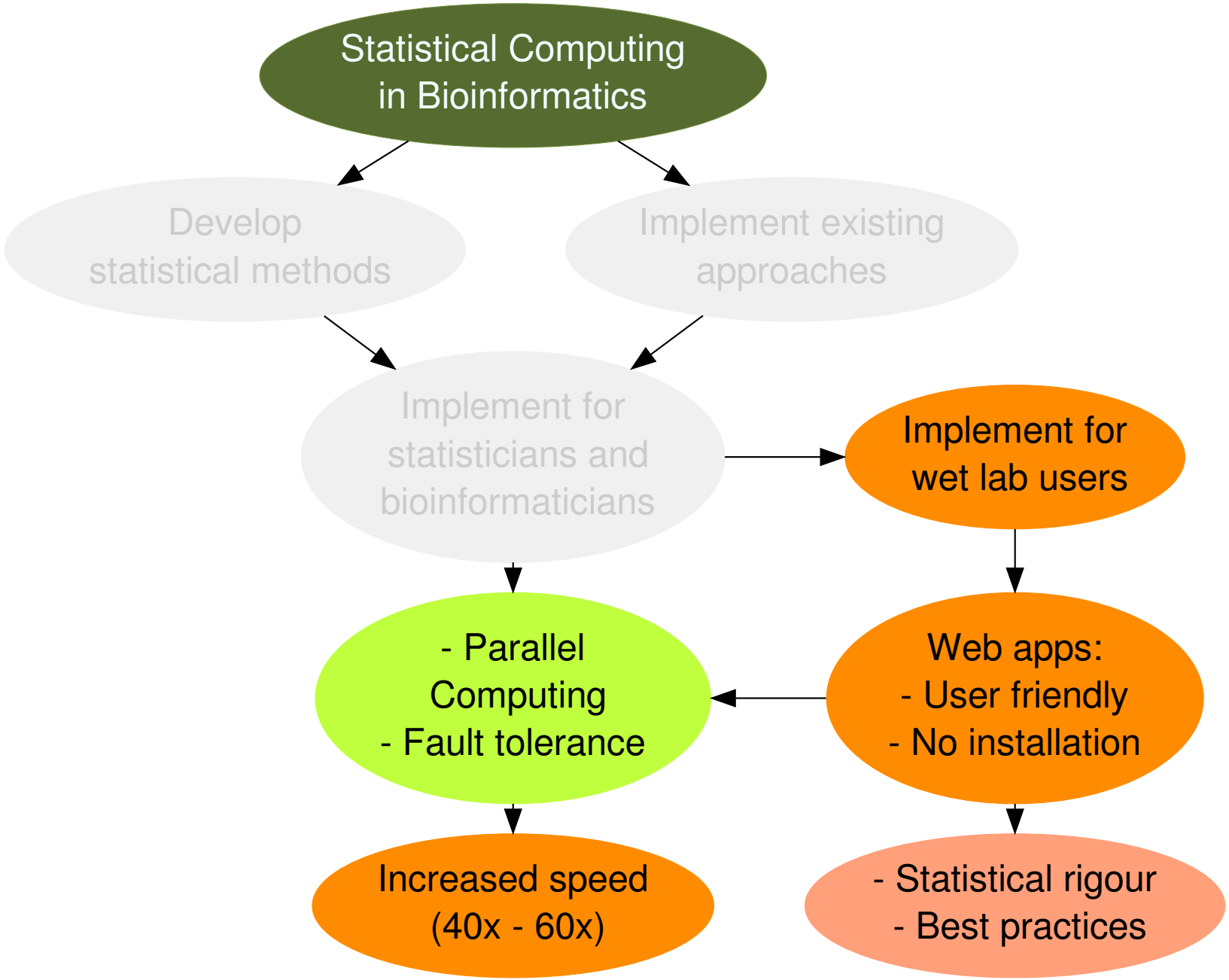
Sometimes collaborations feel like . . .



(From http://www.bitacoradegalileo.com/2010/11/16/giordano-bruno-en-la-cara-oculta-de-la-luna/)

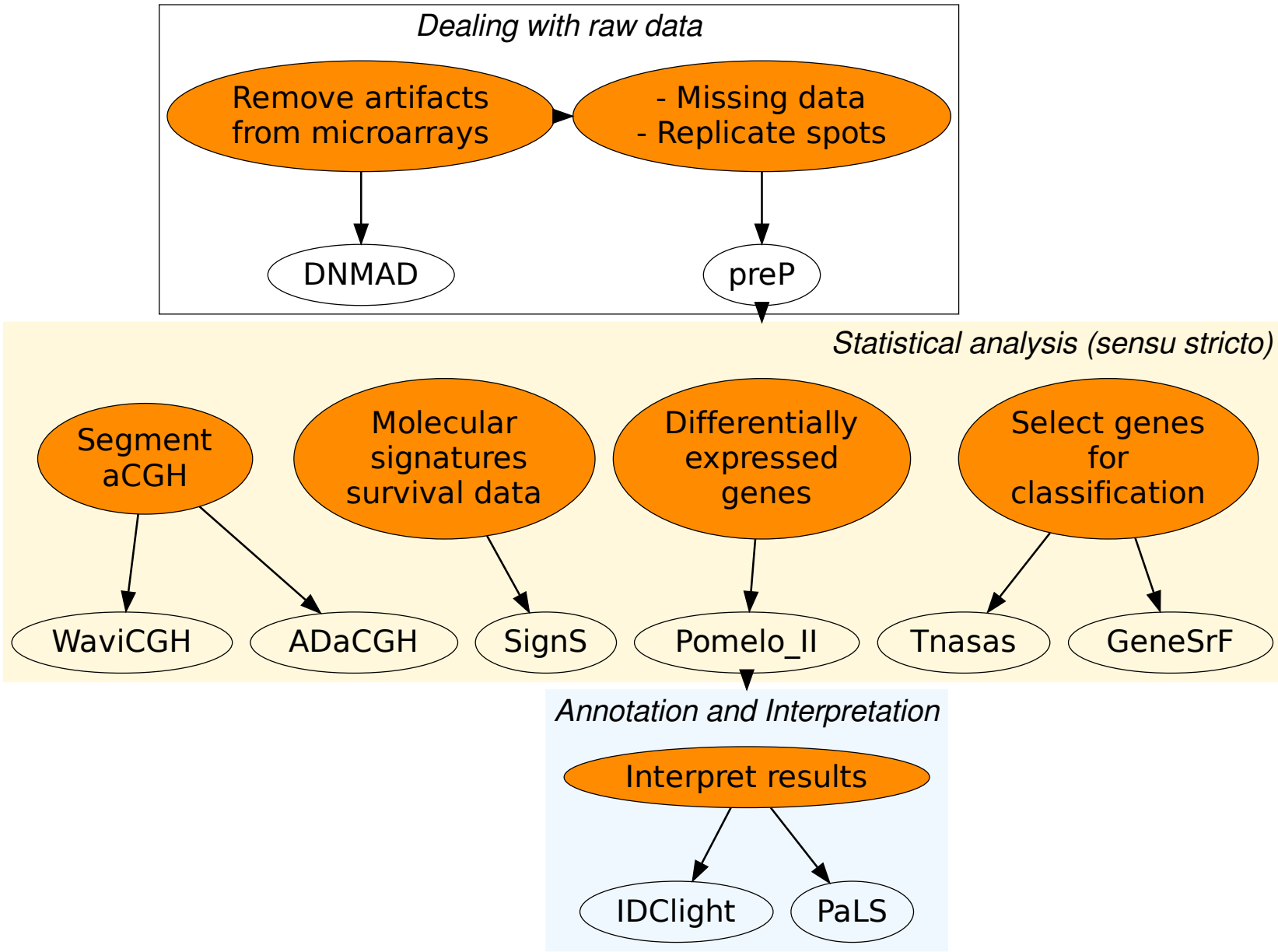
Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Parallelization in web applications



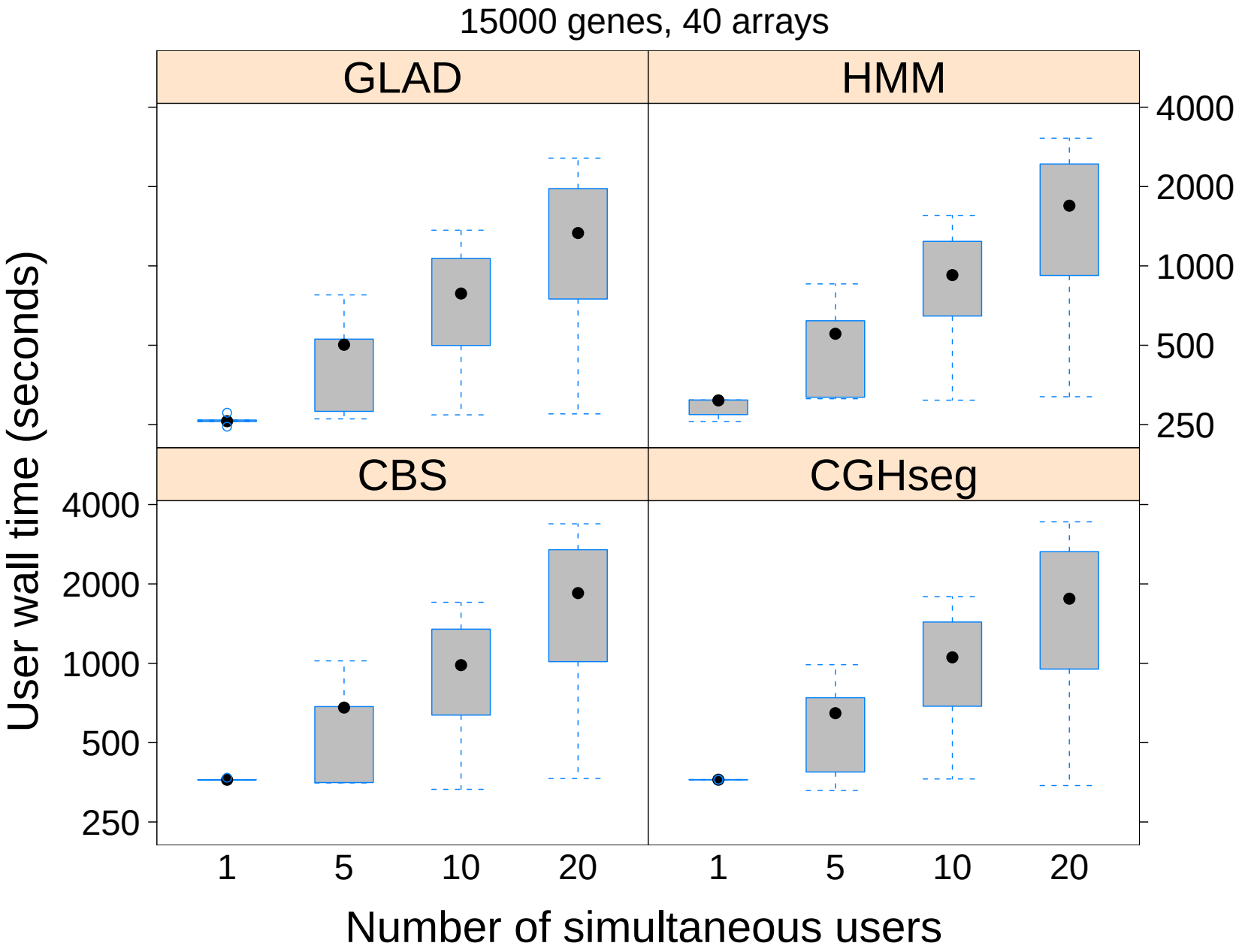
Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Main web-based applications



Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

What do we gain?



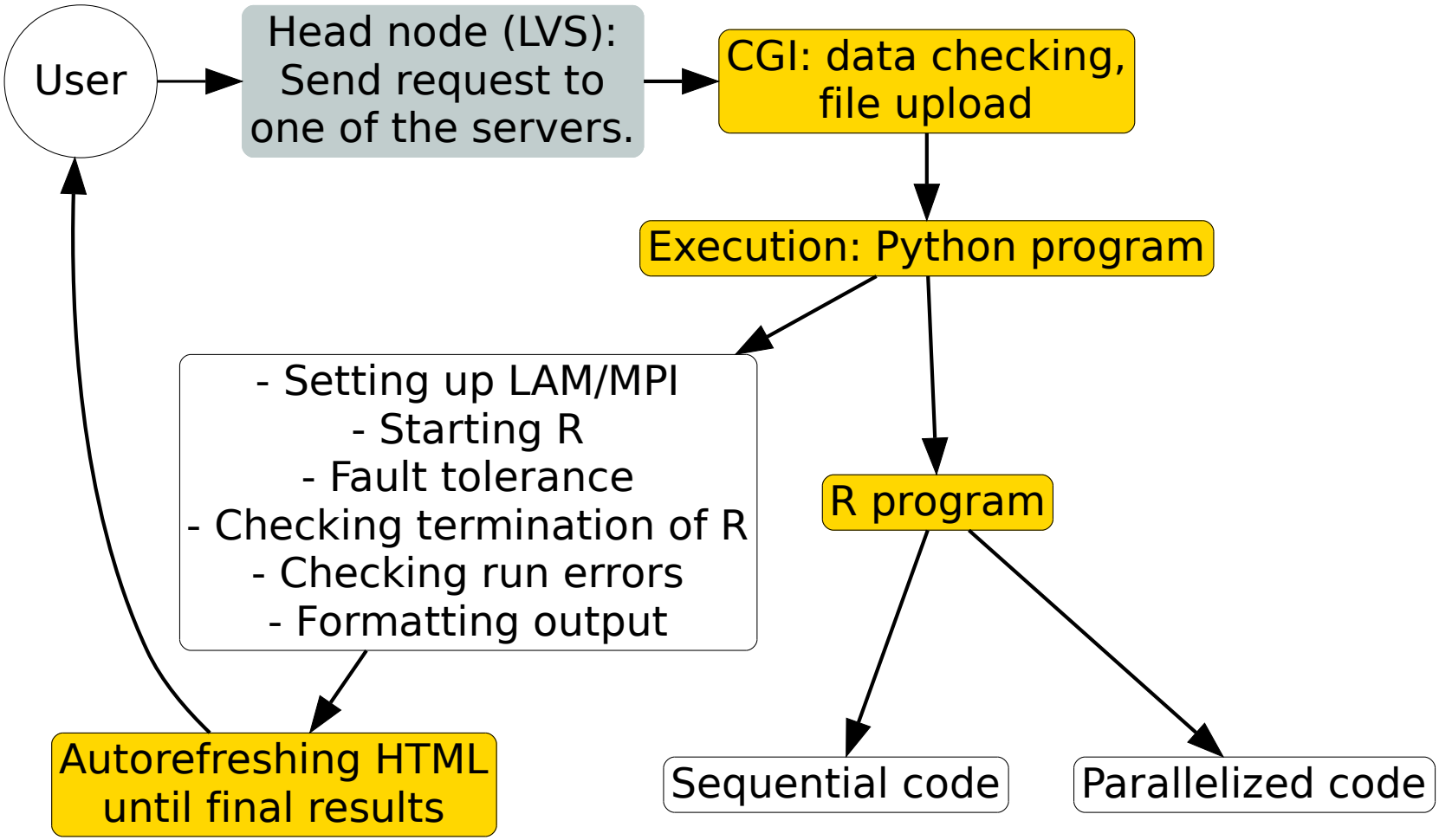
Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

How it works: some key ideas

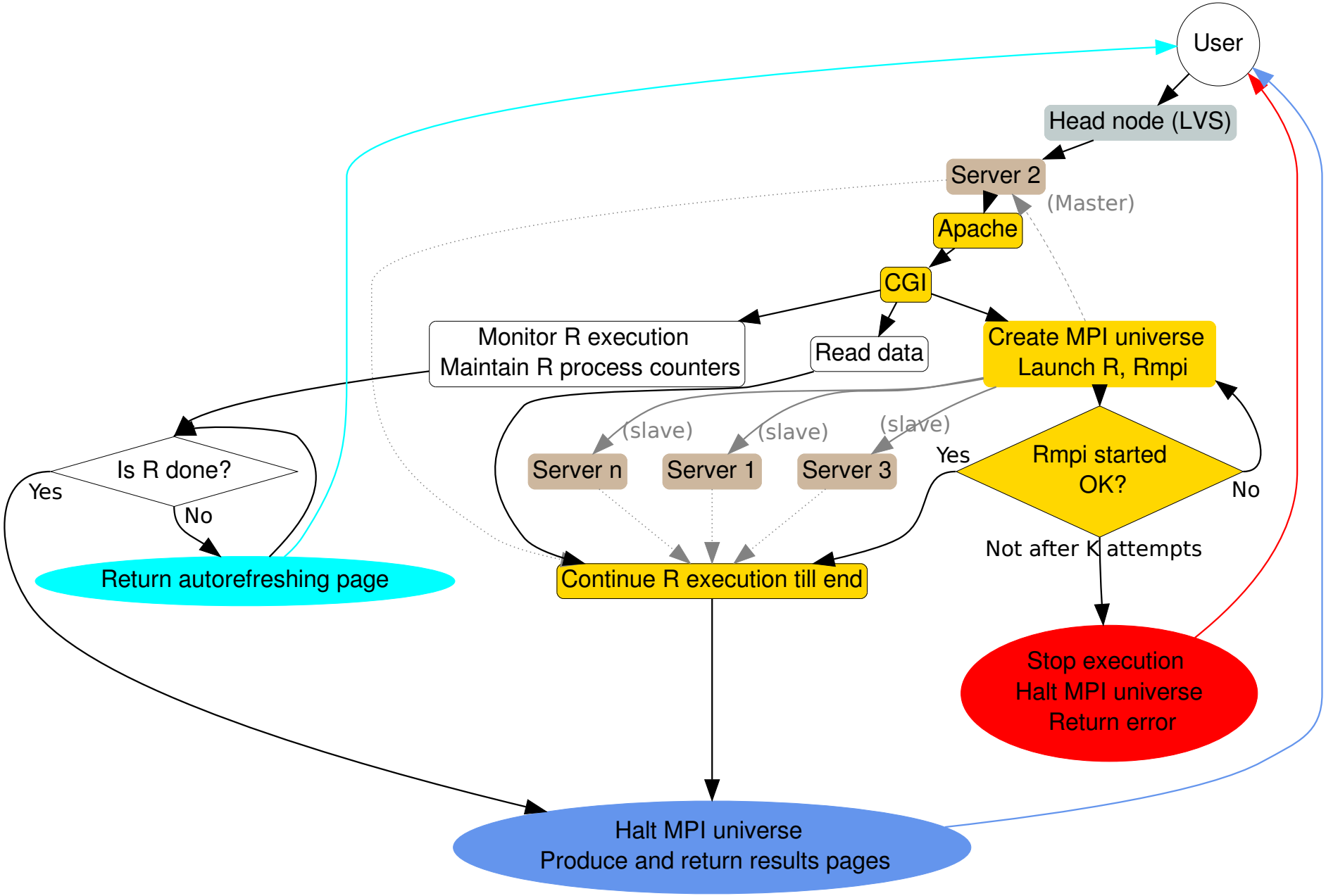
- Each run
  - Parallelization (transparent for users)
  - Fault-tolerance (network problems, machine crashes, bugs)
    - Check-pointing
- Periodic tasks (keep system running 24h, 365 d)
  - Automatic monitorization
  - Automated testing suite

Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

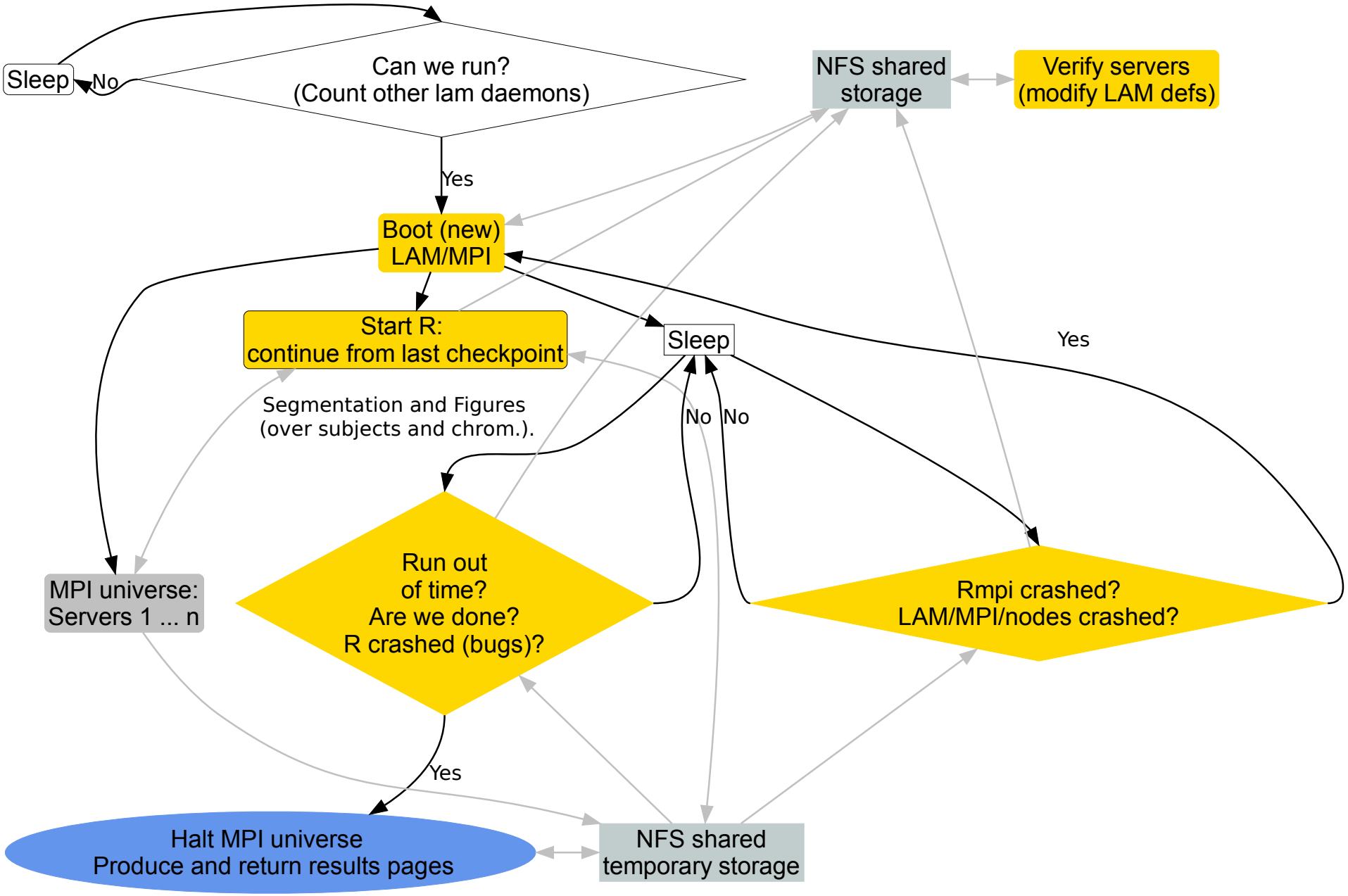
What happens



What happens: details



MPI details


Sleep
No
Can we run?
(Count other lam daemons)
Yes
NFS shared storage
Verify servers (modify LAM defs)
Boot (new) LAM/MPI
Start R: continue from last checkpoint
Sleep
Yes
Segmentation and Figures
(over subjects and chrom.).
No
No
MPI universe: Servers 1 ... n
Run out of time? Are we done? R crashed (bugs)?
Rmpi crashed? LAM/MPI/nodes crashed?
Yes
Halt MPI universe
Produce and return results pages
NFS shared temporary storage

Context
Parallelizing code
Web applications
Web apps: how
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Where is this running?

- http://signs.bioinfo.cnio.es
- http://wavi.bioinfo.cnio.es
- http://genesrf.bioinfo.cnio.es

Context

Parallelizing code

Web applications

Large data sets and parallelization

R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Context

Parallelizing code

Web applications

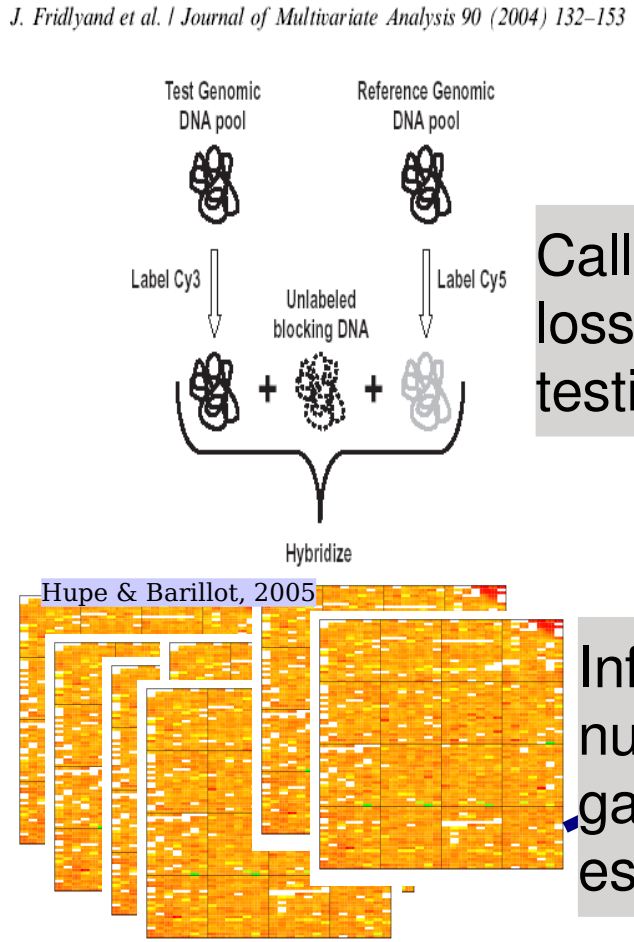
Large data sets and parallelization

R, C, and compression on the fly

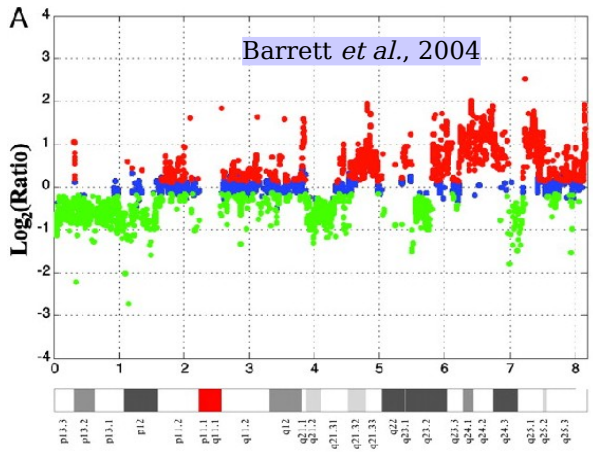
Conclusions, future, et al.

Appendix

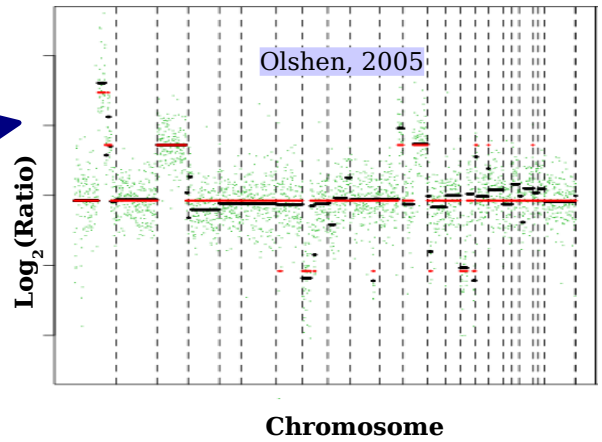
aCGH



Calling gains and losses: hypothesis testing



Inferring number of copy gains/losses: estimation



Arrays: a dot is a DNA fragment. Each array a sample. Each array all chromosomes.

(For analysis, location in chromosome matters)

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Large data sets

- Millions of spots
- Hundreds or thousands of subjects.
- No need to hold everything in RAM at once.

- Package `ff`: "memory-efficient storage of large data on disk and fast access functions."

- Combined with:
  - parallelization
  - shared storage

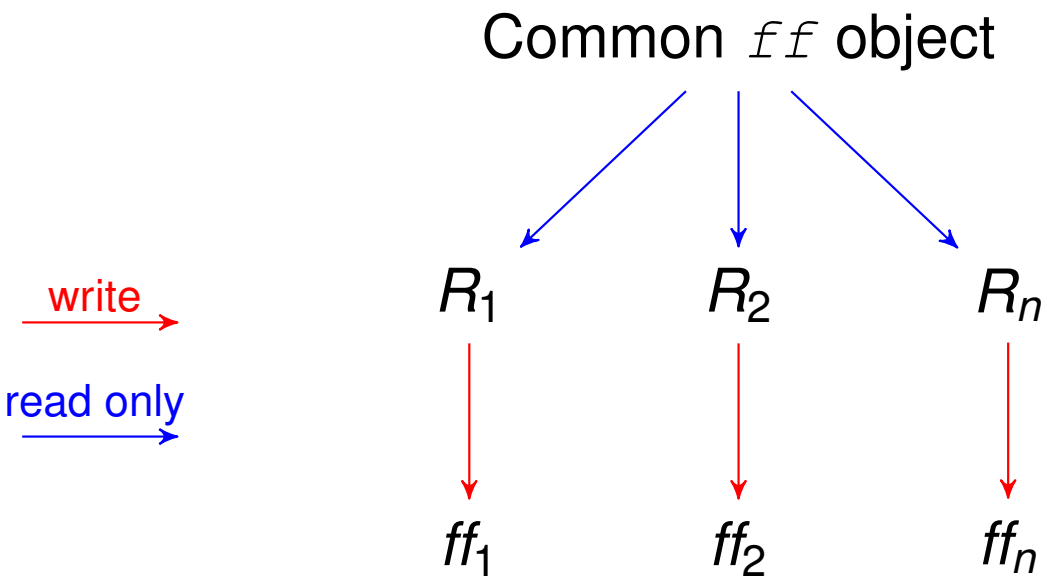
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

ff and parallelization

- *ff* stores the object on disk.
- Read that object from various R processes.
- Different R processes can write in different *ff* objects

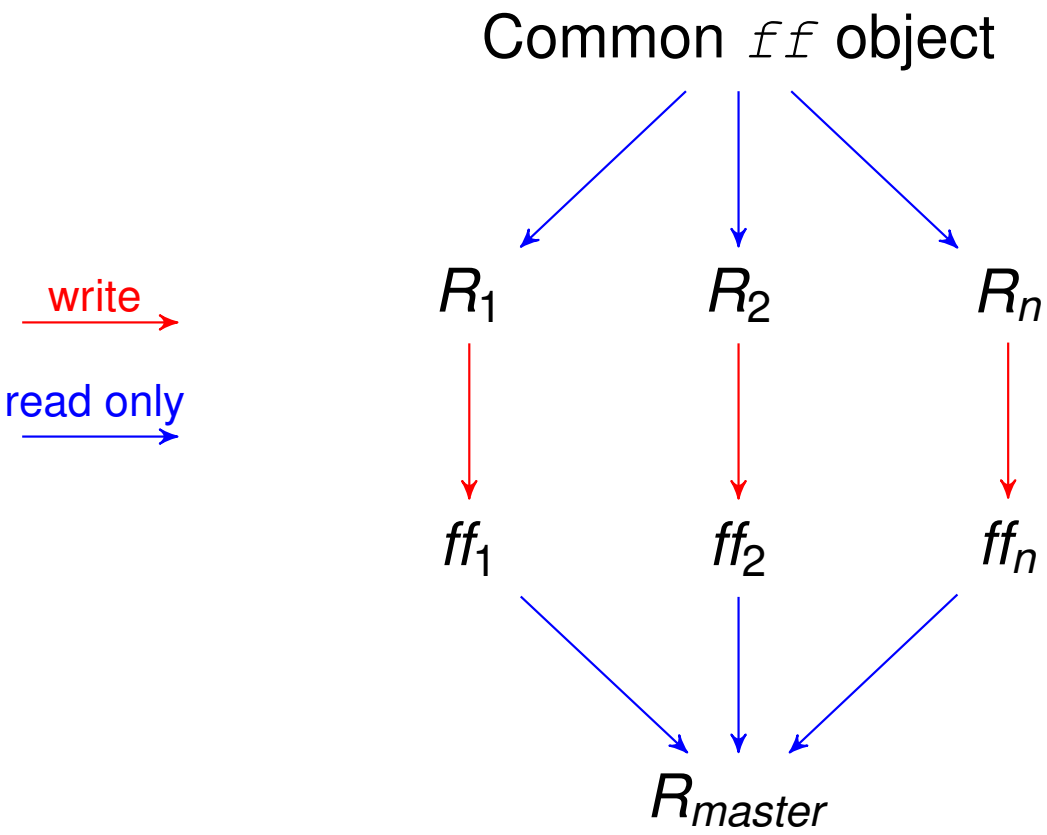
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

ff and parallelization (I)

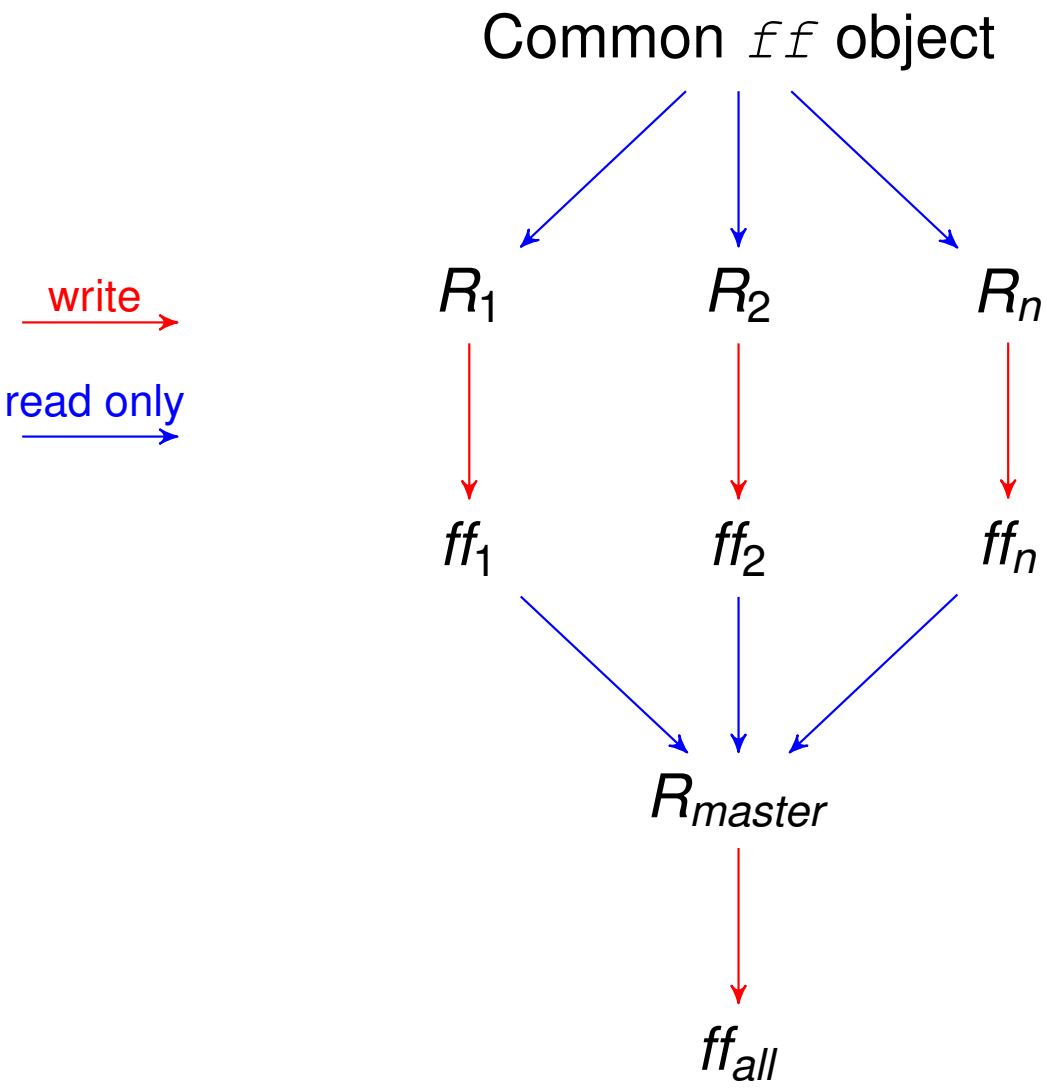


Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

`ff` and parallelization (I)

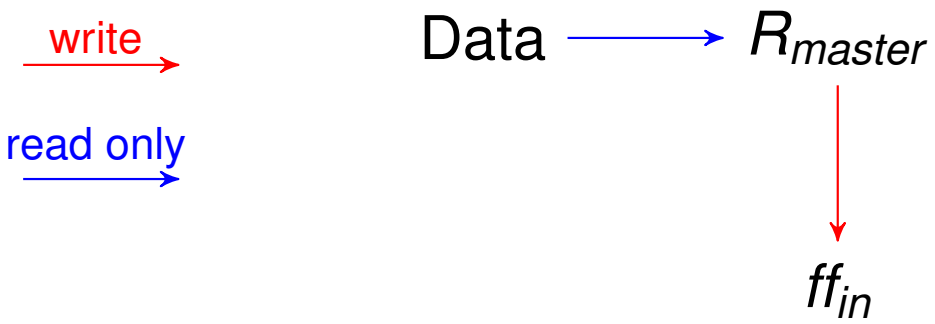


ff and parallelization (I)



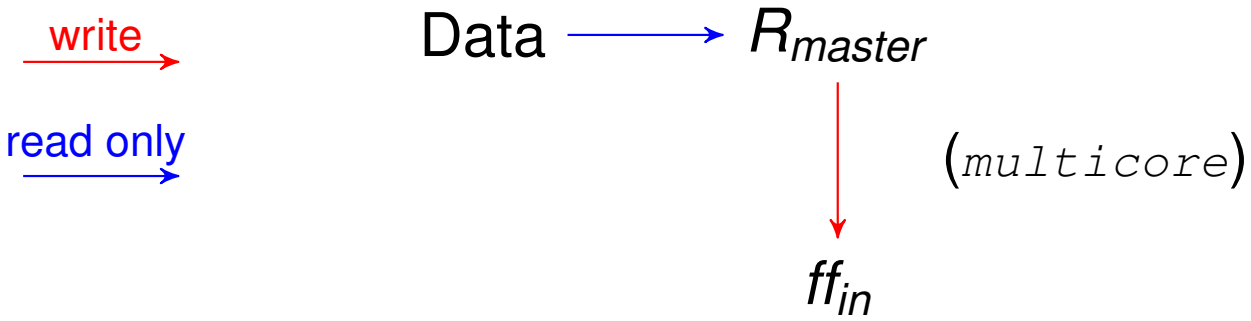
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

`ff` and parallelization (II)



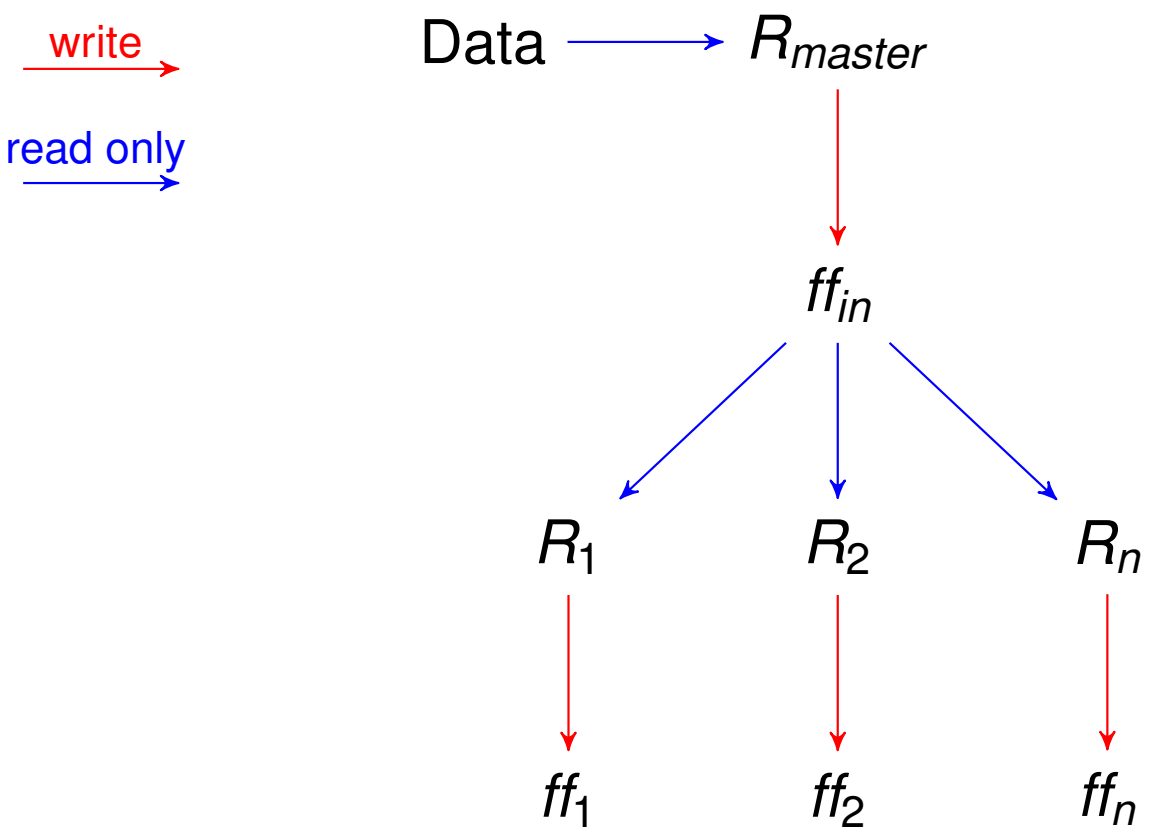
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

ff and parallelization (II)



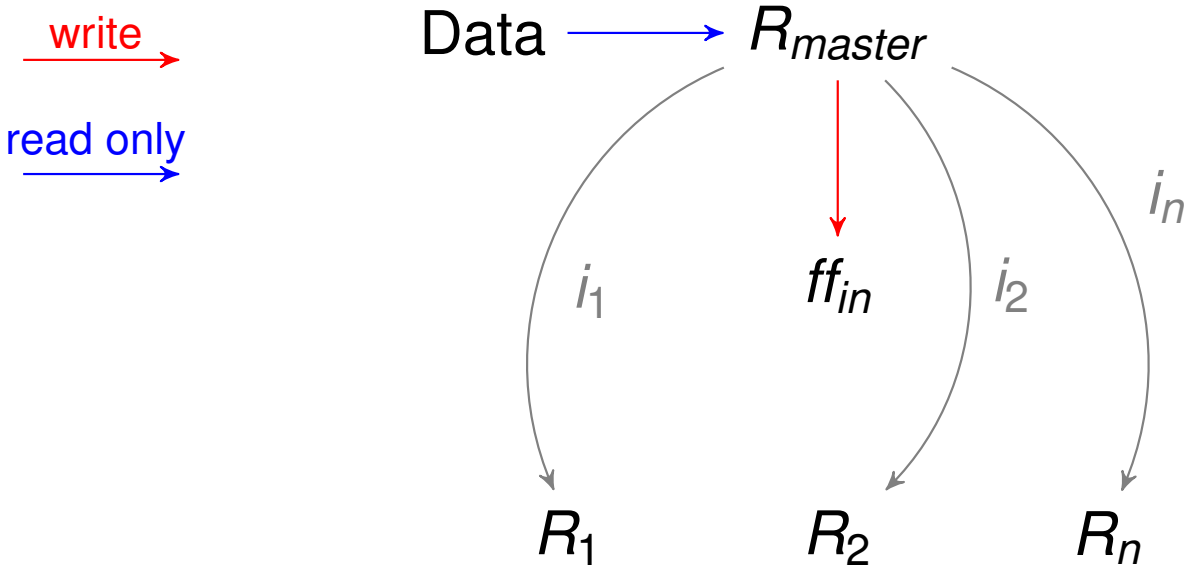
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

`ff` and parallelization (II)

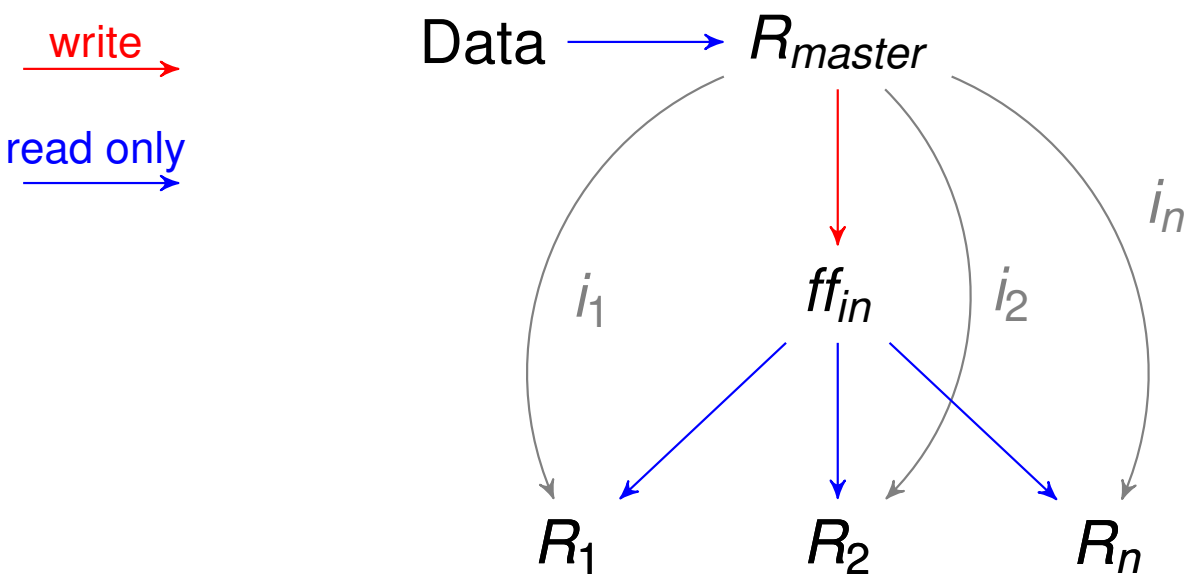


Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

`ff` and parallelization (II)

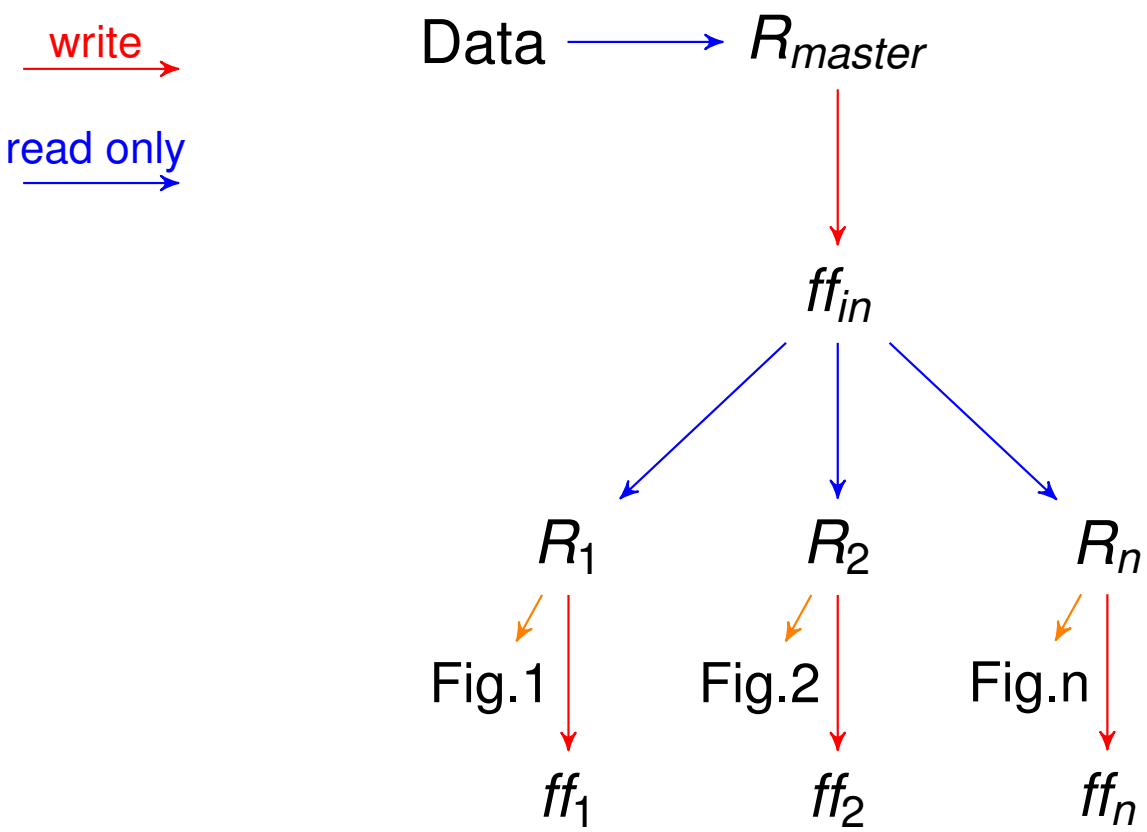


`ff` and parallelization (II)



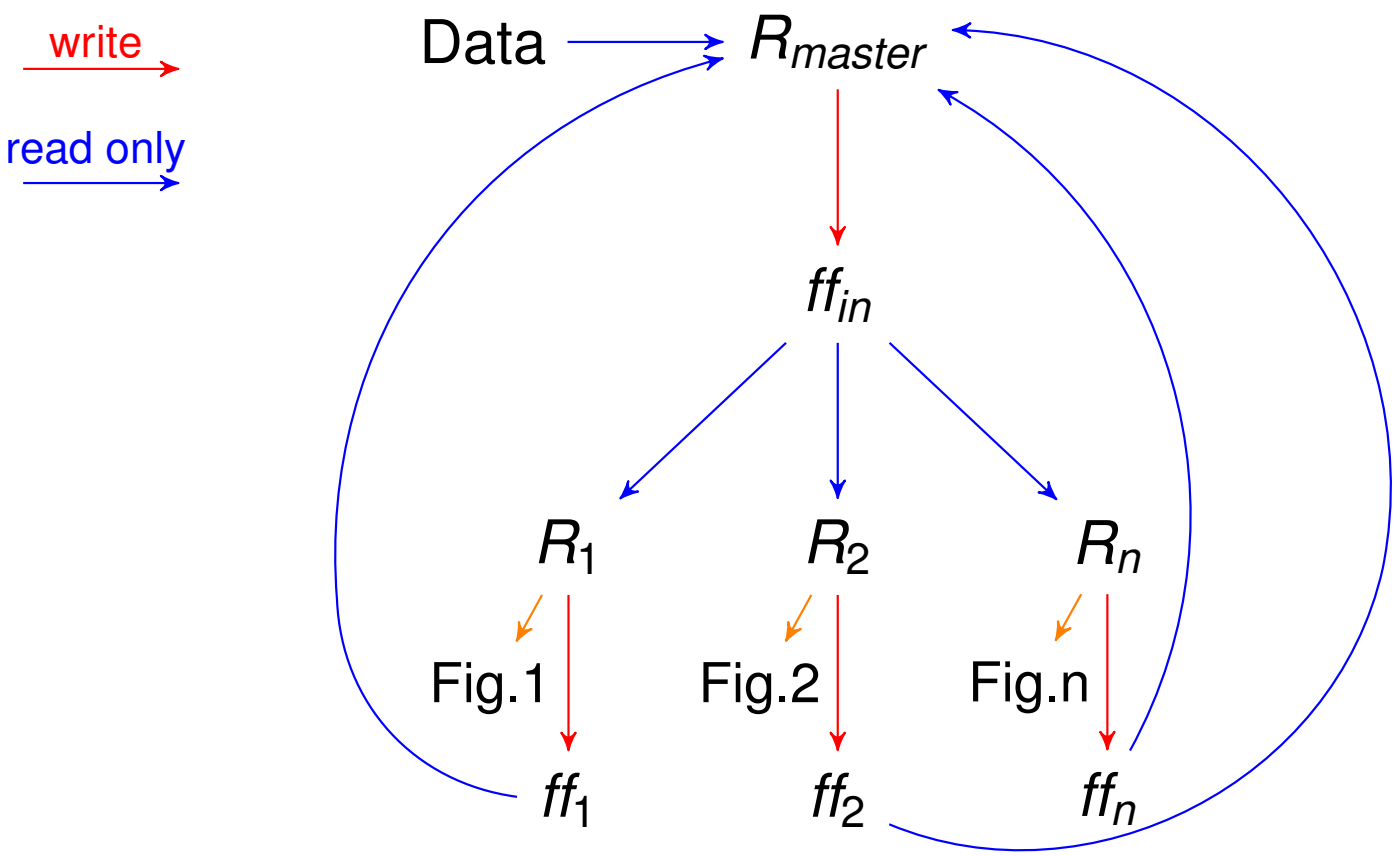
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

`ff` and parallelization (II)

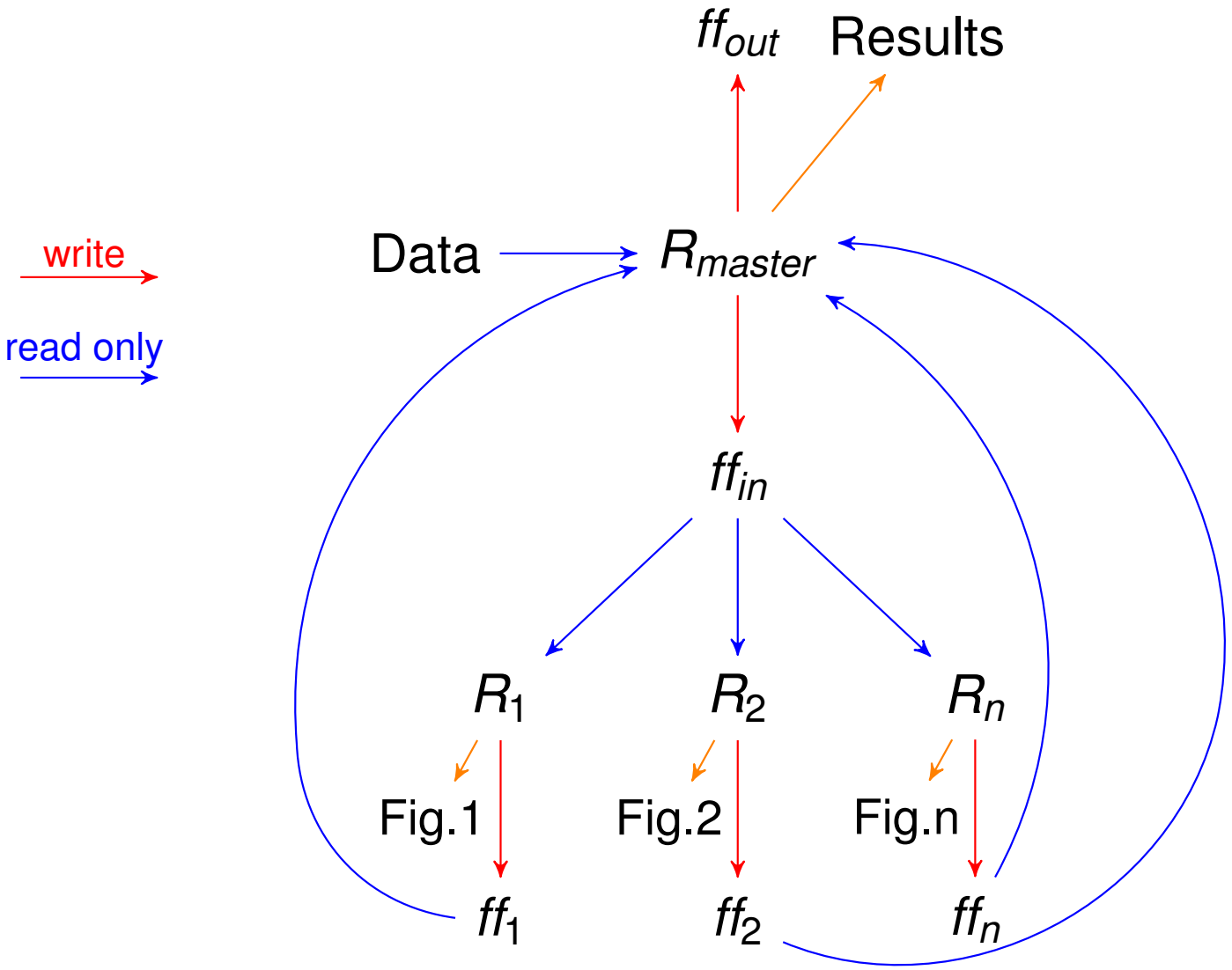


Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

ff and parallelization (II)



ff and parallelization (II)



Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Where is this running?

- *ADaCGH2* (BioConductor package)
- Web-based application http://wavi.bioinfo.cnio.es.

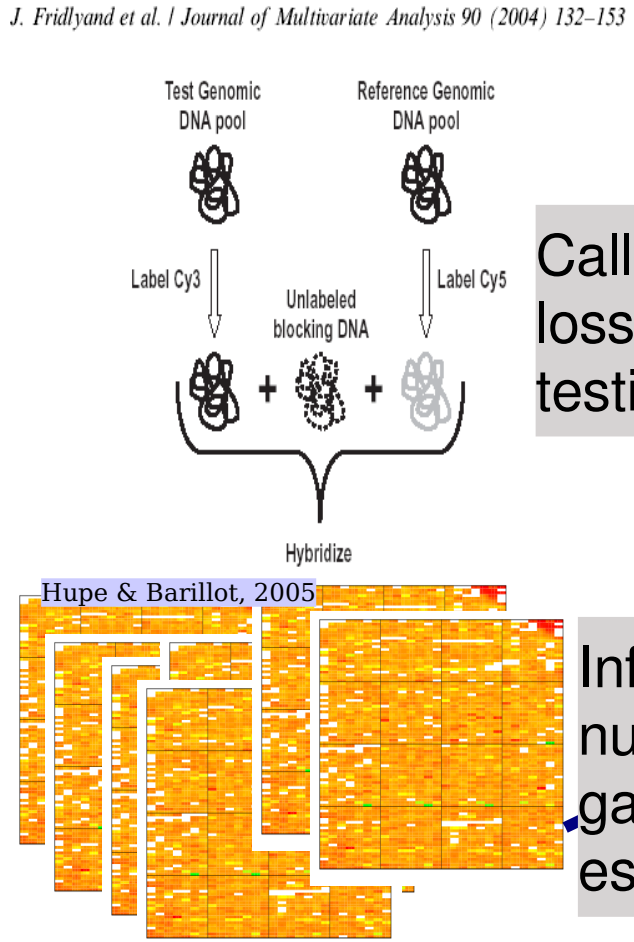
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Context
Parallelizing code
Web applications
Large data sets and parallelization

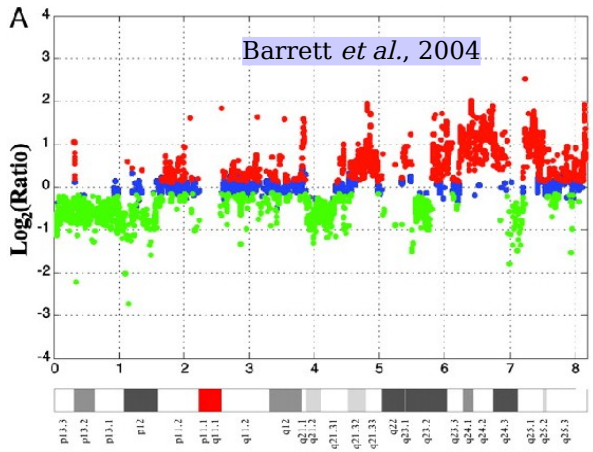
R, C, and compression on the fly

Conclusions, future, et al.
Appendix

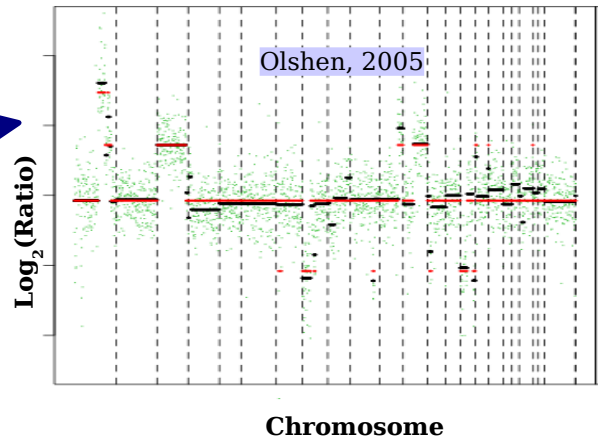
aCGH



Calling gains and losses: hypothesis testing



Inferring number of copy gains/losses: estimation



Arrays: a dot is a DNA fragment. Each array a sample. Each array all chromosomes.

(For analysis, location in chromosome matters)

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Store and access (large) pre-computed results

- HMM for aCGH data with Reversible Jump: Viterbi
- Common regions: “count” on the Viterbi paths.

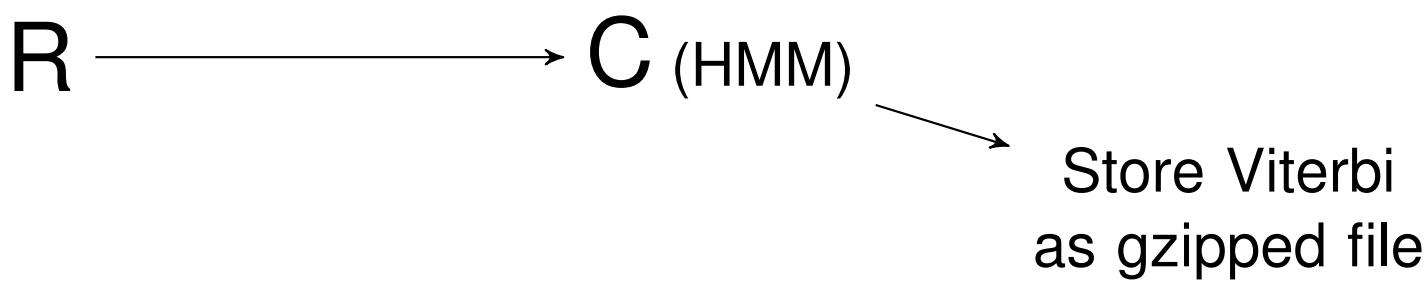
- Fitting HMM/common regions: distinct operations.

- C: number-crunching.
- R: wrapper and figures/tables.
- C: creates large amounts of data.

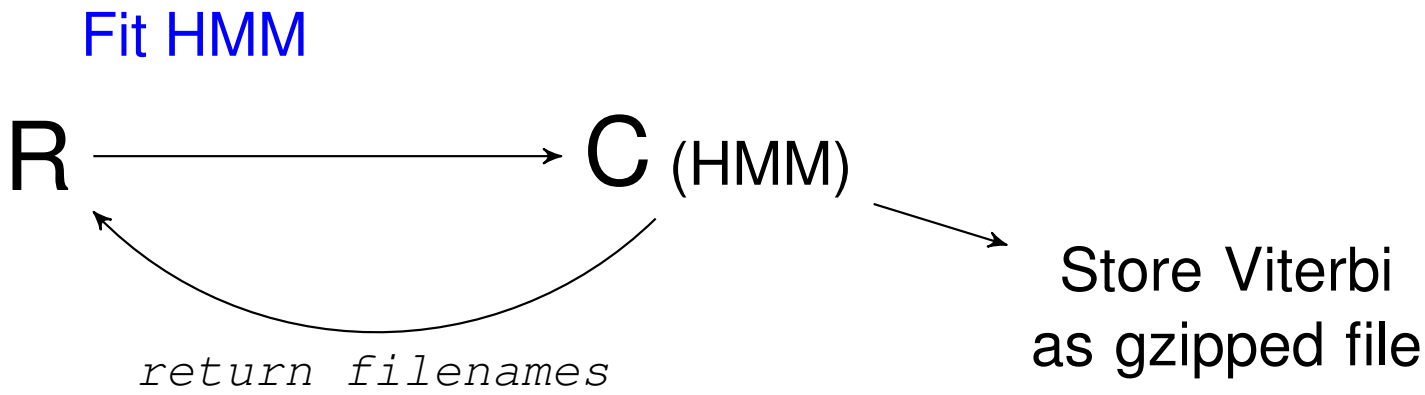
- In package *RJaCGH* (CRAN).

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Fit HMM

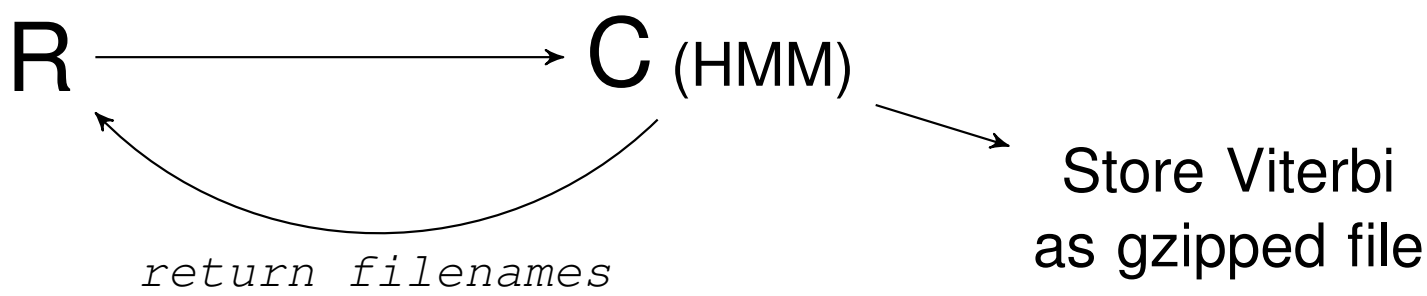


Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

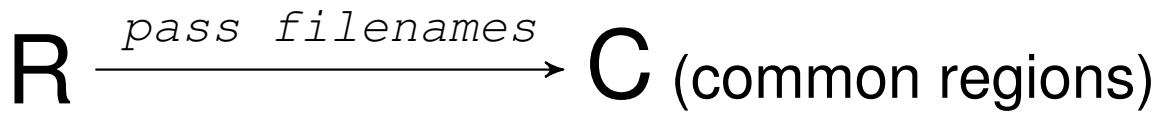


Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

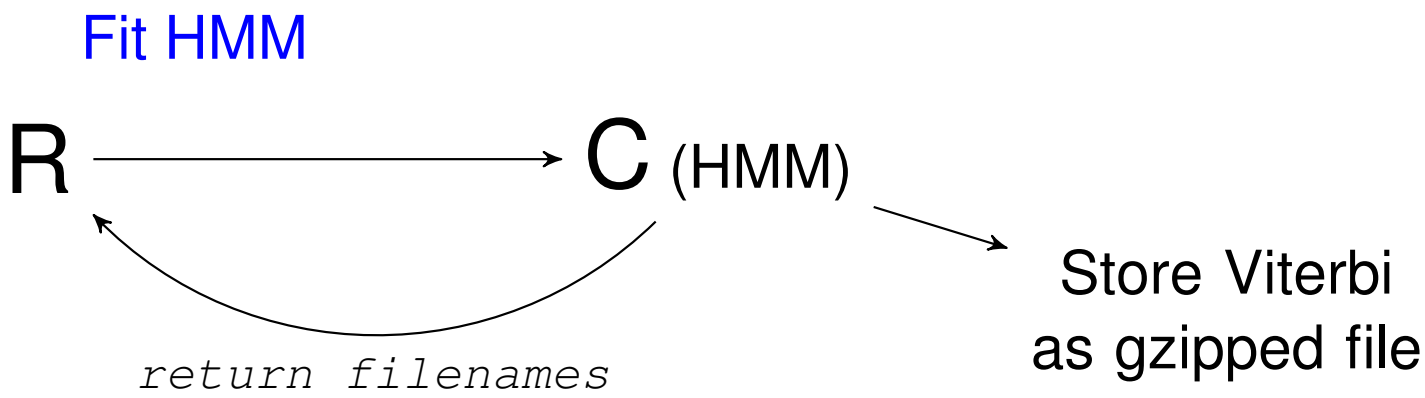
Fit HMM

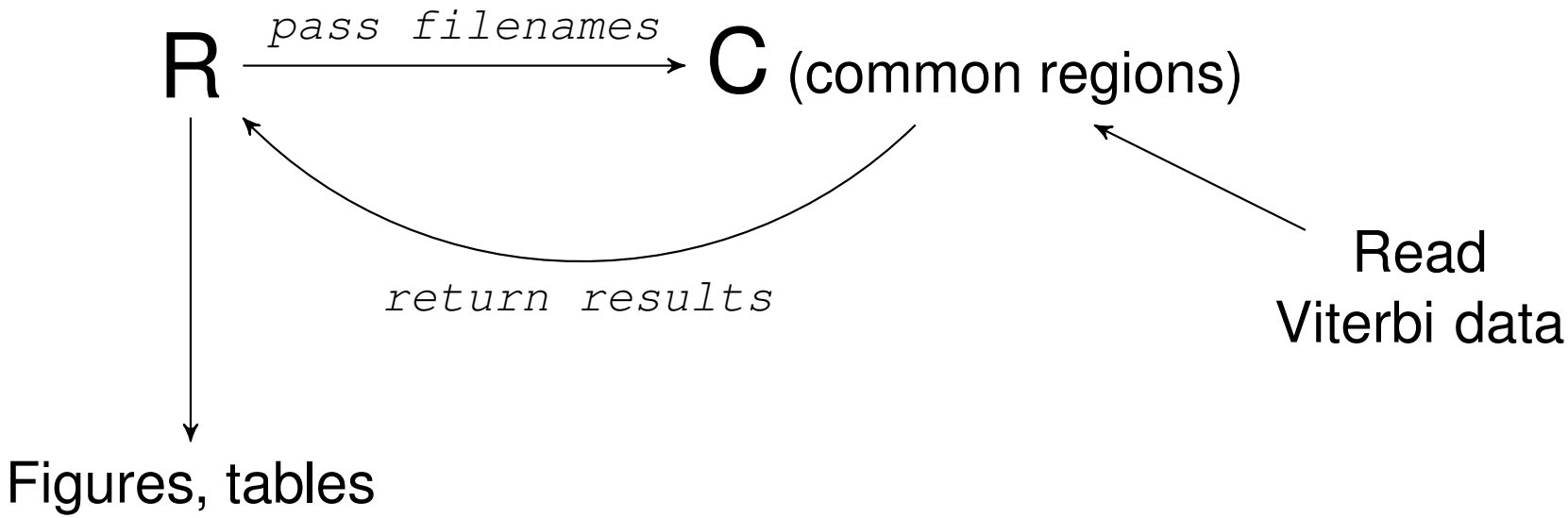


Find common regions



Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix





Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly

Conclusions, future, et al.

Appendix

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

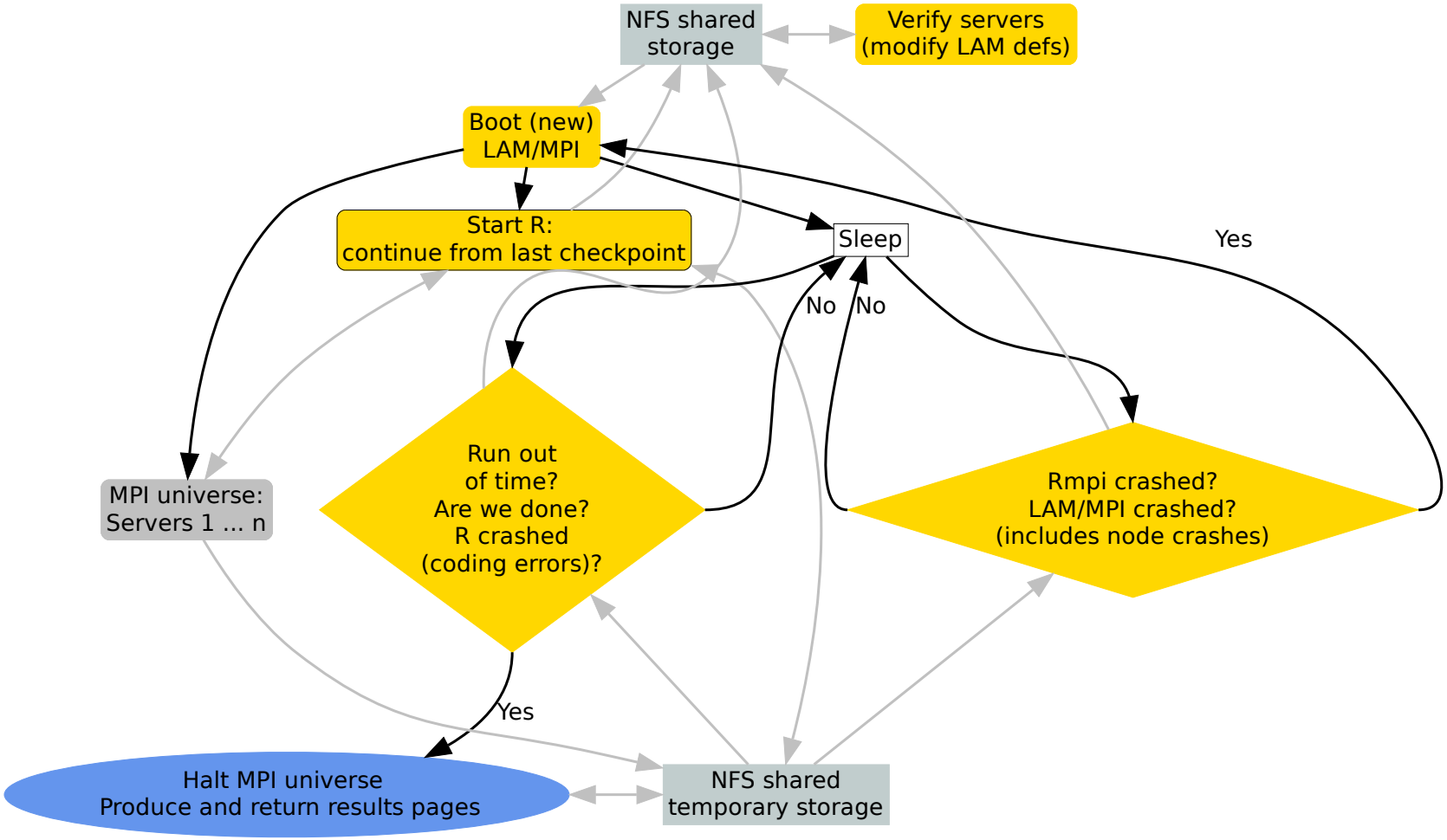
Web-based: A few things we've learned

- Configuration sucks (if you need to modify $>$ 1 file)
- Too many languages
- Adding test cases to the testing suites: web, R
- Documentation: in the code, web pages, LaTeX ...
- Too much R code to catch errors
- User interfaces: who designs them?

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Fault tolerance and communication

- Manual check for errors (R ain't Erlang)
- Too much network traffic



Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Solutions?

- Literate programming and org-mode
- Alternatives to MPI and/or use Erlang
- ...
- Keep things as they are (only a few painful events a year)

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Single machine applications

- Run applications in a single, multicore (e.g., 12) machine
- Just verify if the machine is up

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Rethinking web-based applications

- Users can get into trouble.

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Rethinking web-based applications

- Users can get into trouble.

  Sure, but we can do a good job . . .

  - Provide state-of-the art statistical and computational approaches (R ;-)
  - Pedagogical examples and pipelines
  - Minimize the chance of users getting into trouble

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Rethinking web-based applications

- Users can get into trouble.

  Sure, but we can do a good job . . .

  - Provide state-of-the art statistical and computational approaches (R ;-)
  - Pedagogical examples and pipelines
  - Minimize the chance of users getting into trouble

- Web-based applications are here to stay

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

. . . so . . .

- Forget about them: just write R (plus C, etc) code
- Go for it
  - R will do its job (which is only part of the job)
  - HPC available
  - No problem with large data sets
  - But other tools and work necessary

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Regardless of web-based applications . . .

- Parallel computing can be used routinely
  - (*library(parallel)* in R $\geq$ 2.14.0)
- Large data sets with *ff* + parallelization.

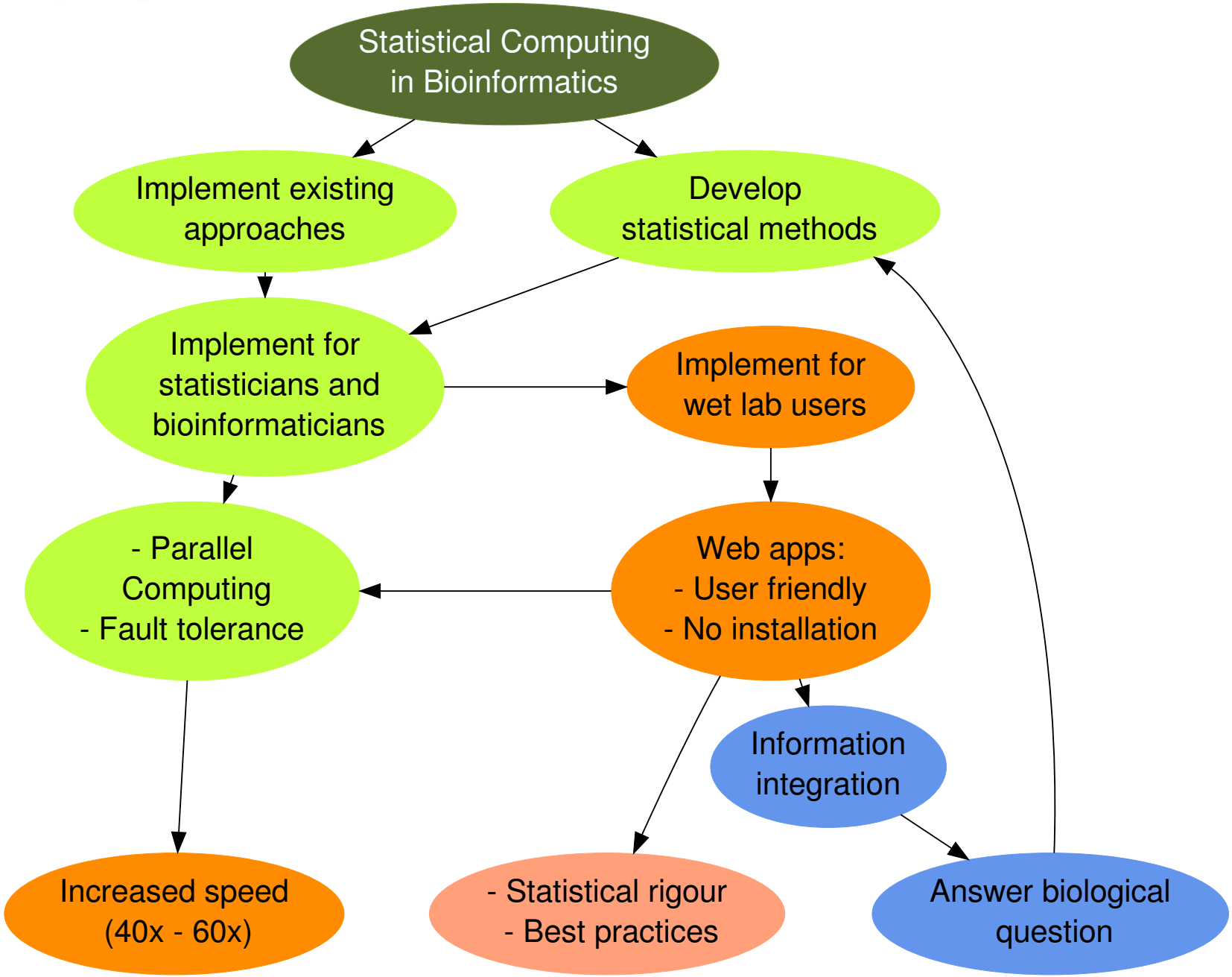
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Acknowledgements

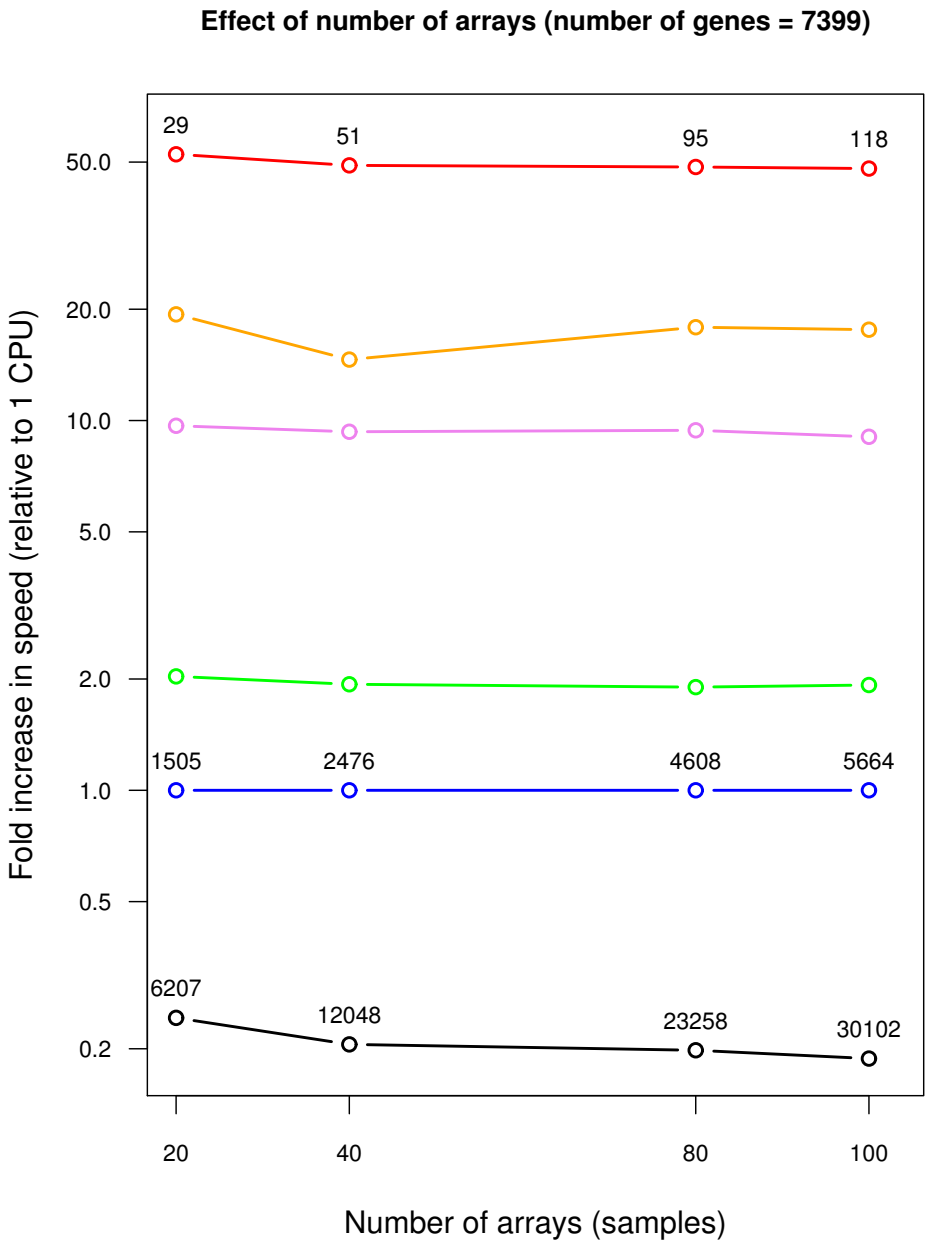
- O. M. Rueda, A. Alibés, A. Cañada, E. R. Morrissey, M. L. Neves, D. Rico.
- Funding: Fundación de Investigación Médica Mutua Madrileña, Project TIC2003-09331-C02-02 of the Spanish MEC and BIO2009-12458 of the Spanish MICINN. Ramón y Cajal Programme of the Spanish Ministry of Education and Science.
- CNIO (Spanish National Cancer Research Center).
- The R users and developers for a vibrant statistical computing community and amazing platform.
- The organizing and scientific committees of the III Jornadas de Usuarios de R.

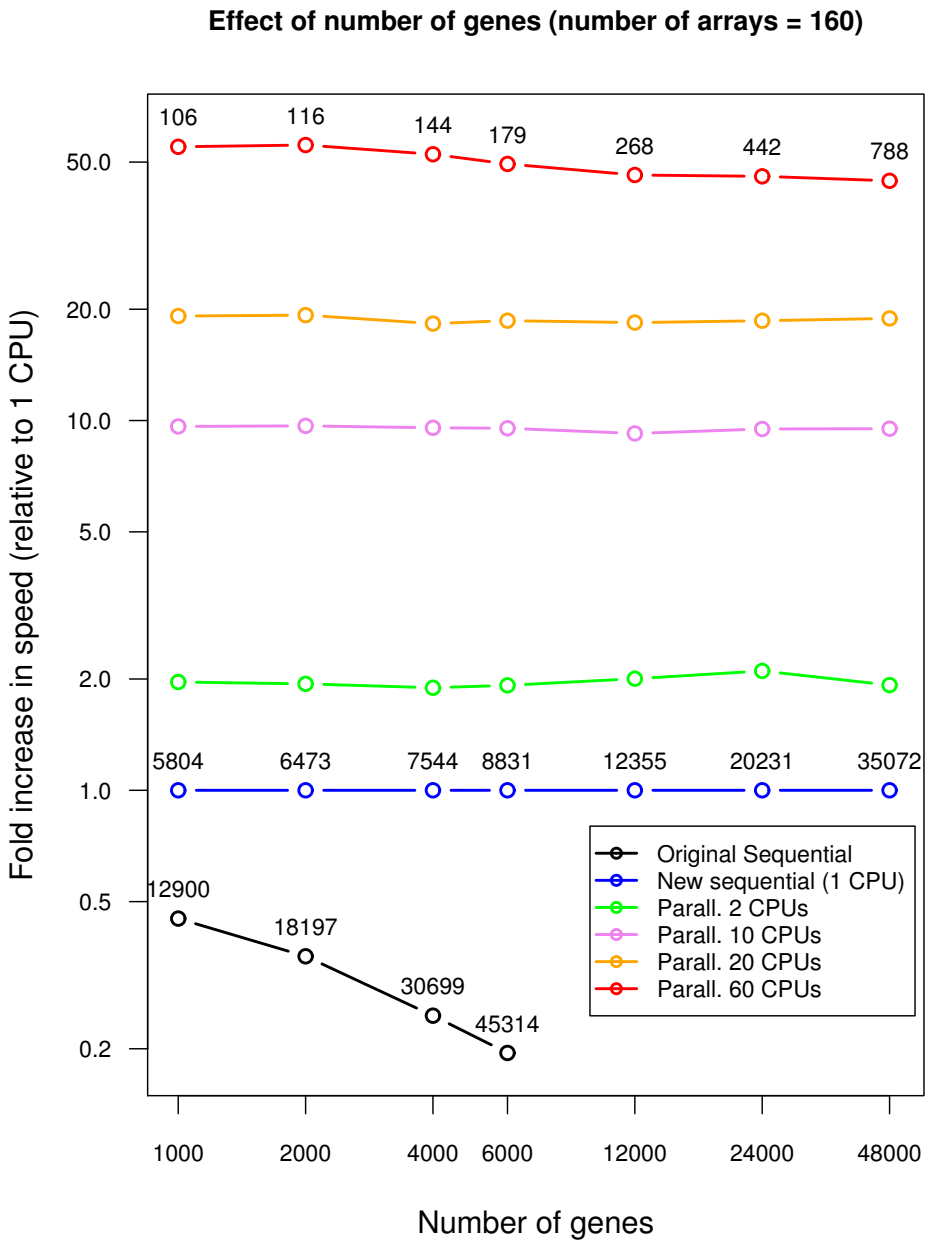
Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Trying to fit it all

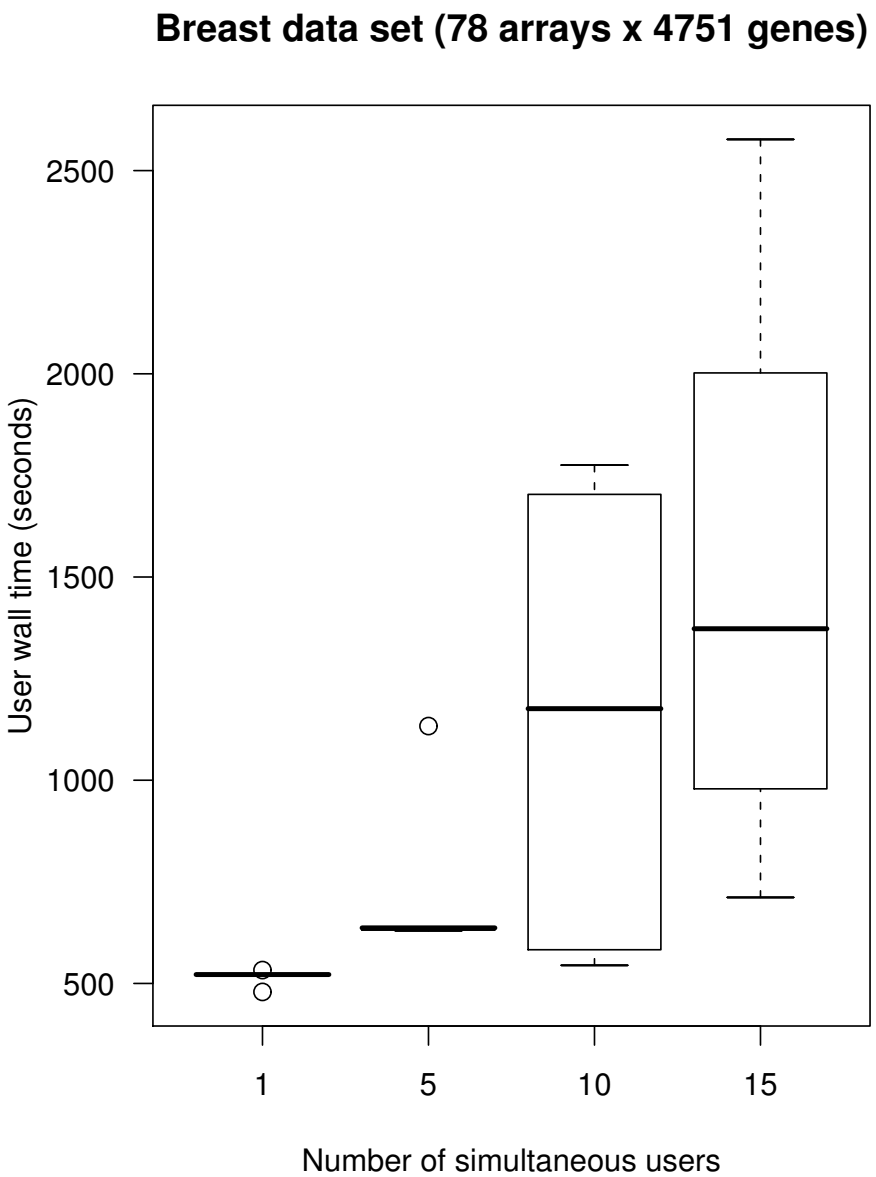


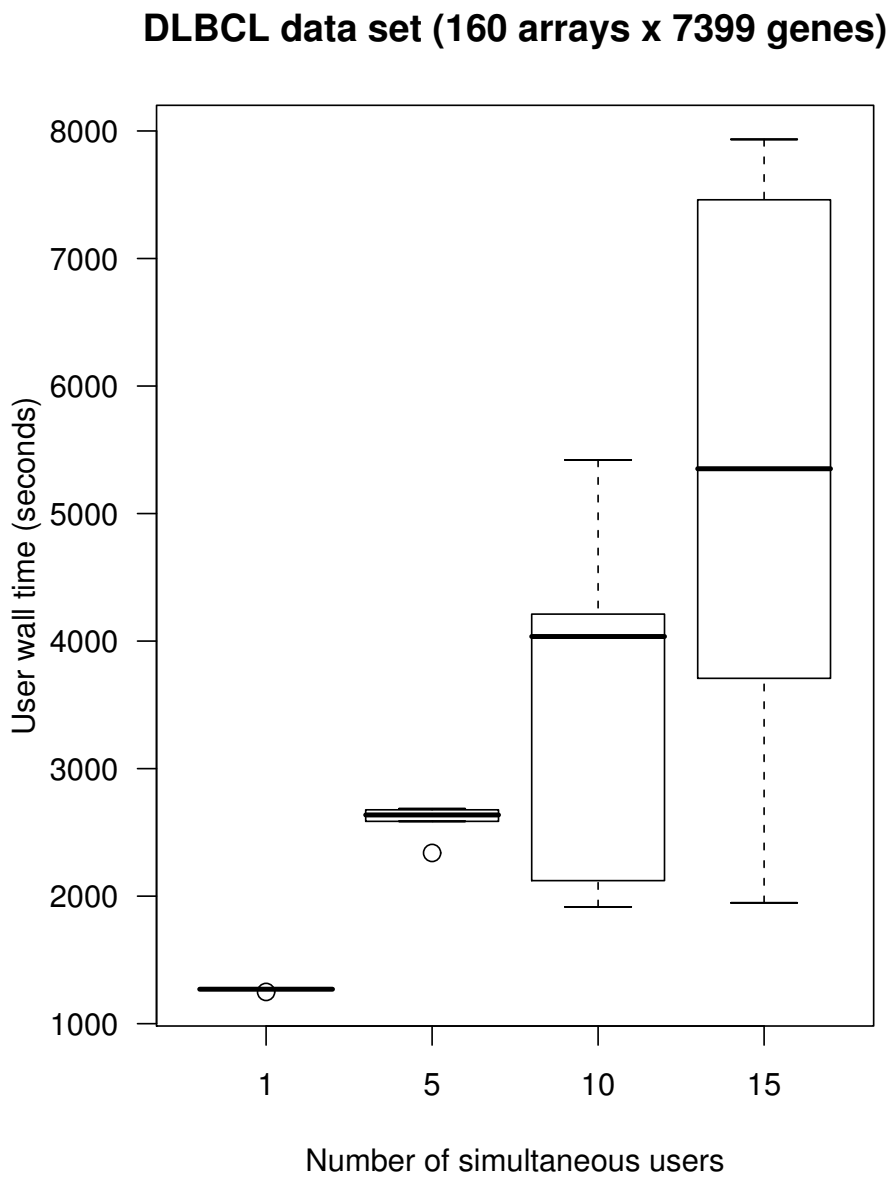
What do we gain?





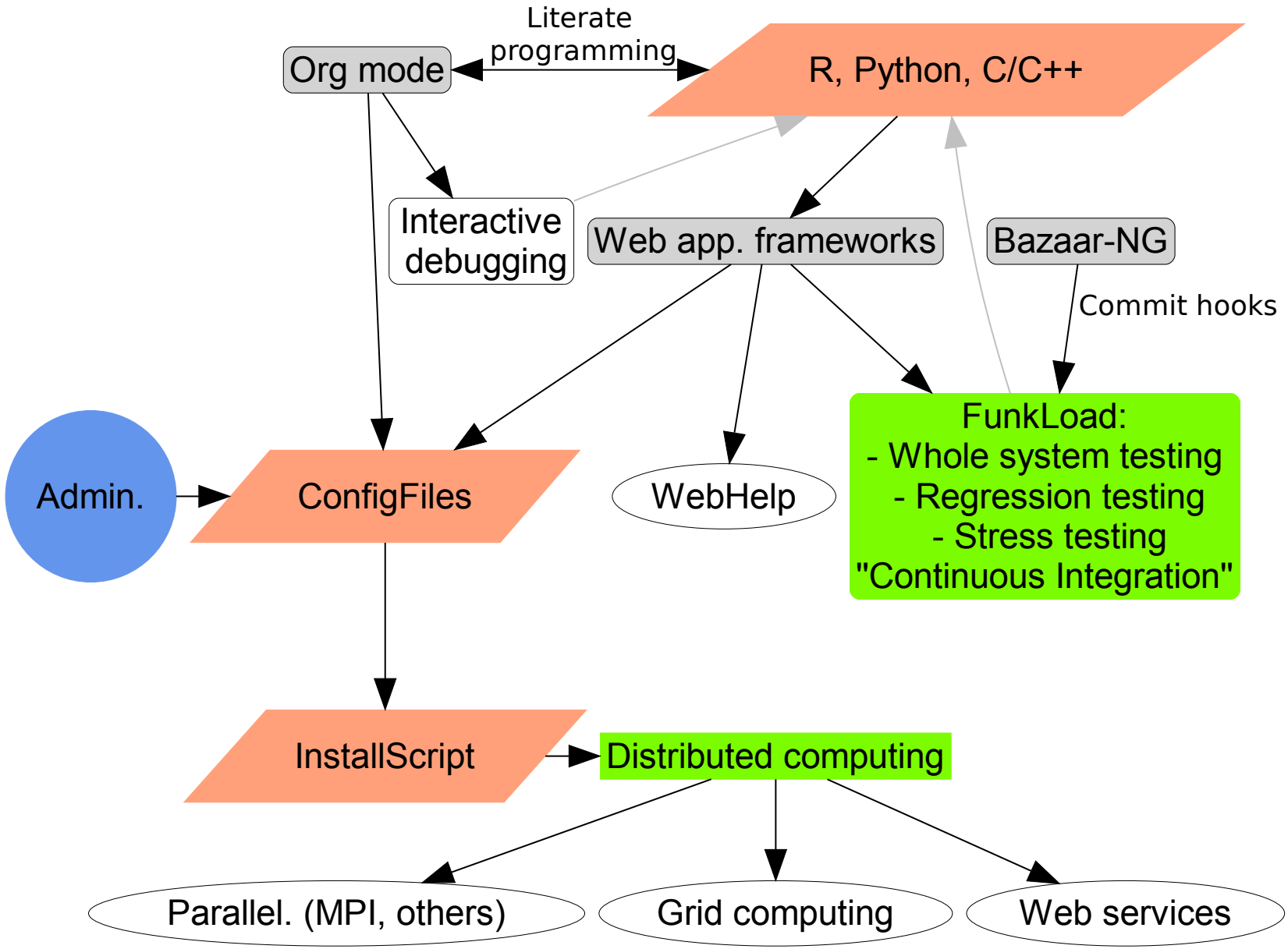
What do we gain?





Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Tools. . .



Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Too many languages

Impedance mismatch problem:

"Building Web-based applications requires the mastering of a number of languages/technologies (e.g. HTML, CSS, CGI, ASP, PHP, XML, etc..). Such languages and technologies were created to address different aspects on a by-need evolutionary manner. The result is a plethora of tools that are fitted together in an ad hoc fashion." El-Ansary, Grolaux, Van Roy, Rafea (2005) *"Overcoming the Multiplicity of Languages and Technologies for Web-Based Development Using a Multi-paradigm Approach"*.

- R and C
- HTML and Python: CGI, data entry, display
- Python (and others): control and monitor MPI
- Javascript: AJAX and figures

Context
Parallelizing code
Web applications
Large data sets and parallelization
R, C, and compression on the fly
Conclusions, future, et al.
Appendix

Other solutions?

Too many languages Use languages designed to overcome this problem: Hop, Links, QHTML.

Fault tolerance and too much traffic Alternatives to MPI?

- Linda and tuple spaces (also between-language funct.)
- Roll-our-own based on Rserve
- Have Erlang control R processes?