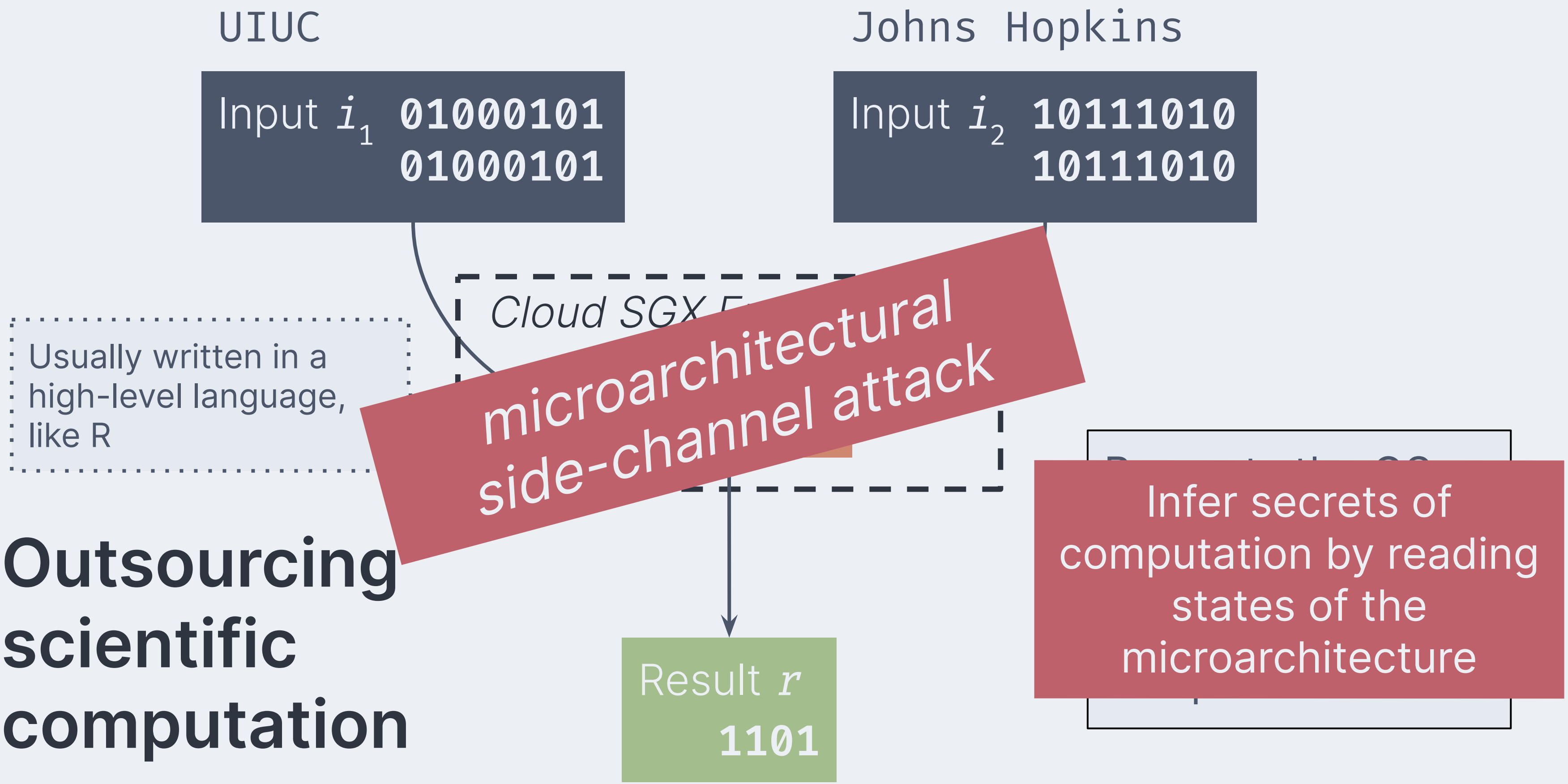
On Building the Data-Oblivious Virtual Environment

Tushar M. Jois, Hyun Bin Lee, Christopher W. Fletcher, Carl A. Gunter

Learning from Authoritative Security Experiment Results (LASER) Workshop
February 25, 2021



Usually written in a high-level language, like R

microarchitectural side-channel attack

Infer secrets of computation by reading states of the microarchitecture

Outsourcing scientific computation

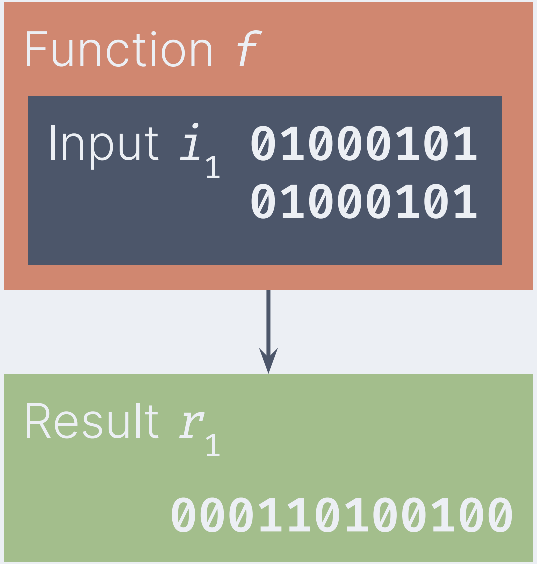
da·ta-o·bliv·i·ous com·pu·ta·tion (*n.*)

a program execution with the same observable characteristics **regardless** of the inputs provided

see also **constant-time programming**

Key insight:

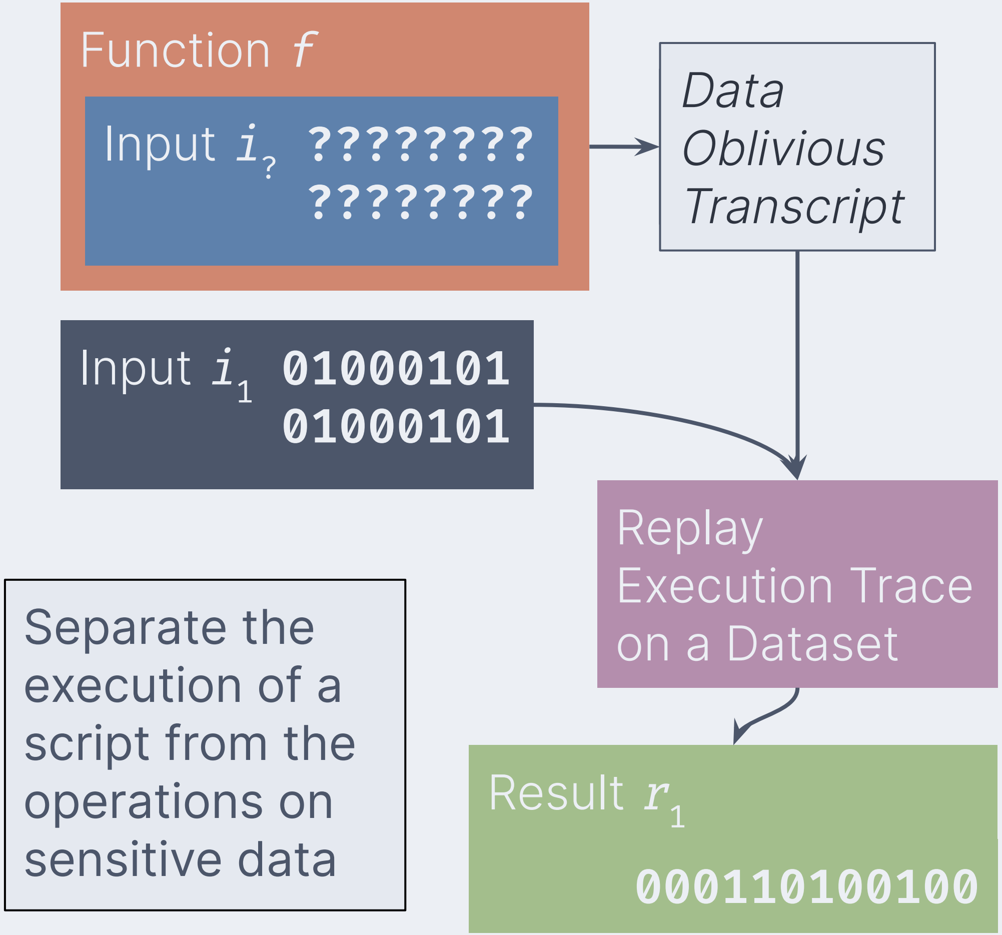
data-oblivious → data-unnecessary



Function f

Input i_1 01000101 01000101

Result r_1 000110100100



Function f

Input $i_?$???????? ????????

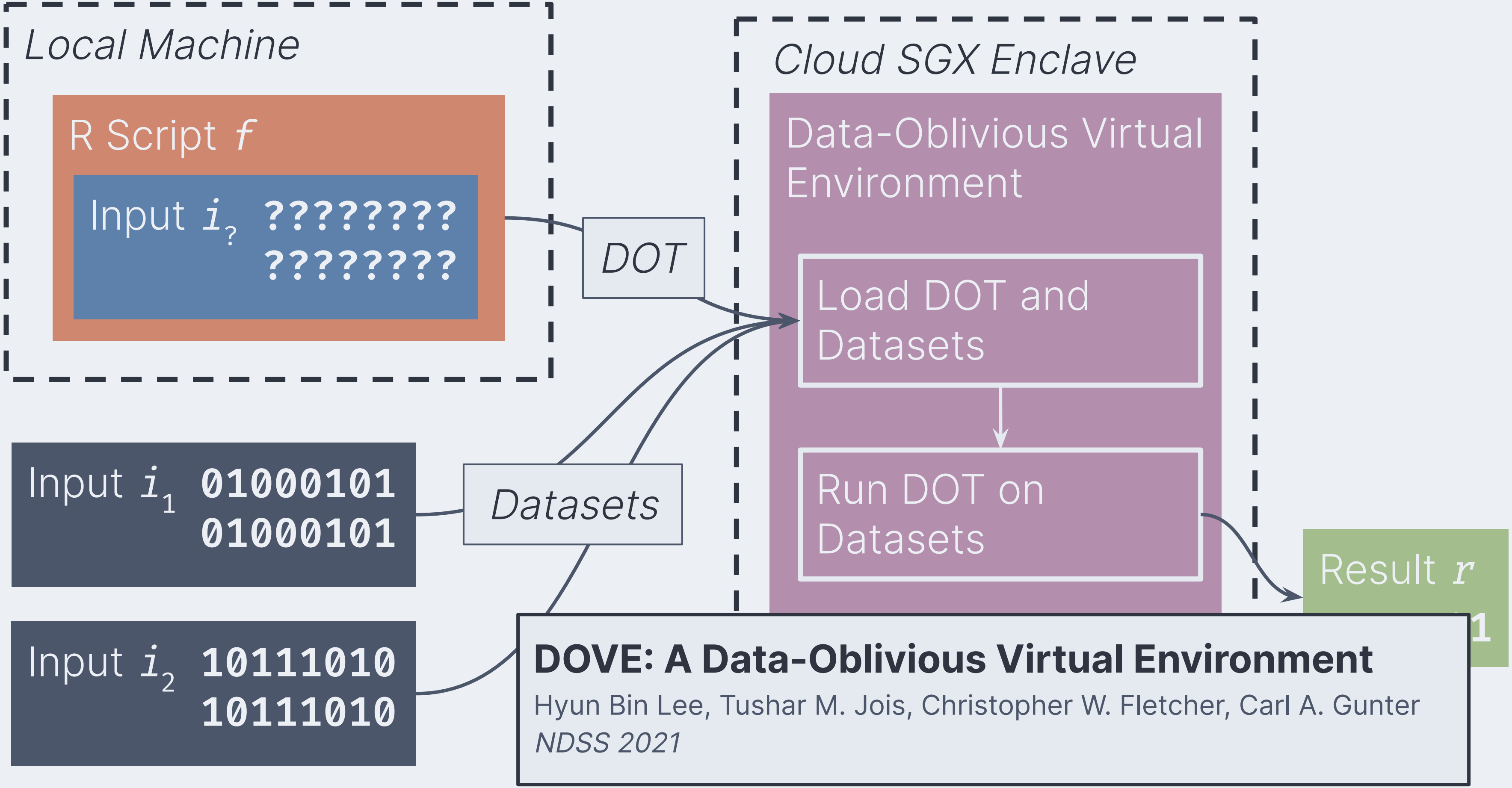
Data Oblivious Transcript

Input i_1 01000101 01000101

Replay Execution Trace on a Dataset

Result r_1 000110100100

Separate the execution of a script from the operations on sensitive data



DOVE: A Data-Oblivious Virtual Environment

Hyun Bin Lee, Tushar M. Jois, Christopher W. Fletcher, Carl A. Gunter

NDSS 2021

Our goal:

Design the first data-oblivious R stack.

A soft selective sweep during rapid evolution of gentle behaviour in an Africanized honeybee

Arian Avalos, Hailin Pan, Cai Li, et al.

Nature Communications 2017 8(1) 1550

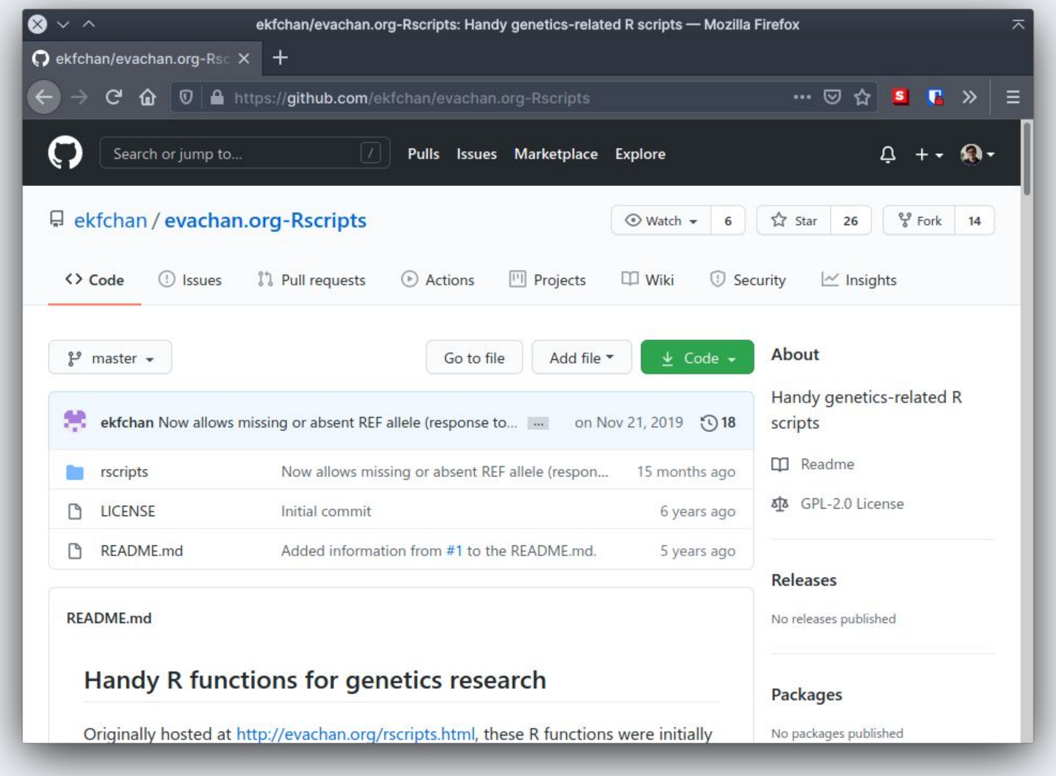
ILLINOIS

College of Agricultural, Consumer
& Environmental Sciences

🇺🇸 🇵🇷 🇨🇳 🇩🇰

Why use the bee study?

- A real, publicly-available dataset (1.3 GB)
- Similar to human genomics workloads
- Cross-university collaboration
- R code from a repository of genomics scripts



ekfchan / evachan.org-Rscripts

ekfchan Now allows missing or absent REF allele (response to... on Nov 21, 2019 18

rscripts	Now allows missing or absent REF allele (respon...	15 months ago
LICENSE	Initial commit	6 years ago
README.md	Added information from #1 to the README.md.	5 years ago

About

Handy genetics-related R scripts

Readme

GPL-2.0 License

Releases

No releases published

Packages

No packages published

README.md

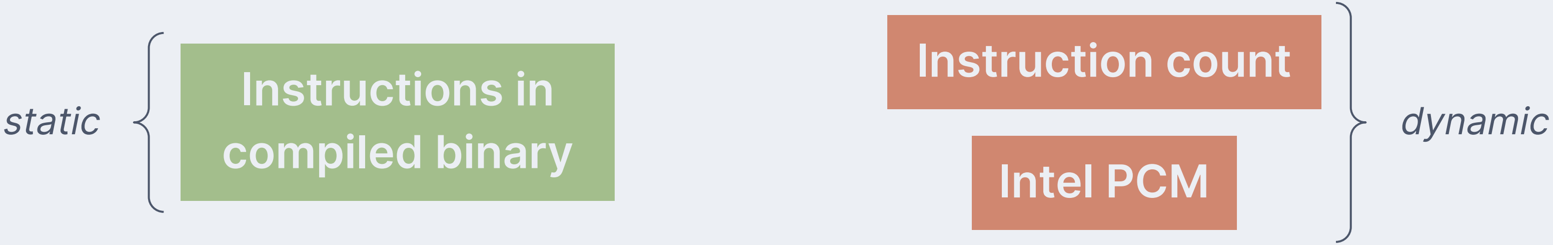
Handy R functions for genetics research

Originally hosted at http://evachan.org/rscripts.html, these R functions were initially

A **case study** for evaluating the data-obliviousness of R

Experimentally evaluating data-obliviousness

```
calc_snp_stats ← function(geno)
```



static

Instructions in compiled binary

Instruction count

Intel PCM

dynamic

Instructions in compiled binary

Two types of problematic instructions:

- Variable-time instructions
- Conditional jumps on sensitive data

On Subnormal Floating Point and Abnormal Timing

Marc Andrysco, David Kohlbrenner, Keaton Mowery, et al.

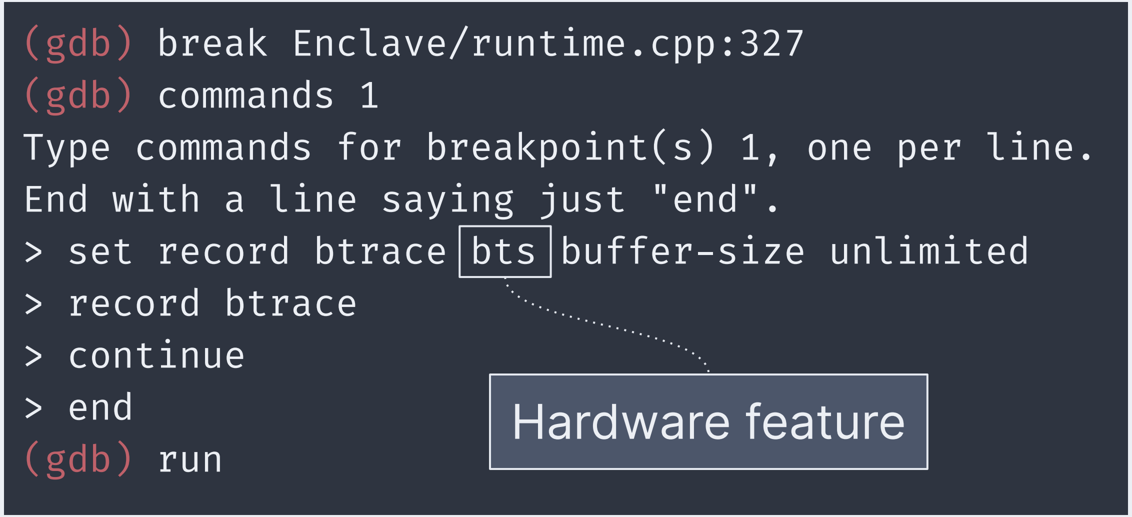
IEEE S&P (Oakland) 2015

Conditional jumps must **NOT** touch sensitive data

Instructions from `libfixedtimefixedpoint`

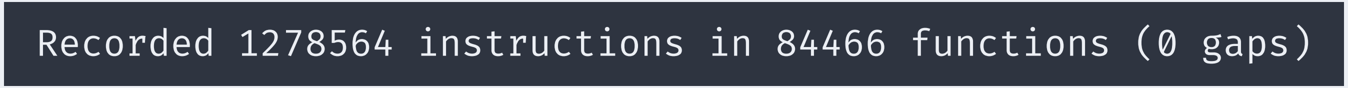
```
add
and
call
cdqe
cmp
mul
je
jmp
jne
lea
mov
movabs
movsd
movsx
movsxd
movzx
mul
neg
not
or
pop
push
rep
ret
sar
sbb
seta
setae
setbe
sete
setg
setl
setle
setne
shli
shr
sub
test
xor
```

Instruction count



```
(gdb) break Enclave/runtime.cpp:327
(gdb) commands 1
Type commands for breakpoint(s) 1, one per line.
End with a line saying just "end".
> set record btrace bts buffer-size unlimited
> record btrace
> continue
> end
(gdb) run
```

Hardware feature



```
Recorded 1278564 instructions in 84466 functions (0 gaps)
```

Intel Performance Counter Monitor (PCM)

cycle counts

```
getCycles
getCyclesLostDueL3CacheMisses
getCyclesLostDueL2CacheMisses
```

bytes to/from memory controller

```
getBytesReadFromMC
getBytesWrittenToMC
getIORequestBytesFromMC
```

cache hits & misses

```
getL2CacheHitRatio
getL3CacheHitRatio
getL3CacheMisses
getL2CacheMisses
getL2CacheHits
getL3CacheHitsNoSnoop
getL3CacheHitsSnoop
getL3CacheHits
```

```
calc_snp_stats ← function(geno)
{
    ## Eva KF Chan
    ## http://evachan.org

    m ← nrow(geno)      ## number of snps
    n ← ncol(geno)      ## number of individuals

    geno[(geno≠0) & (geno≠1) & (geno≠2)] ← NA
    geno ← as.matrix(geno)

    n0 ← apply(geno==0,1,sum,na.rm=T)
    n1 ← apply(geno==1,1,sum,na.rm=T)
    n2 ← apply(geno==2,1,sum,na.rm=T)
    n ← n0 + n1 + n2

    ## (snip) ##
}
```

NA

Similar to `null` in other languages

In

tion

side-channel

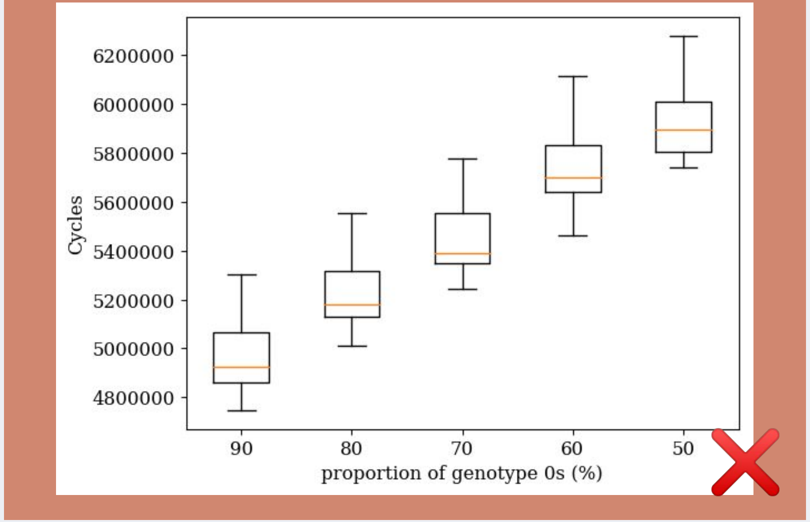
Side-channels in &

Instructions in compiled binary

Conditional branches on data

❌

Intel PCM



❌

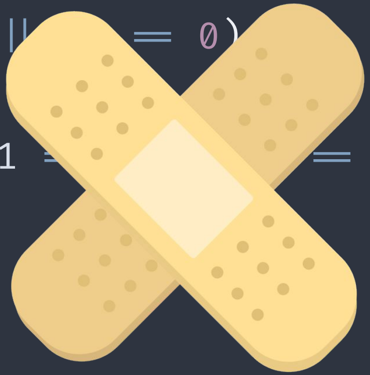
Instruction count

Expression	Value	Instr. Count
`0 & 0`	`0`	45
`0 & 1`	`0`	45
`0 & NA`	`0`	45
`1 & 0`	`0`	47
`NA & 0`	`0`	47
`NA & 1`	`NA`	53
`NA & NA`	`NA`	53
`1 & 1`	`1`	54
`1 & NA`	`NA`	57

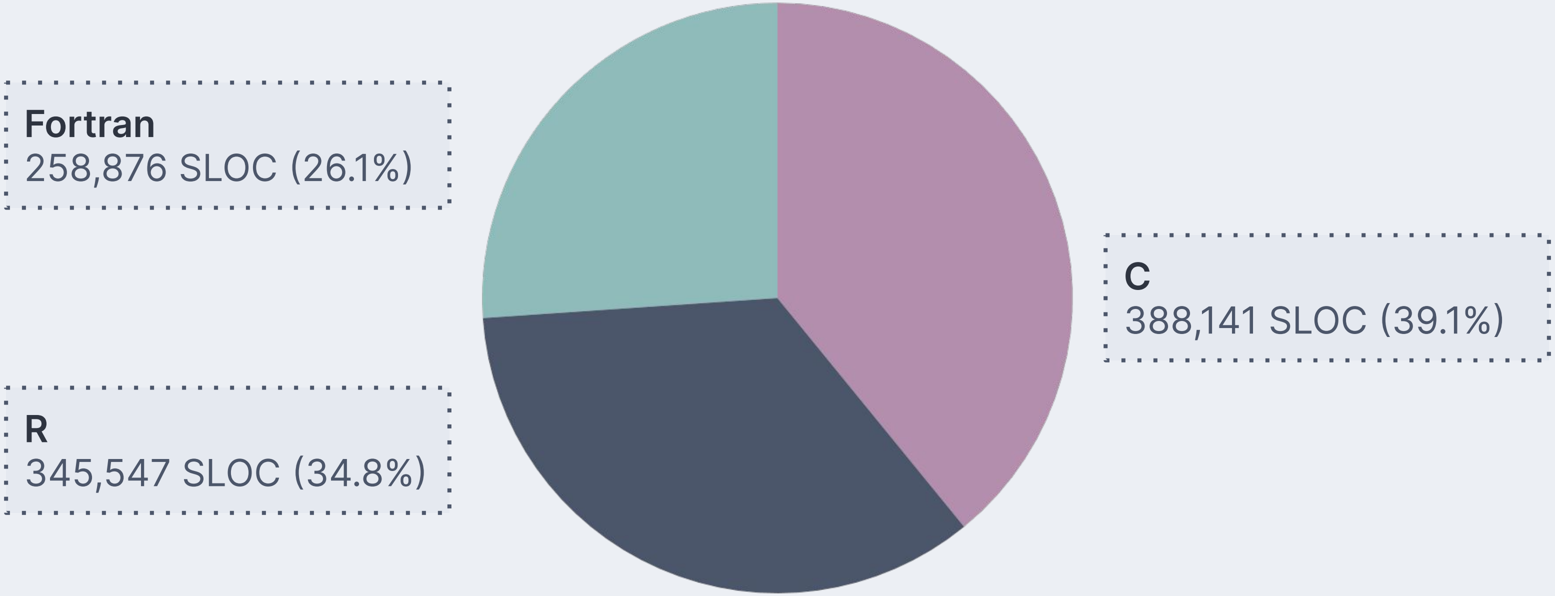
❌

```
geno[(geno≠0) & (geno≠1) & (geno≠2)] ← NA
```

```
# R interpreter implementation of &
if (x1 == 0 ||      == 0)
 pa[i] = 0;
} else if (x1 =             == NA) {
 pa[i] = NA;
} else {
 pa[i] = 1;
}
```



R interpreter



Solution design

Build a data-oblivious virtual environment

Correctness

Data-obliviousness

- *Instructions in compiled binary*
- *Instruction count*
- *Intel PCM*

Expressiveness

Efficiency

Solution design

Build a data-oblivious virtual environment

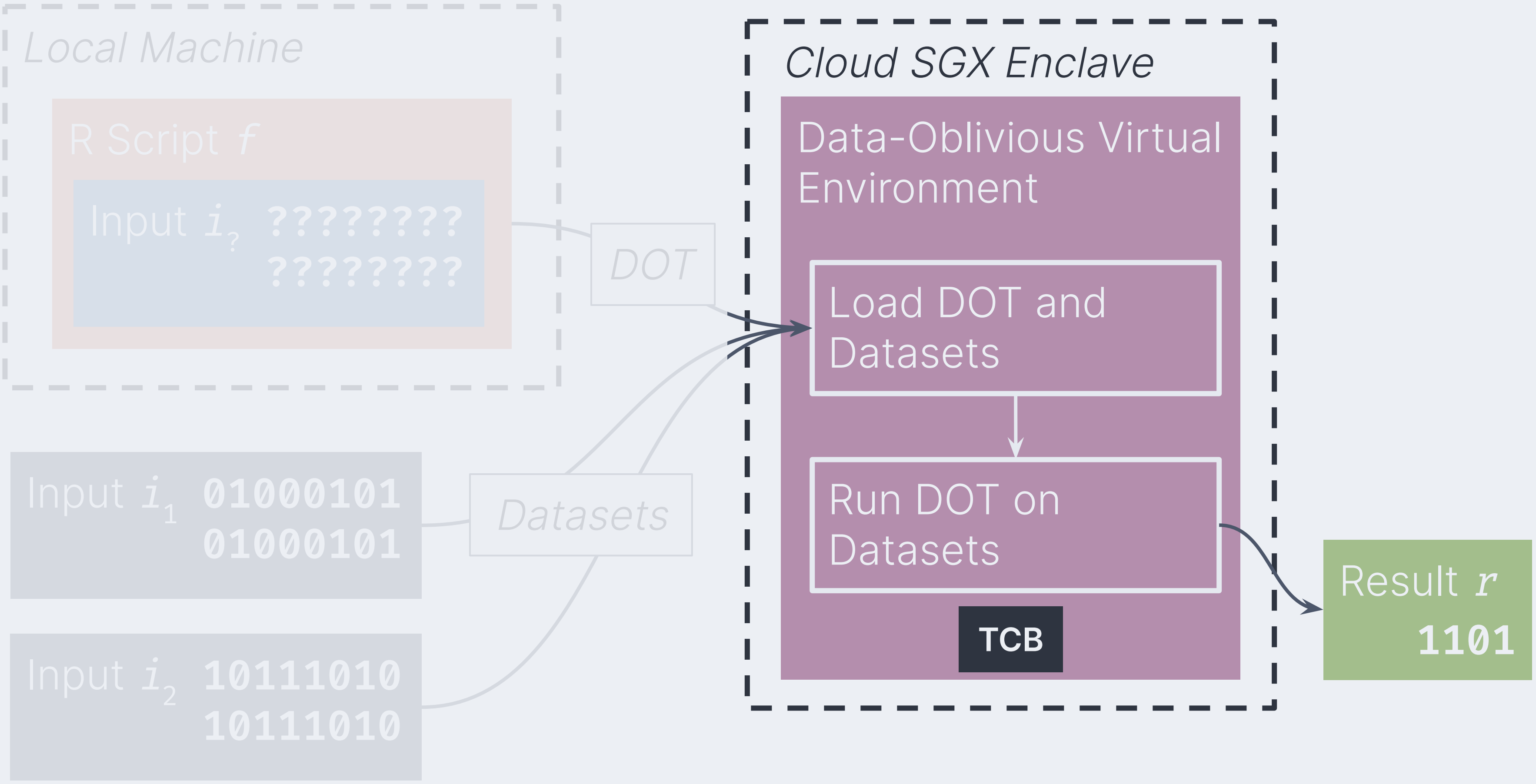
~~Correctness~~

Data-obliviousness

- *Instructions in compiled binary*
- *Instruction count*
- *Intel PCM*

Expressiveness

Efficiency



Local Machine

R Script f

Input $i_?$????????
????????

DOT

Input i_1 01000101
01000101

Input i_2 10111010
10111010

Datasets

Cloud SGX Enclave

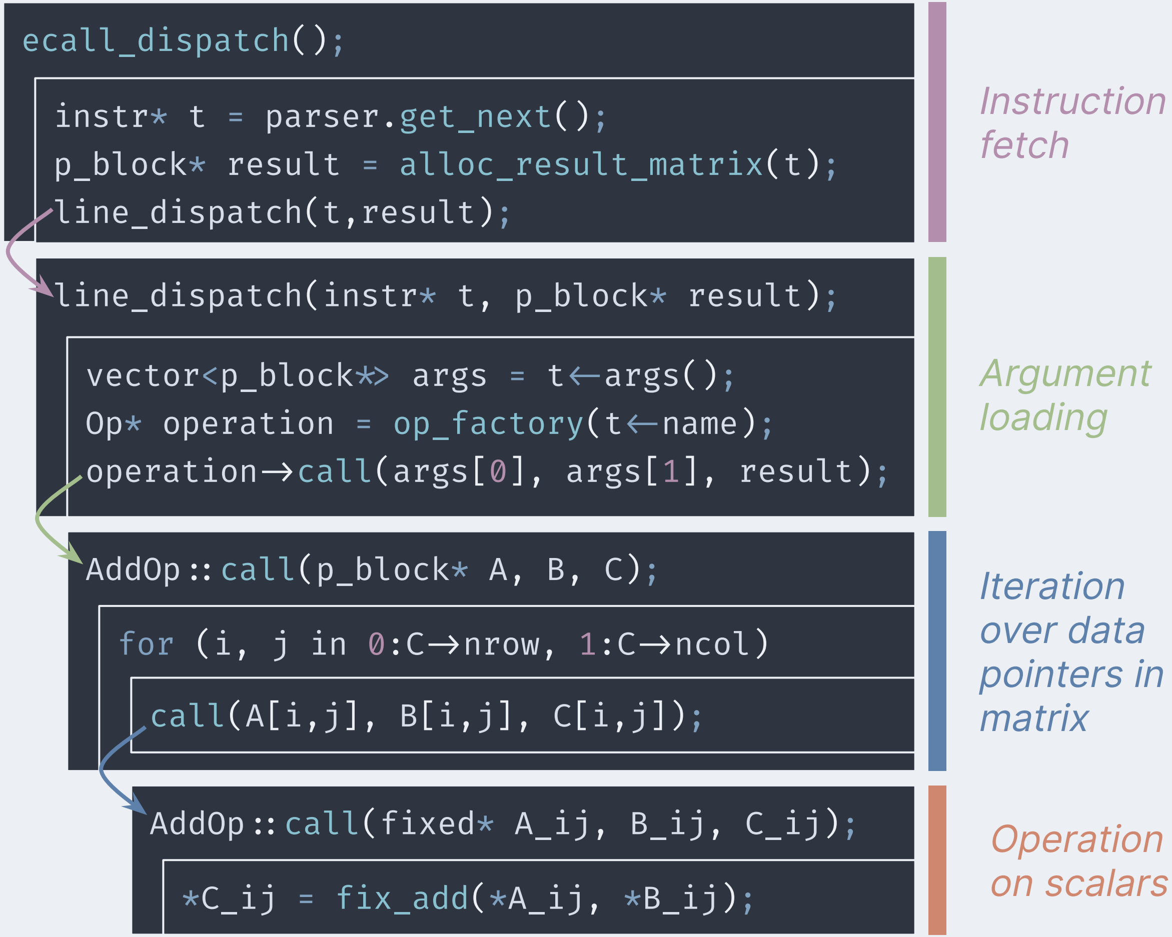
Data-Oblivious Virtual Environment

Load DOT and Datasets

Run DOT on Datasets

TCB

Result r
1101



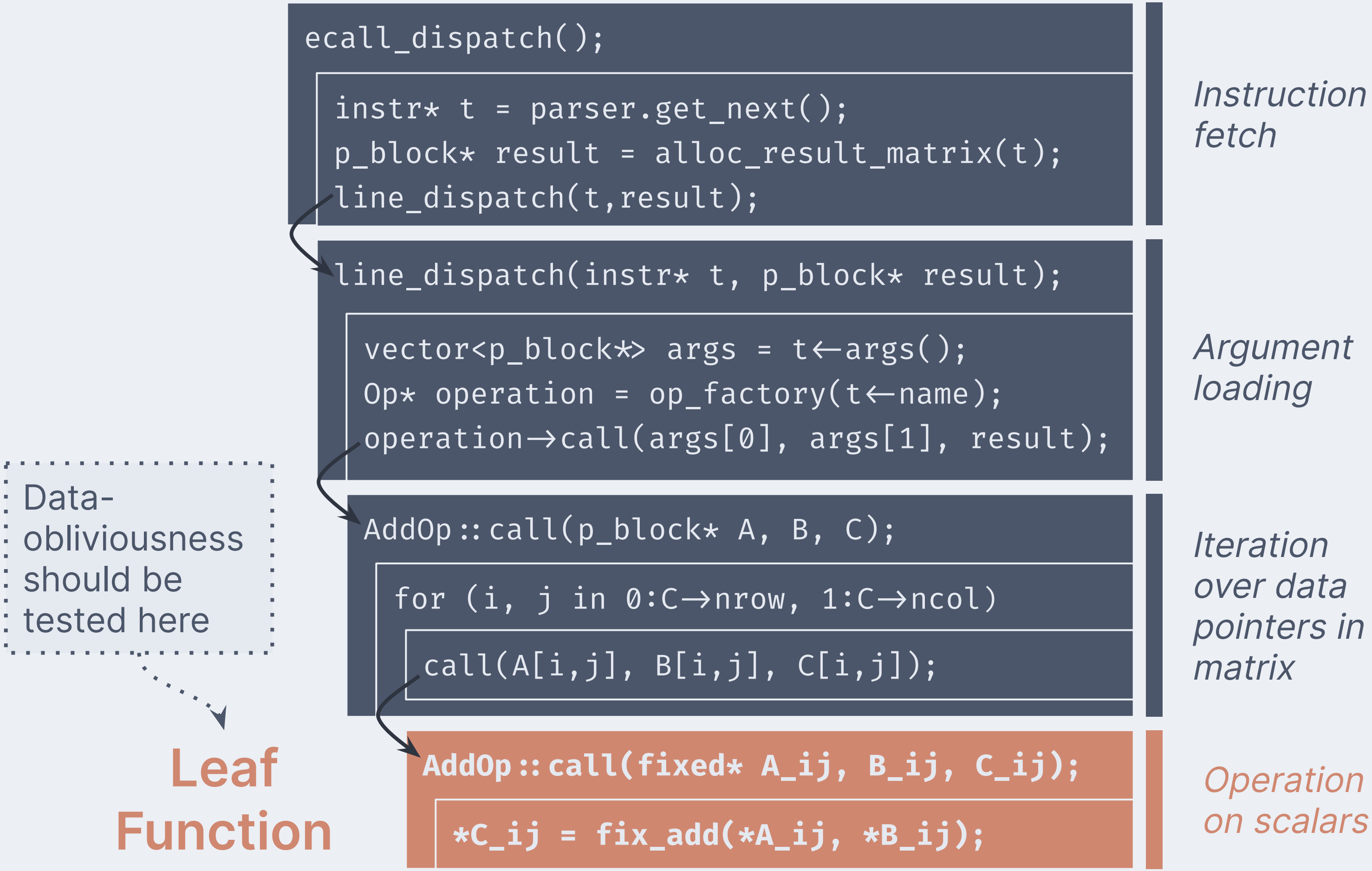
```
ecall_dispatch();
  instr* t = parser.get_next();
  p_block* result = alloc_result_matrix(t);
  line_dispatch(t,result);

  line_dispatch(instr* t, p_block* result);
    vector<p_block*> args = t←args();
    Op* operation = op_factory(t←name);
    operation→call(args[0], args[1], result);

    AddOp::call(p_block* A, B, C);
      for (i, j in 0:C→nrow, 1:C→ncol)
        call(A[i,j], B[i,j], C[i,j]);

        AddOp::call(fixed* A_ij, B_ij, C_ij);
          *C_ij = fix_add(*A_ij, *B_ij);
```

Instruction fetch

Argument loading

Iteration over data pointers in matrix

Operation on scalars



```
ecall_dispatch();
    instr* t = parser.get_next();
    p_block* result = alloc_result_matrix(t);
    line_dispatch(t,result);

line_dispatch(instr* t, p_block* result);
    vector<p_block*> args = t←args();
    Op* operation = op_factory(t←name);
    operation→call(args[0], args[1], result);

AddOp::call(p_block* A, B, C);
    for (i, j in 0:C→nrow, 1:C→ncol)
        call(A[i,j], B[i,j], C[i,j]);

AddOp::call(fixed* A_ij, B_ij, C_ij);
    *C_ij = fix_add(*A_ij, *B_ij);
```

Instruction fetch

Argument loading

Iteration over data pointers in matrix

Operation on scalars

Data-obliviousness should be tested here

Leaf Function

Side-channels in leaf functions

Instructions in compiled binary

```
cmovne
```

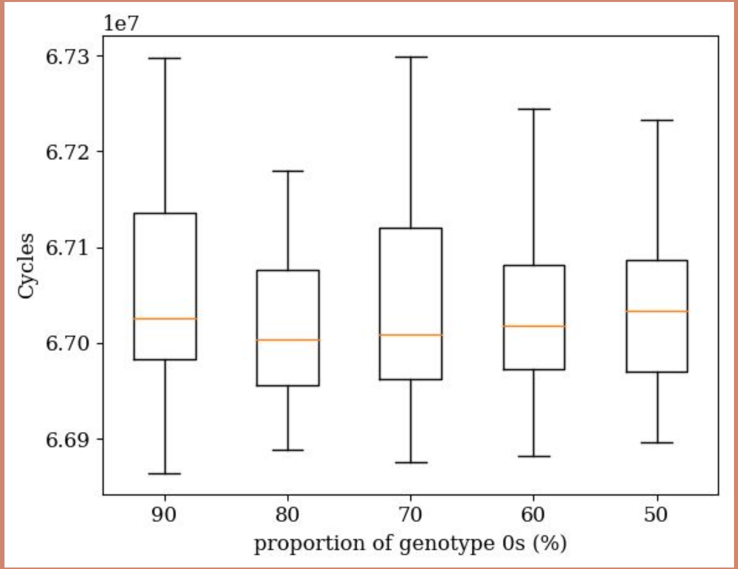
✔

Instruction count

```
TESTS
    Testing Abs (1/45)...
    Testing Abs, ratio 0.1...
    Testing Abs, ratio 0.2...

          Passed
```

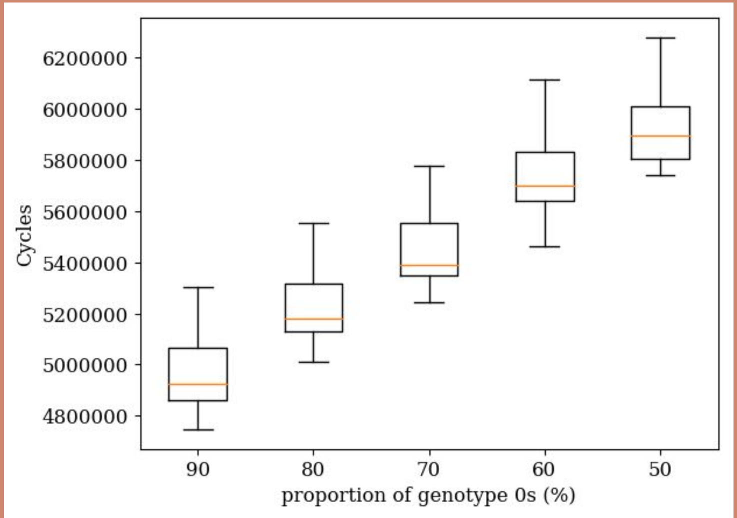
✔

Intel PCM

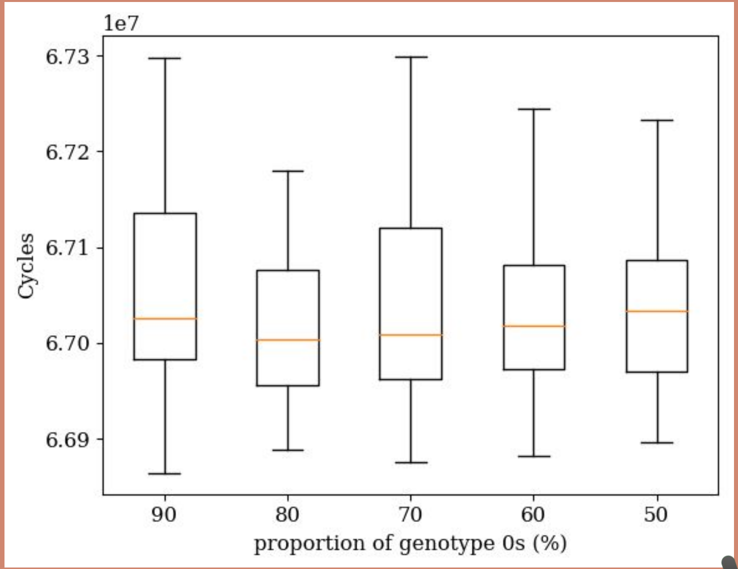


```
geno[(geno≠0) & (geno≠1) & (geno≠2)] ← NA
```

Intel PCM (Base R)



Intel PCM (DOVE)



✓

Solution design

Build a data-oblivious virtual environment

~~Correctness~~

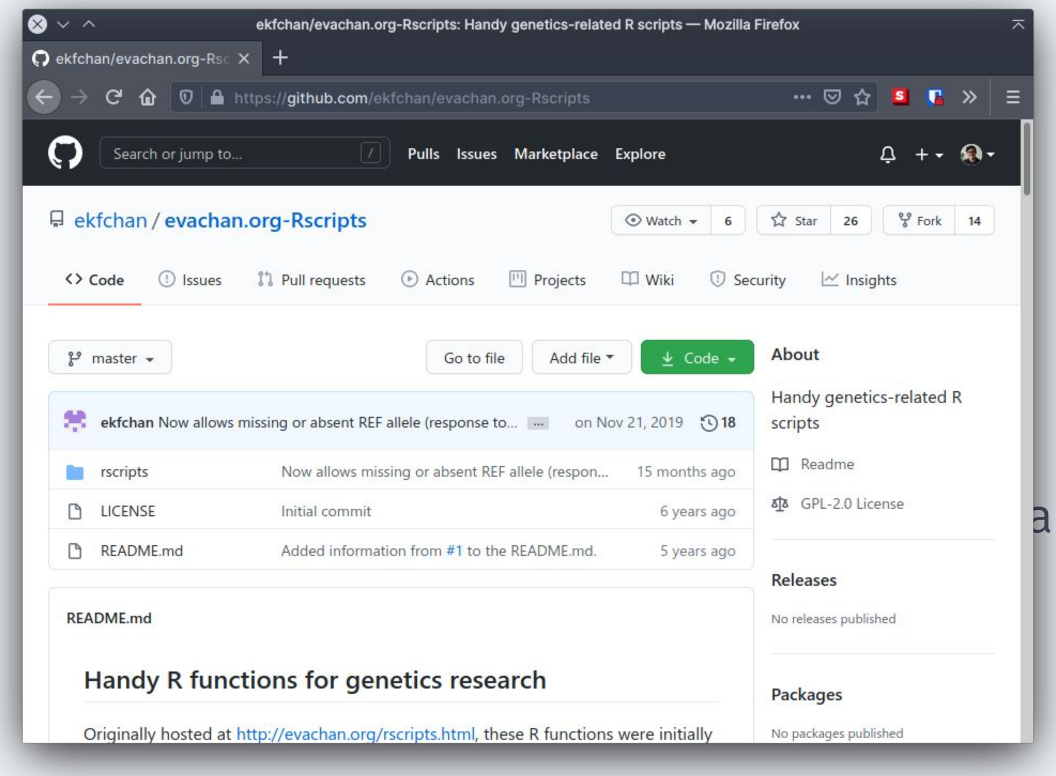
~~Data-obliviousness~~

- *~~Instructions in compiled binary~~*
- *~~Instruction count~~*
- *~~Intel PCM~~*

Expressiveness

Efficiency

abs sqrt
sign +
< ≥
any sum
is.infinite



ekfchan / evachan.org-Rscripts

ekfchan Now allows missing or absent REF allele (response to... on Nov 21, 2019 18

rscripts	Now allows missing or absent REF allele (respon...	15 months ago
LICENSE	Initial commit	6 years ago
README.md	Added information from #1 to the README.md.	5 years ago

About

Handy genetics-related R scripts

Readme

GPL-2.0 License

Releases

No releases published

Packages

No packages published

README.md

Handy R functions for genetics research

Originally hosted at http://evachan.org/rscripts.html, these R functions were initially

sin tan
%/% >
! all
a is.nan
matrix dim

Solution design

Build a data-oblivious virtual environment

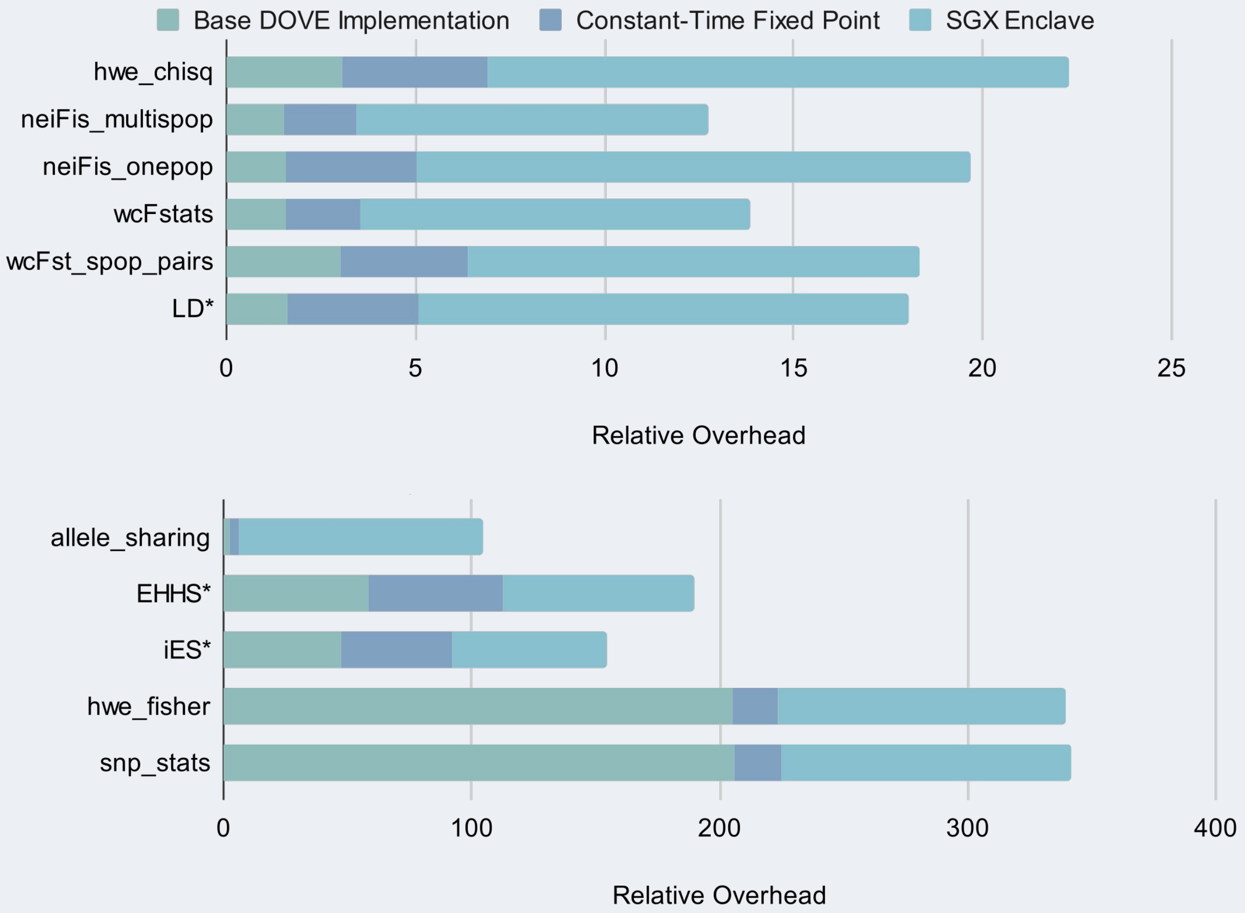
~~Correctness~~

~~Data-obliviousness~~

- ~~*Instructions in compiled binary*~~
- ~~*Instruction count*~~
- ~~*Intel PCM*~~

~~Expressiveness~~

Efficiency



$O(m * n)$ space

2,808,570×60 dataset

(*)

$O(m^2)$ space

10,000×60 dataset

Solution design

Build a data-oblivious virtual environment

~~Correctness~~

~~Data-obliviousness~~

- *~~Instructions in compiled binary~~*
- *~~Instruction count~~*
- *~~Intel PCM~~*

~~Expressiveness~~

~~Efficiency~~

https://github.com/**dove-project/benchmarks**

Discussion

- Did you use experimentation artifacts borrowed from the community?
- Did you attempt to replicate or reproduce results of earlier research as part of your work?
- What can be learned from your methodology and your experience using your methodology?
- What did you try that did not succeed before getting to the results you
- Did you produce any intermediate results including possible unsuccessful tests or experiments?
- Did you share experimentation artifacts with the community?

Discussion

- Did you use experimentation artifacts borrowed from the community?
- Did you attempt to replicate or reproduce results of earlier research as part of your work?
- What can be learned from your methodology and your experience using your methodology?
- What did you try that did not succeed before getting to the results you
- Did you produce any intermediate results including possible unsuccessful tests or experiments?
- Did you share experimentation artifacts with the community?

Discussion

- Did you use experimentation artifacts borrowed from the community?
- Did you attempt to replicate or reproduce results of earlier research as part of your work?
- What can be learned from your methodology and your experience using your methodology?
- What did you try that did not succeed before getting to the results you
- Did you produce any intermediate results including possible unsuccessful tests or experiments?
- Did you share experimentation artifacts with the community?

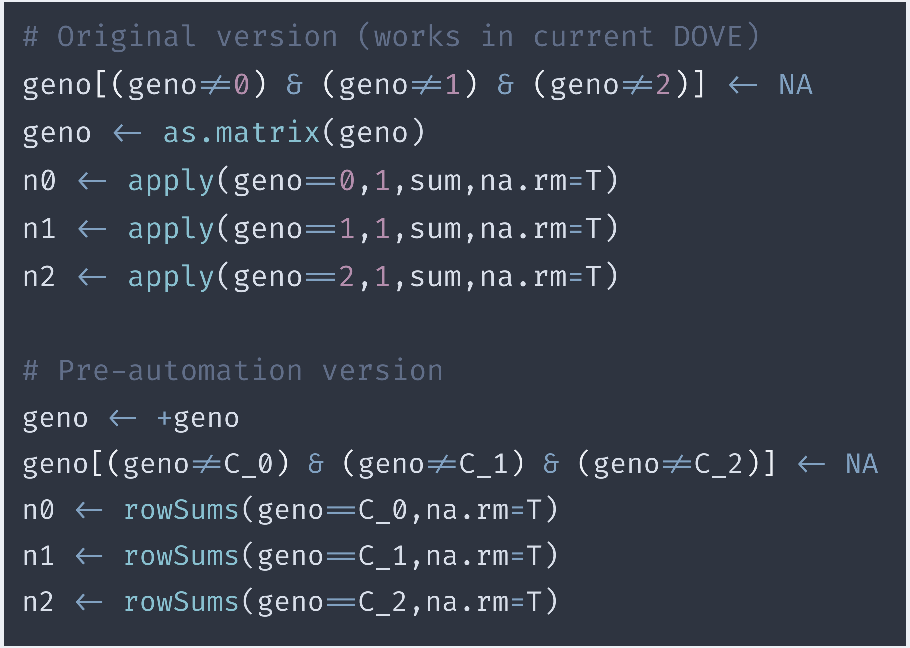
Intermediate results: data-obliviousness

- Fisher test is used in script to measure deviation from Hardy-Weinberg Equilibrium
- Originally a part of external library, didn’t test it, but clearly wrong assumption
  - When we started to look at it, saw it failed our instruction tests -- factorials
  - Rewrote it to use front-end primitives -- worse performance, but security guaranteed (and smaller TCB)
- Insecure (4.9x overhead) to secure (315x overhead)

$$p = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)!\ (c+d)!\ (a+c)!\ (b+d)!}{a!\ \ b!\ \ c!\ \ d!\ \ n!}$$

Intermediate results: expressiveness

- Original DOVE design required end-users to modify their R code before a DOT was generated
- Not a good design
    - restricts expressiveness to what the user knows how to write using DOVE
    - Might as well learn a new language
- Created an automator that instruments R base functions & structures to use DOVE counterparts
    - No need to manually write DOVE



```
# Original version (works in current DOVE)
geno[(geno≠0) & (geno≠1) & (geno≠2)] ← NA
geno ← as.matrix(geno)
n0 ← apply(geno==0,1,sum,na.rm=T)
n1 ← apply(geno==1,1,sum,na.rm=T)
n2 ← apply(geno==2,1,sum,na.rm=T)

# Pre-automation version
geno ← +geno
geno[(geno≠C_0) & (geno≠C_1) & (geno≠C_2)] ← NA
n0 ← rowSums(geno==C_0,na.rm=T)
n1 ← rowSums(geno==C_1,na.rm=T)
n2 ← rowSums(geno==C_2,na.rm=T)
```

Intermediate results: efficiency

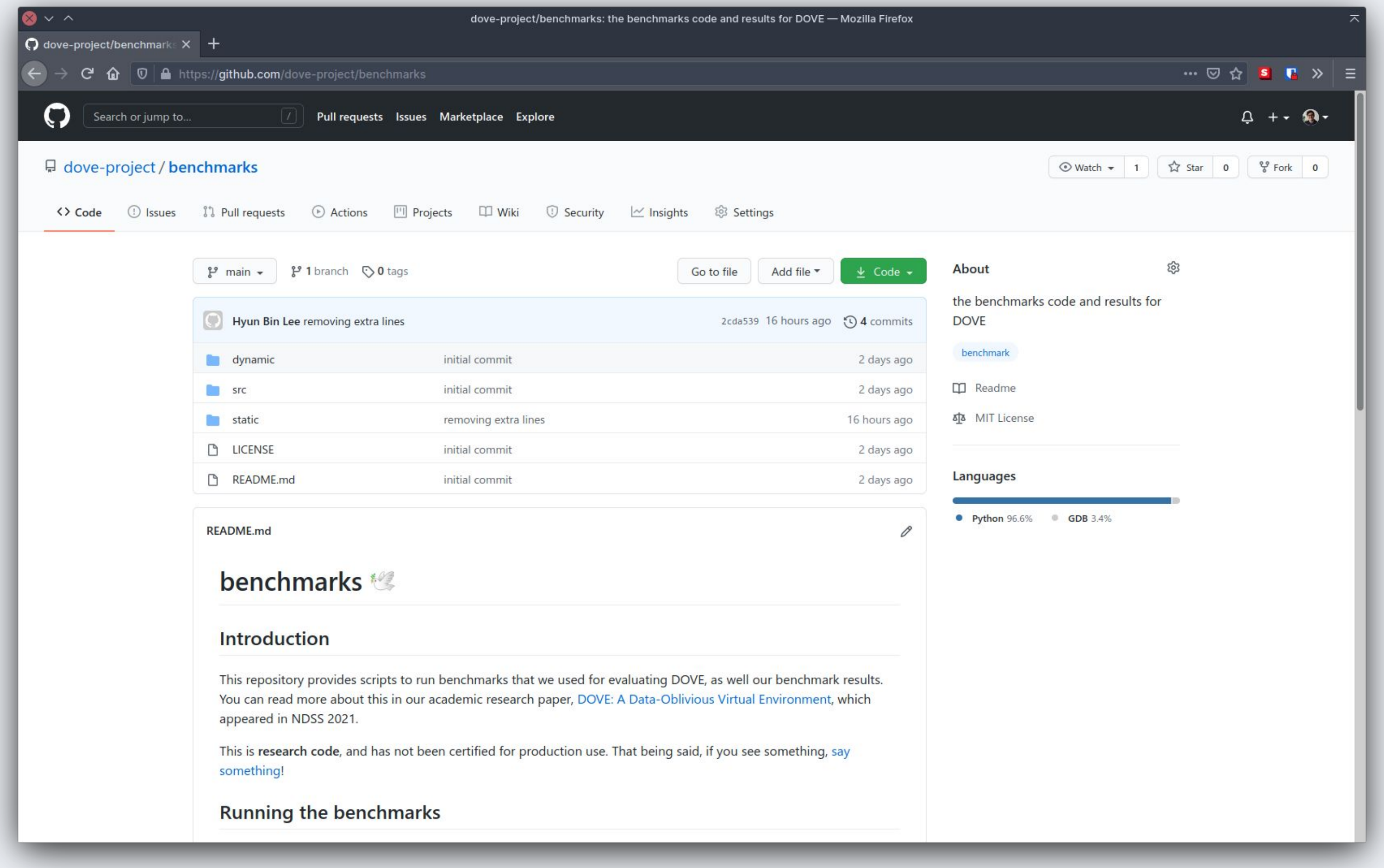
- Originally didn’t have `for` loops
  - Applications used `apply`, `rowSums`, and similar
- Applications that used loops had awful performance
  - Loops would just get unrolled
  - DOT became size $O(n)$
- Performance made us realize that loops were important enough
  - `apply` wasn’t enough
  - So, we implemented it

Script	**Overhead before `for`**	**Overhead after `for`**
`allele_sharing`	295x	105x
`EHHS*`	1246x	189x
`iES*`	1204x	154x
`LD*`	220x	18x

Discussion

- Did you use experimentation artifacts borrowed from the community?
- Did you attempt to replicate or reproduce results of earlier research as part of your work?
- What can be learned from your methodology and your experience using your methodology?
- What did you try that did not succeed before getting to the results you
- Did you produce any intermediate results including possible unsuccessful tests or experiments?
- Did you share experimentation artifacts with the community?

`https://github.com/`**`dove-project/benchmarks`**



Hyun Bin Lee removing extra lines 2cda539 16 hours ago 4 commits

dynamic	initial commit	2 days ago
src	initial commit	2 days ago
static	removing extra lines	16 hours ago
LICENSE	initial commit	2 days ago
README.md	initial commit	2 days ago

About

the benchmarks code and results for DOVE

benchmarks 🕊️

Introduction

This repository provides scripts to run benchmarks that we used for evaluating DOVE, as well our benchmark results. You can read more about this in our academic research paper, DOVE: A Data-Oblivious Virtual Environment, which appeared in NDSS 2021.

This is **research code**, and has not been certified for production use. That being said, if you see something, say something!

Running the benchmarks

Future work

Future plans

- Extend DOVE to more languages and frameworks
- Implement data-oblivious performance enhancement
- Understand what data-oblivious hardware instructions can support a system like DOVE

Post-workshop paper

- Review systematically the R side channels we discovered
- Re-run all benchmarks using most modern versions of the stack
    - New versions of libraries, R interpreter
- Several runs of the same benchmarks
    - Variance between benchmarks
- Look into performance on other enclaves, if possible

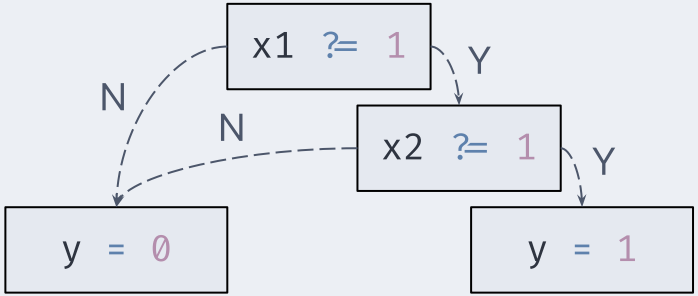
DOVE

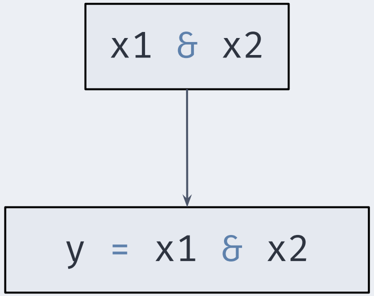
https://github.com/**dove-project/benchmarks**

```
# μArch Vulnerable
# Assume x1, x2 are private

if (x1 && x2) {
  y = 1;
} else {
  y = 0;
}
```

```
# Fixed (under assumptions)


y = x1 & x2;
```

Execution Trace





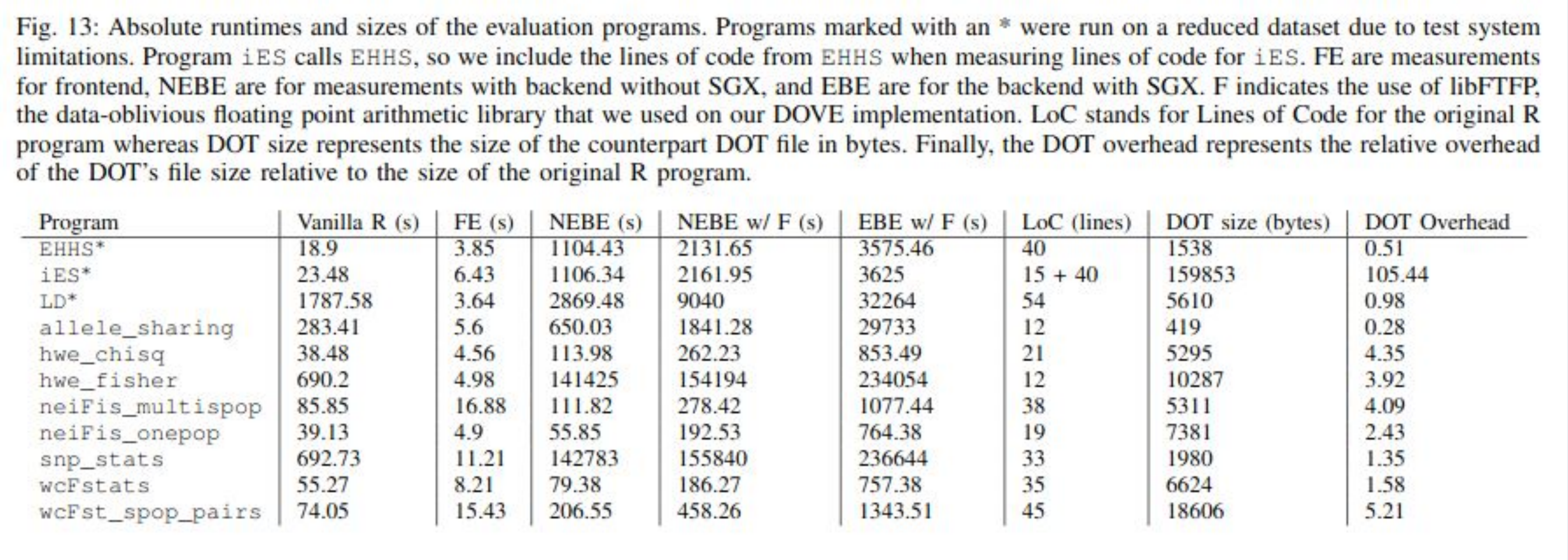


Fig. 13: Absolute runtimes and sizes of the evaluation programs. Programs marked with an * were run on a reduced dataset due to test system limitations. Program iES calls EHHS, so we include the lines of code from EHHS when measuring lines of code for iES. FE are measurements for frontend, NEBE are for measurements with backend without SGX, and EBE are for the backend with SGX. F indicates the use of libFTFP, the data-oblivious floating point arithmetic library that we used on our DOVE implementation. LoC stands for Lines of Code for the original R program whereas DOT size represents the size of the counterpart DOT file in bytes. Finally, the DOT overhead represents the relative overhead of the DOT's file size relative to the size of the original R program.

Program	Vanilla R (s)	FE (s)	NEBE (s)	NEBE w/ F (s)	EBE w/ F (s)	LoC (lines)	DOT size (bytes)	DOT Overhead
EHHS*	18.9	3.85	1104.43	2131.65	3575.46	40	1538	0.51
iES*	23.48	6.43	1106.34	2161.95	3625	15 + 40	159853	105.44
LD*	1787.58	3.64	2869.48	9040	32264	54	5610	0.98
allele_sharing	283.41	5.6	650.03	1841.28	29733	12	419	0.28
hwe_chisq	38.48	4.56	113.98	262.23	853.49	21	5295	4.35
hwe_fisher	690.2	4.98	141425	154194	234054	12	10287	3.92
neiFis_multispop	85.85	16.88	111.82	278.42	1077.44	38	5311	4.09
neiFis_onepop	39.13	4.9	55.85	192.53	764.38	19	7381	2.43
snp_stats	692.73	11.21	142783	155840	236644	33	1980	1.35
wcFstats	55.27	8.21	79.38	186.27	757.38	35	6624	1.58
wcFst_spop_pairs	74.05	15.43	206.55	458.26	1343.51	45	18606	5.21