mmap

crazy *big*, crazy *fast*, crazy *flexible*

Presented July 25, 2018
CRUG + ChiPy, Chicago IL, USA

Jeffrey A. Ryan jeff.a.ryan@gmail.com

DISCLAIMER

These are my opinions

Only my opinions

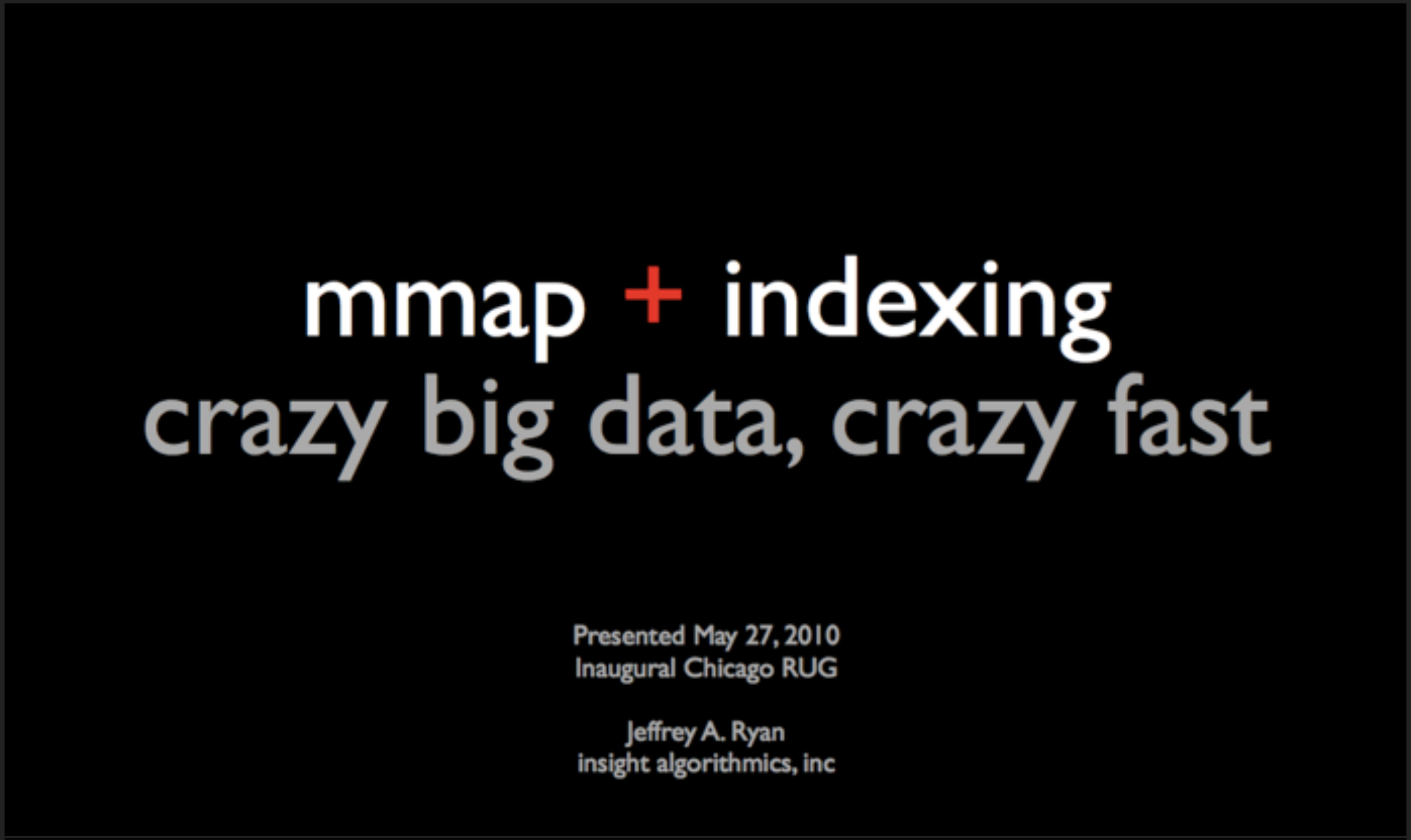
Not my employer's opinions

Not friend's opinions (especially not Dirk's :)

No warranty or guarantees

But *maybe* this will be useful.

Maybe you've seen this already?



mmap + indexing
crazy big data, crazy fast

Presented May 27, 2010
Inaugural Chicago RUG

Jeffrey A. Ryan
insight algorithmics, inc

Background

R and Python since 2003

Co-founder of R/Finance Conference 2009

Quant in Hedge Fund

Created quantmod, xts, etc.

DataCamp course on xts

What is mmap ?

memory-mapped file (like)

system call (core operating system)

constrained by address space, not memory

fast access - once “mapped” behaves like RAM

Why does this matter ?

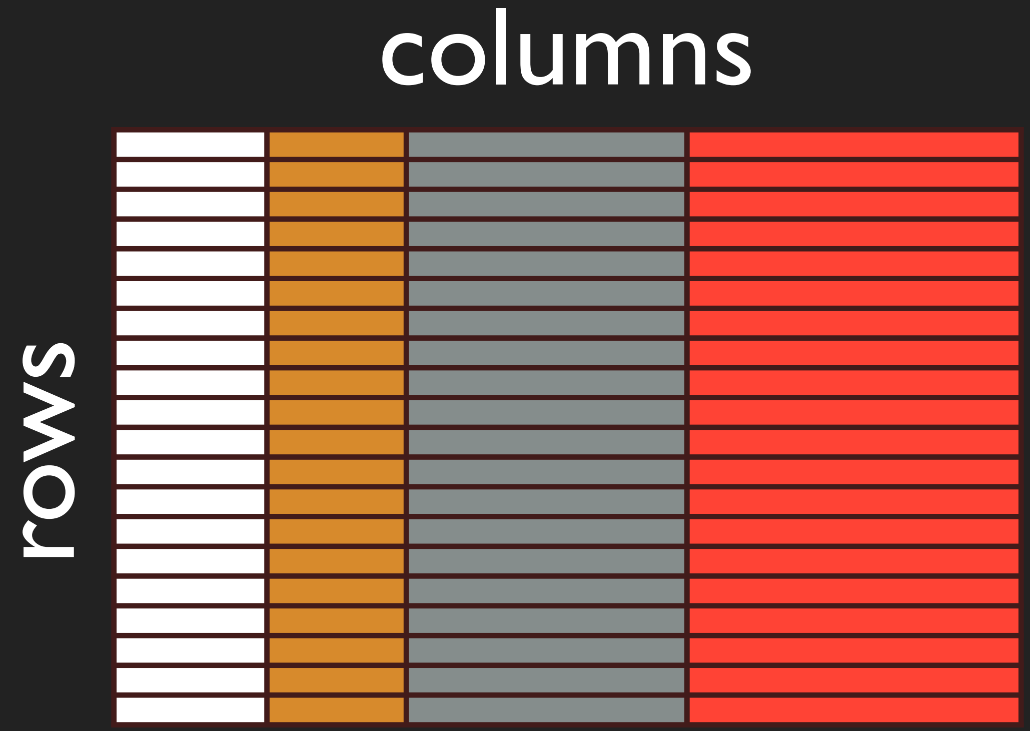
larger than memory datasets

language agnostic - just bytes

cross language sharing is easy (R + Python)

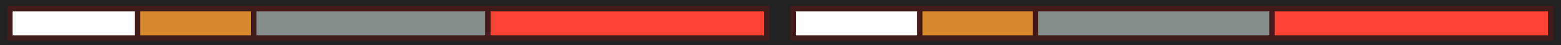
Isn’t this like a database ? YES!

Database Design 101

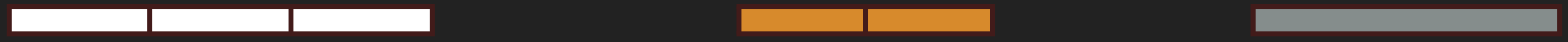


Row or Column Based

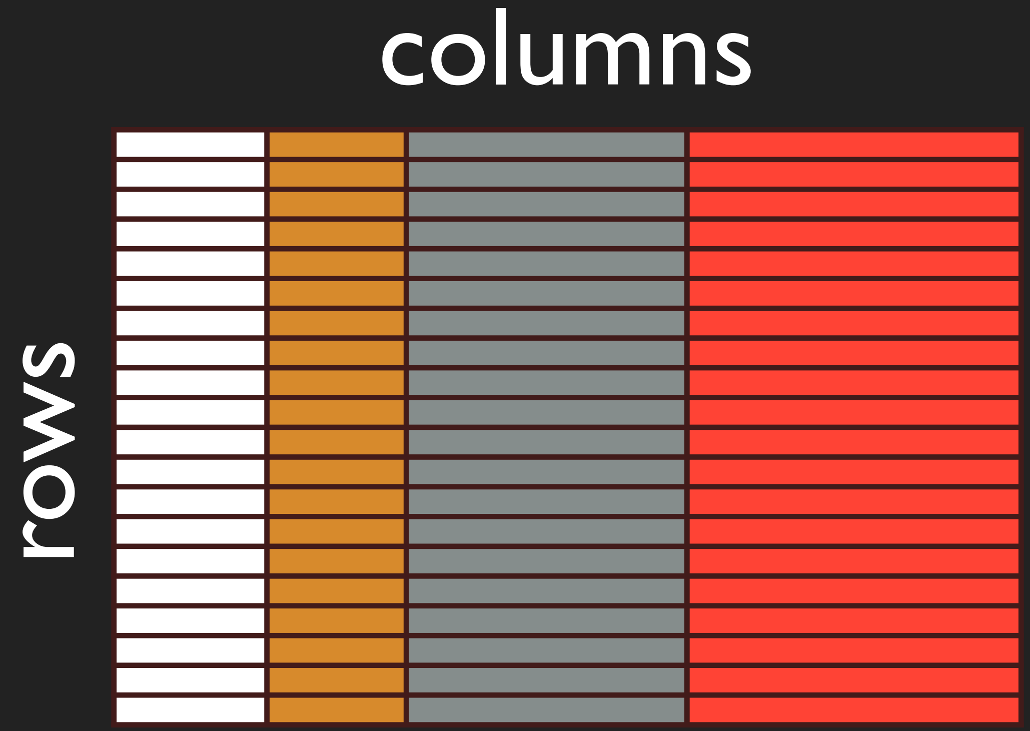
By Row



By Column



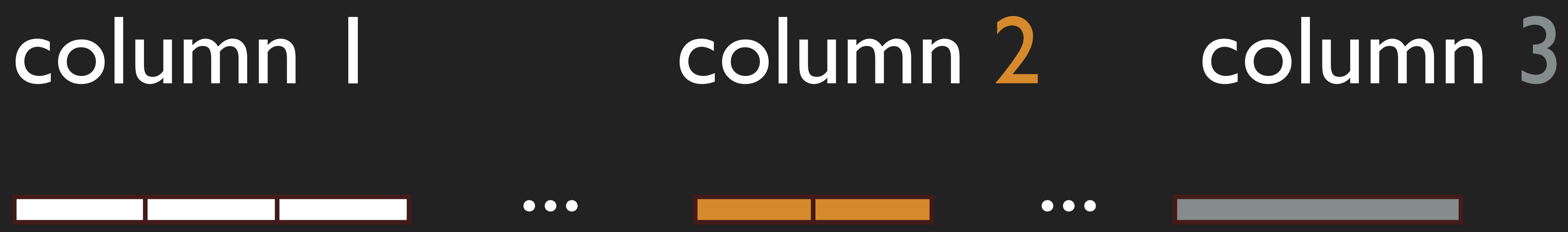
R is Column Oriented



i.e.

data.frames store column values sequentially

a.k.a column-major



Why mmap ?

R is memory limited

Need memory many times data size

Access isn’t always columnar

Maybe you access all row values together ?

Sharing

Pipeline requires different tools - language agnostic serialization!

Random access

Data access isn’t always linear

mmap package

OS system call

very low level API - you see what the C call sees

virtually map files into memory on demand

mapped files are treated *as if* they are memory

cross-platform - Windows and UNIX-alikes

mmap similar (but different) to the R packages *ff* and *bigmemory*

mmap package

mmap	R	C	bytes
raw()	raw	unsigned char	1
bits()	integer	int	1/32
char()	raw	char	1
uchar()	raw	unsigned char	1
int8()	integer	signed char	1
uint8()	integer	unsigned char	1
int16()	integer	signed short	2
uint16()	integer	unsigned short	2
int24()	integer	three byte int	3
uint24()	integer	unsigned three byte int	3
int32()	integer	int	4
integer()	integer	int	4
real32()	double	single precision float	4
real64()	double	double precision float	8
double()	double	double precision float	8
cplx()	complex	complex	16
complex()	complex	complex	16
char(n)	character	fixed-width ascii	n+1
char(n,nul=F)	character	non-nul terminated	n
character(n)	character	fixed-width ascii	n+1
struct(...)	list	struct of above types	variable

Big Endian Support as of Summer 2018 !

mmap package

```
> # 4-byte (int32 or integer)
> # 8-byte (int64)
> # 8-byte float (real64 or double)

> m <- mmap(tmp, int32())
> m
<mmap:/var/folde...>  (int) int [1:10000000] 1 2 3 4 5 6 …

> m <- mmap(tmp, int64())
> m
<mmap:/var/folde...>  (int64) num [1:10000000] 1 2 3 4 5 6 …
> file.info(tmp)$size
[1] 8e+07

> m[34]
[1] 34
```

By Vector

mmap package

```
> # 2-byte (int16)
> # 4-byte (int32 or integer)
> # 8-byte float (real64 or double)

> record.type <- struct(short=int16(),int=int32(),double=real64())
> record.type

struct: (short) integer(0)
           (int) integer(0)
           (double) double(0)

> nbytes(record.type) # 14 bytes in total
   [1] 14

> m <- mmap(tmp, record.type)
> m[1]
$short
   [1] 1
$int
   [1] 366214
$double
   [1] -1.382365
```

By Row

mmap package

```
> example(mmap)

> example(struct)

> example(types)
```

Further Information

http://cran.r-project.org/web/packages/mmap/index.html

http://past.rinfinance.com/agenda/2012/talk/JeffRyan.pdf

mmap + indexing

Real World Example

67,836,671 equity option contracts
13 columns, 12GB on disk

```
> system.time( db[symbols=="AAPL"] )
   user  system elapsed
  0.012   0.000   0.012

> db[symbols=="AAPL"]
91428 hits
```

mmap + indexing

Real World Example

6 queries in 3.3 seconds

get a single contract as an xts time-series given OSI key

last 3 days of all AAPL April calls that have a delta at some point between .5 and .8, showing bid,ask,iv, and volume as an xts time-series

number of records on April 13

osi, bid and ask of AAPL puts (delta<0) on April 13, expiring on the April 17

same, sorted by decreasing iv, excluding no-bid contracts, limit to 15

plot 3 day EMA of bid-ask spread of AAPL options with IV between 20% and 30%