Visualization in Testing a Volatile Memory Forensic Tool

By

Hajime Inoue, Frank Adelstein and Robert Joyce

Presented At

The Digital Forensic Research Conference

DFRWS 2011 USA New Orleans, LA (Aug 1st - 3rd)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http:/dfrws.org

Digital Forensic Research Workshop (DFRWS 2011)
August 1st, 2011

Visualization in Testing a Volatile Memory Forensic Tool

Hajime (Jim) Inoue
hinoue (at) atc-nycorp.com

Frank Adelstein
Robert Joyce

ATC-NY
ARCHITECTURE TECHNOLOGY CORPORATION

Follow Along

- Paper on page S42
- Flickr: http://doiop.com/dfrws2011

Volatile Memory Analysis

- Study growing rapidly since 2005 DFRWS memory challenge
- String search/carving
- Tools such as PTFinder, Volatility allow reconstruction (for Windows)
- Growing disk sizes, use of encryption motivate its increasingly important role
- No tool on Macs since OS X 10.4

Current Imaging Techniques

- Hardware
  - Cards
  - Firewire
- Software
  - Windows: mdd, win32dd, dumpmmf
  - Linux: /dev/mem, /dev/crash
    - Broken by design
  - OS X: 10.0-10.4 (PPC)
    - Broken by design
    - kmem=1 will create broken /dev/kmem, /dev/mem

Mac Memory Reader (mmr) on OS X

- Implemented as a Kernel Extension (kext)
- Supports >= 10.4 (ppc, i386, x64)
- Creates two devices
  - `/dev/pmap` – physical memory map
    - same format as `showbootermemorymap` debug macro
  - `/dev/mem` – physical memory device
- Userspace utility outputs as Mach-o dump file
  - dd also supported

Memory Imager Properties

4 metrics for evaluation

1. Speed – Copy memory fast for consistency
2. Completeness – Copy all of physical memory
3. Accuracy – Copy memory accurately
4. Non-Interference – Don't overwrite memory

Testing our /dev/mem

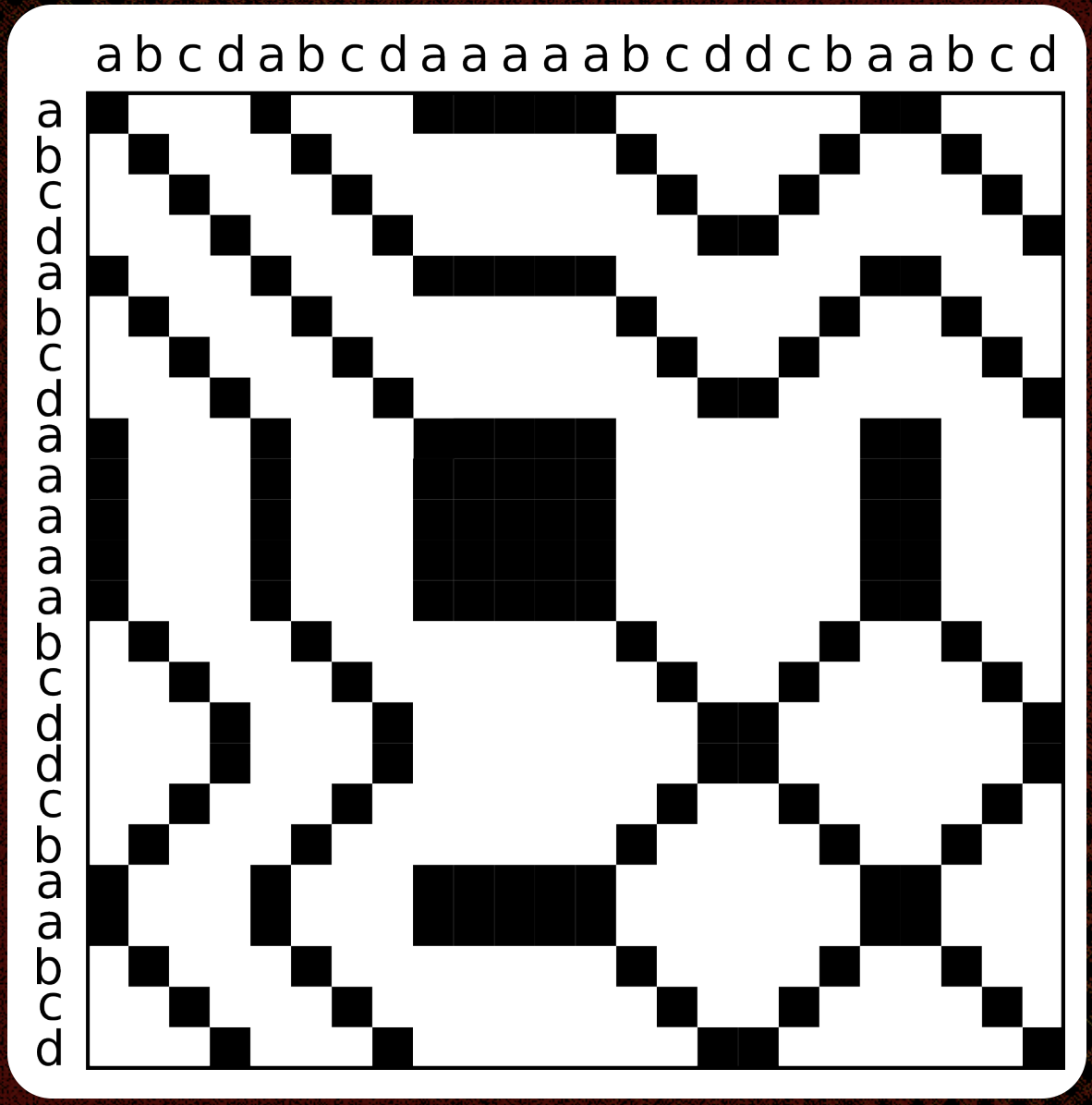
comparison against dd

1. Speed: 14.2 secs (mmr) vs 16.4 secs (dd)
2. Completeness: 131072 (mmr) vs 131072 (dd)
3. Accuracy: 93% similar (using netcat)
4. Non-interference:
   - Average similarity after 5 runs
     - mmr: 97%
     - dd: 32%

Testing our /dev/mem
Finding our bug

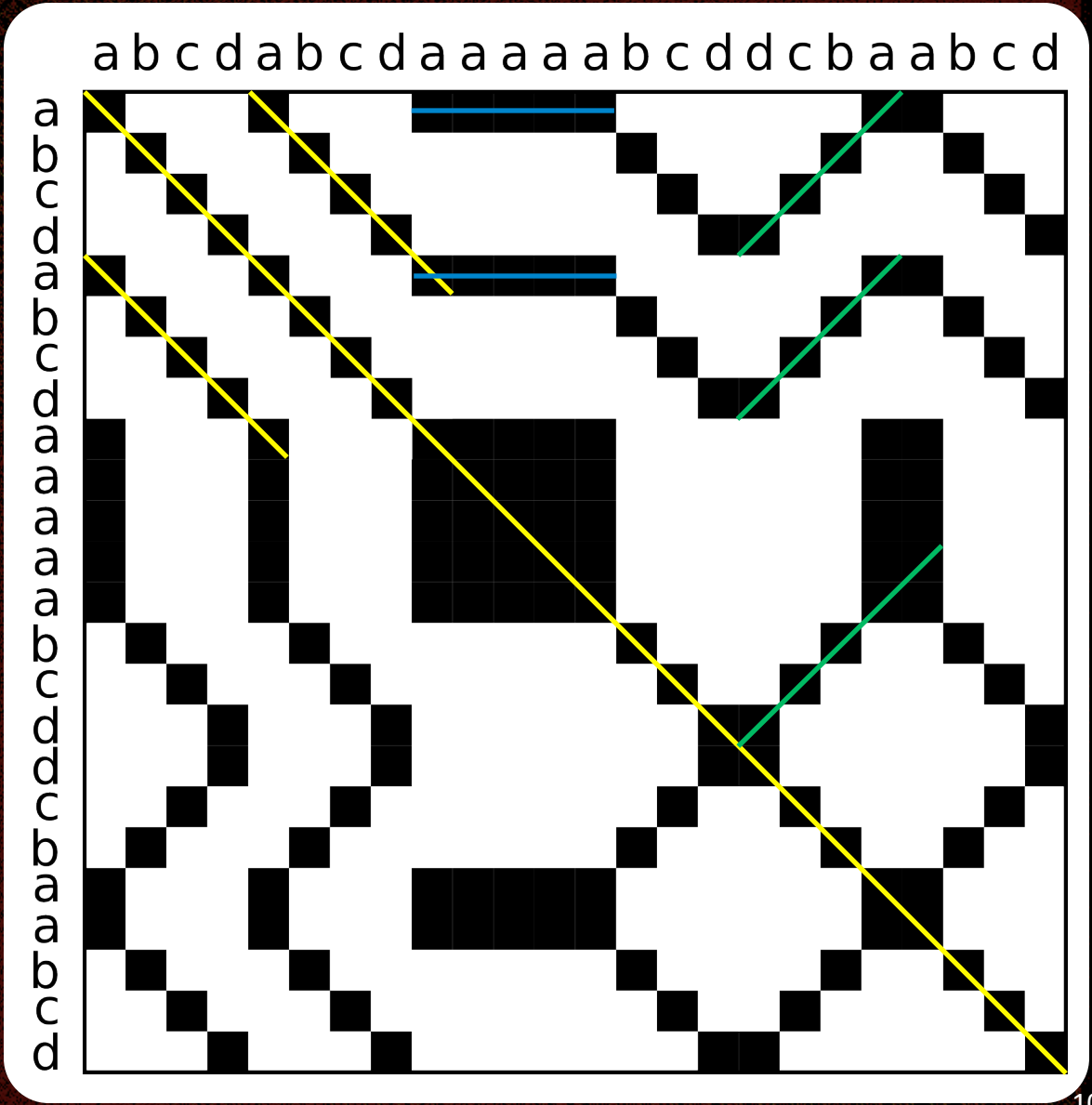
- Using debug info to check symbols
- String injection
  - Inject kext with known unique string
  - Look for it in image file
  - Multiple copies?
    - Disk cache, user space applications, kernel
- String extraction found large identical sequences of pages ☹

Dotplots



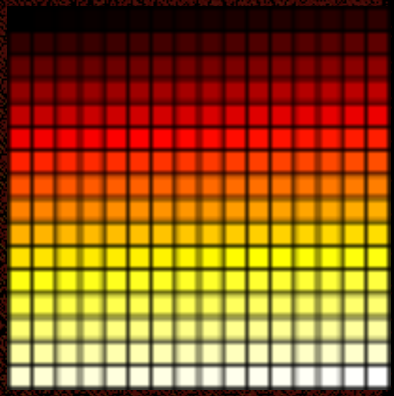
Dotplots

- Forward Similarity
- Retrograde Similarity
- Duplication

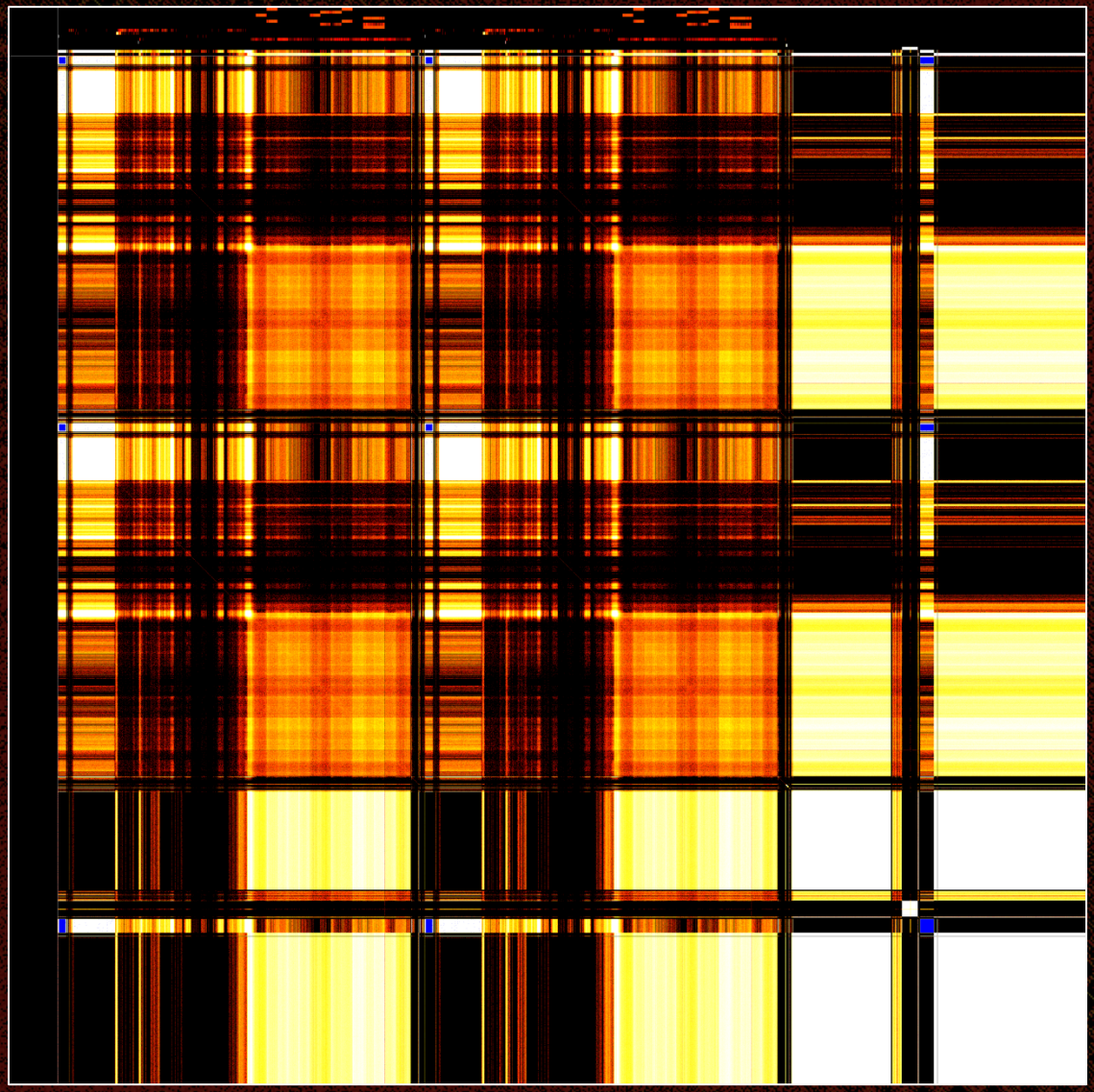


Physical Memory Dotplots

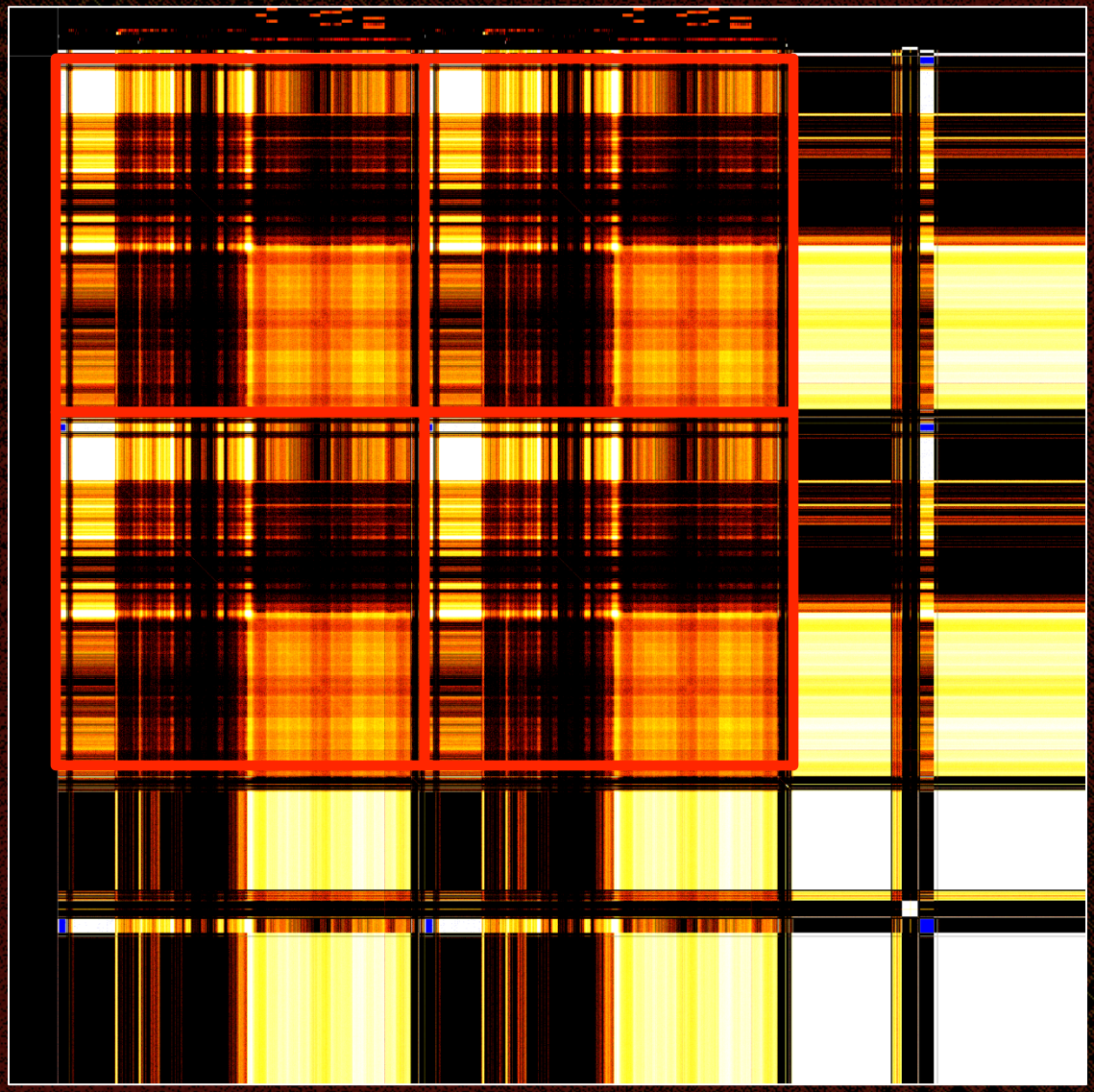
- Each page (hash) is a symbol
- Many pages per pixel (page window)
- Blackbody radiation palette
  - Black – no similarity
  - White – all pages identical
  - Blue – all pages are the zero page
- Sampling reduces runtime



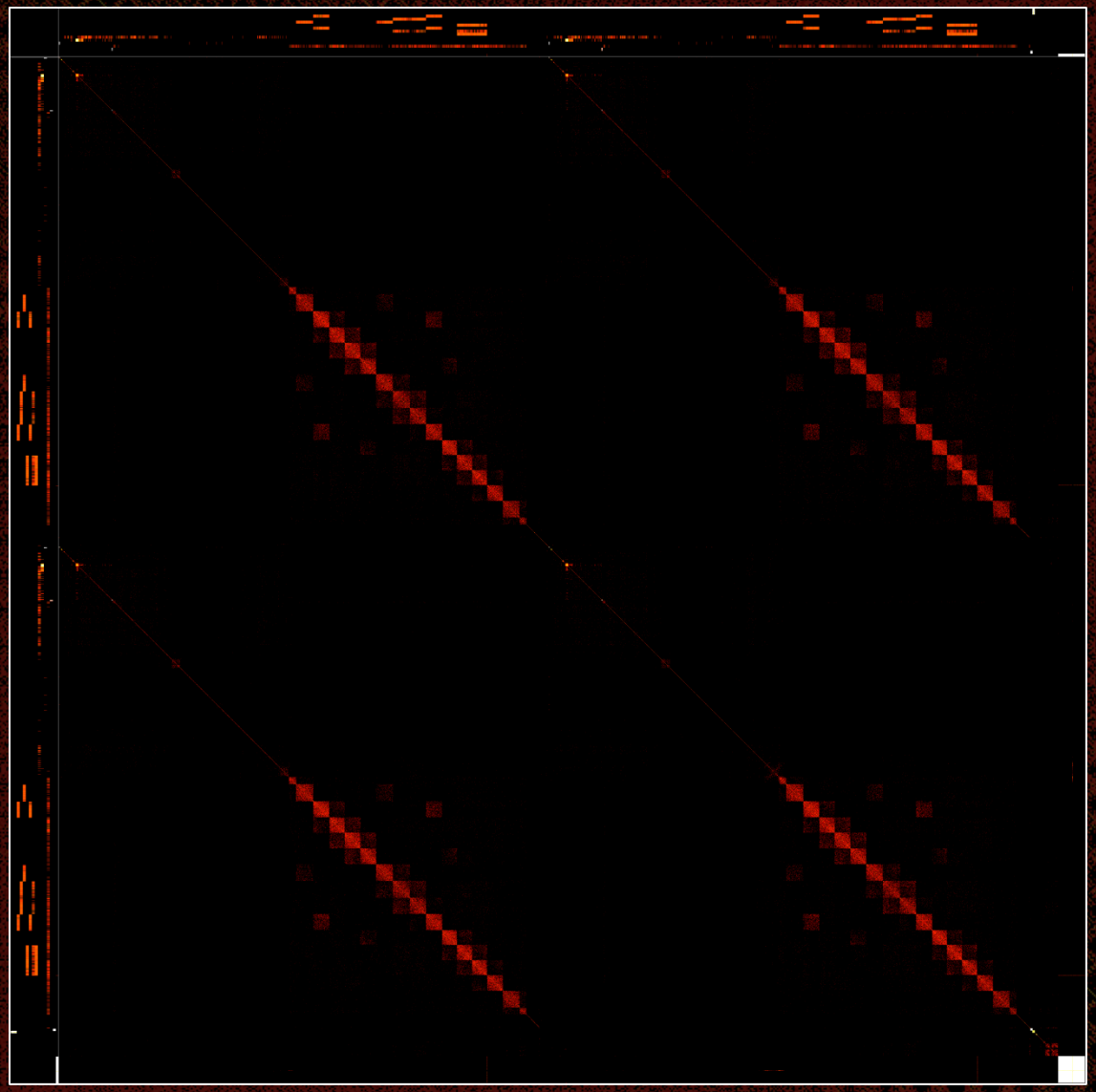
Visualizing our bug



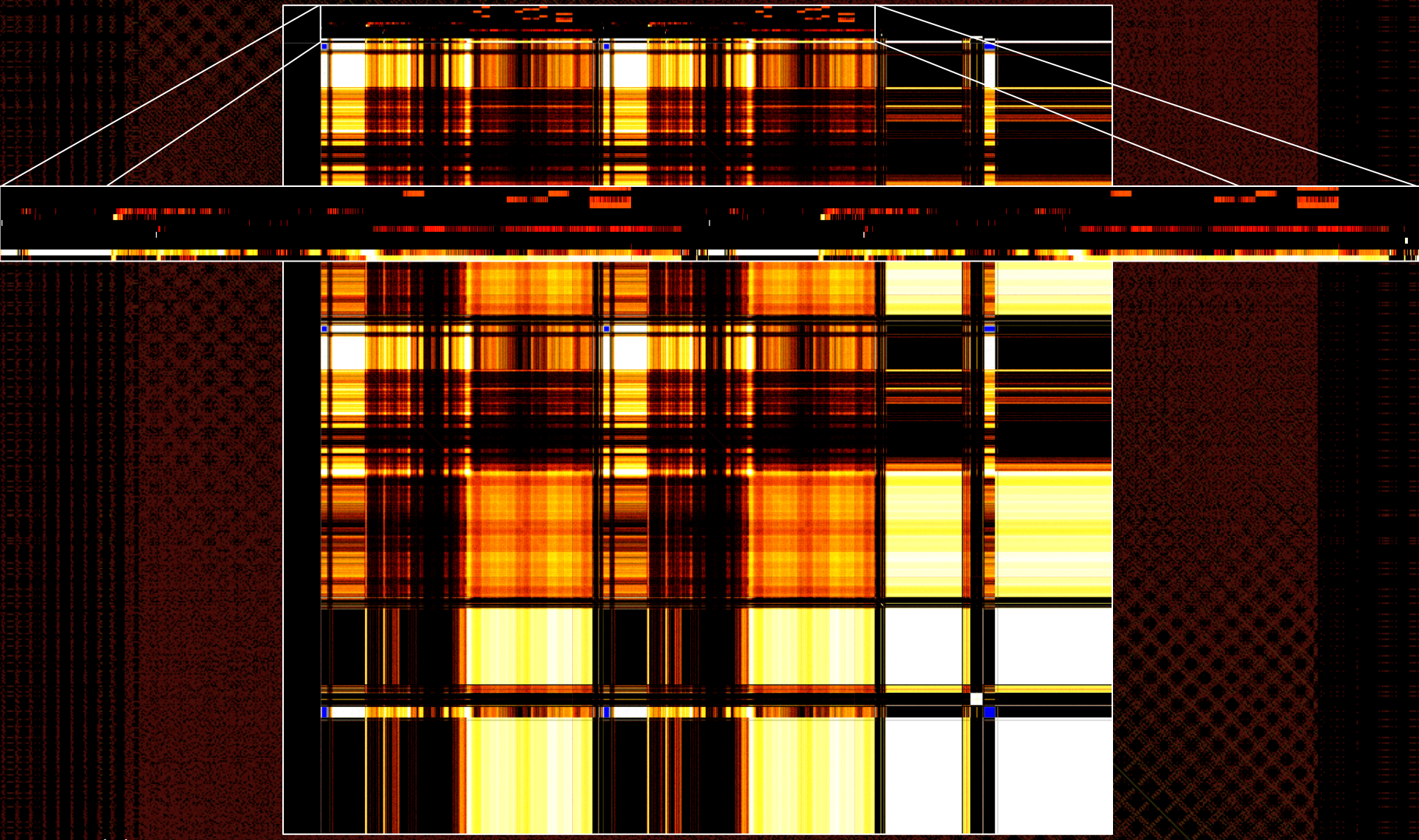
Visualizing our bug



Visualizing our bug: No Zero Pages

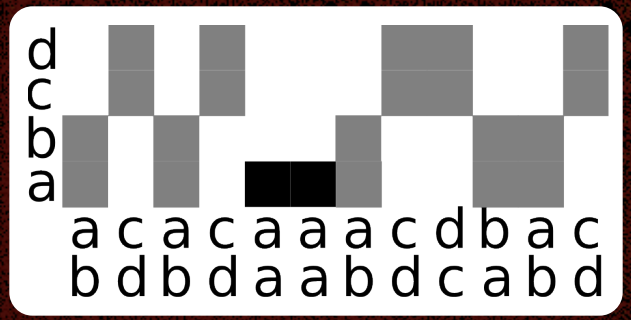


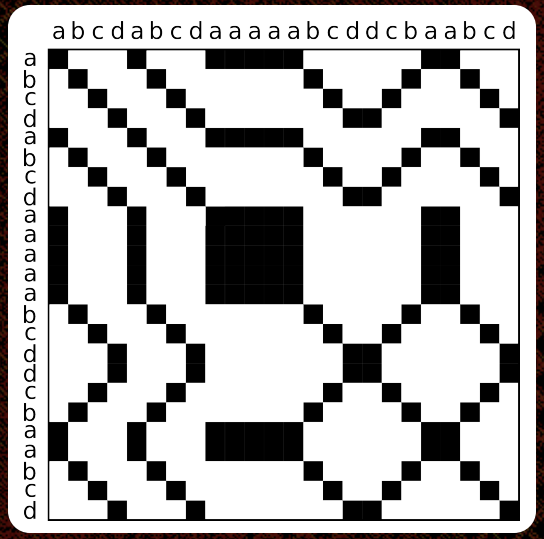
Visualizing our bug: Density Plots



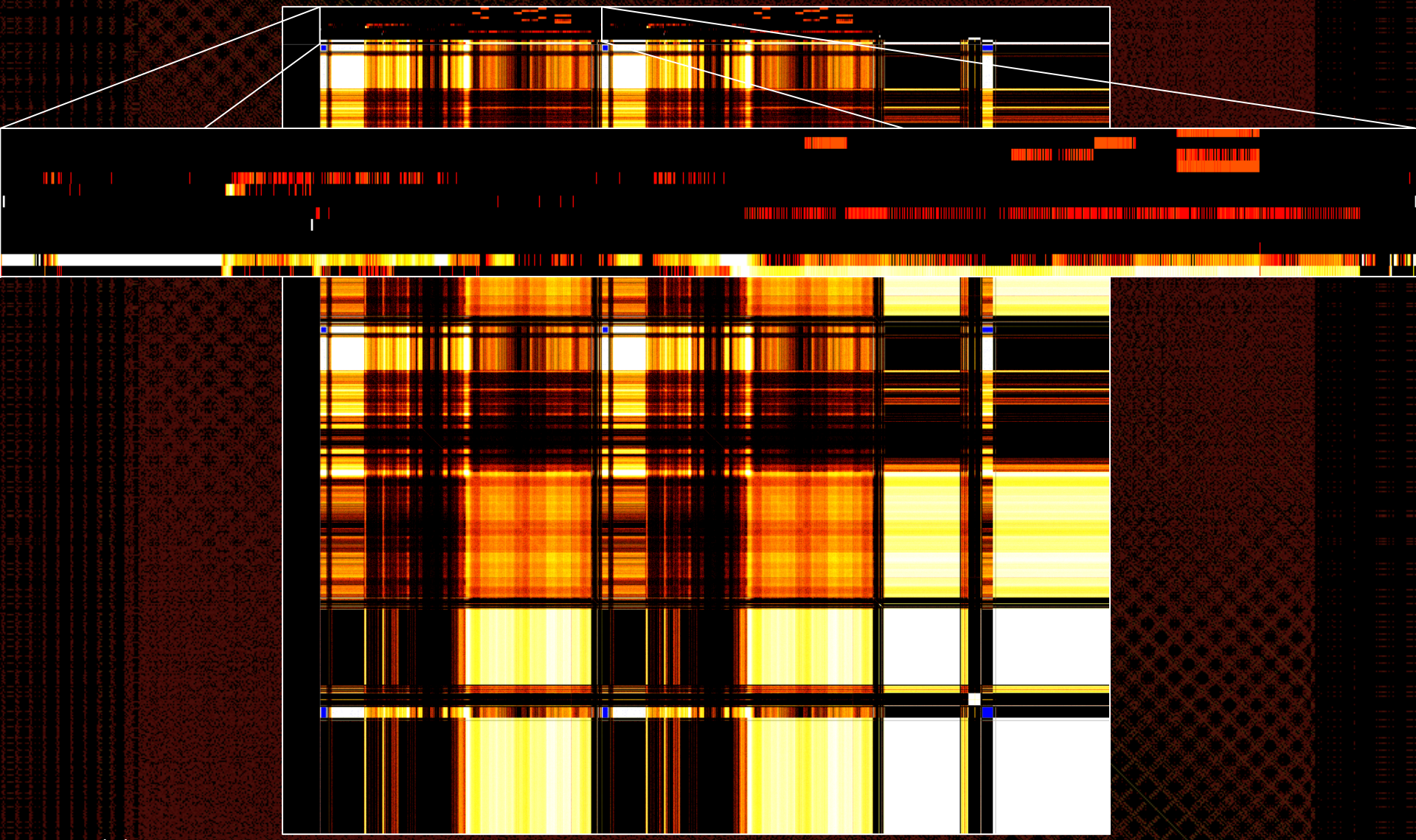
Density Plots

- Calculate ordered list of page frequencies
- Plot density of *n* most frequent (we chose *n*=8) against each page window.

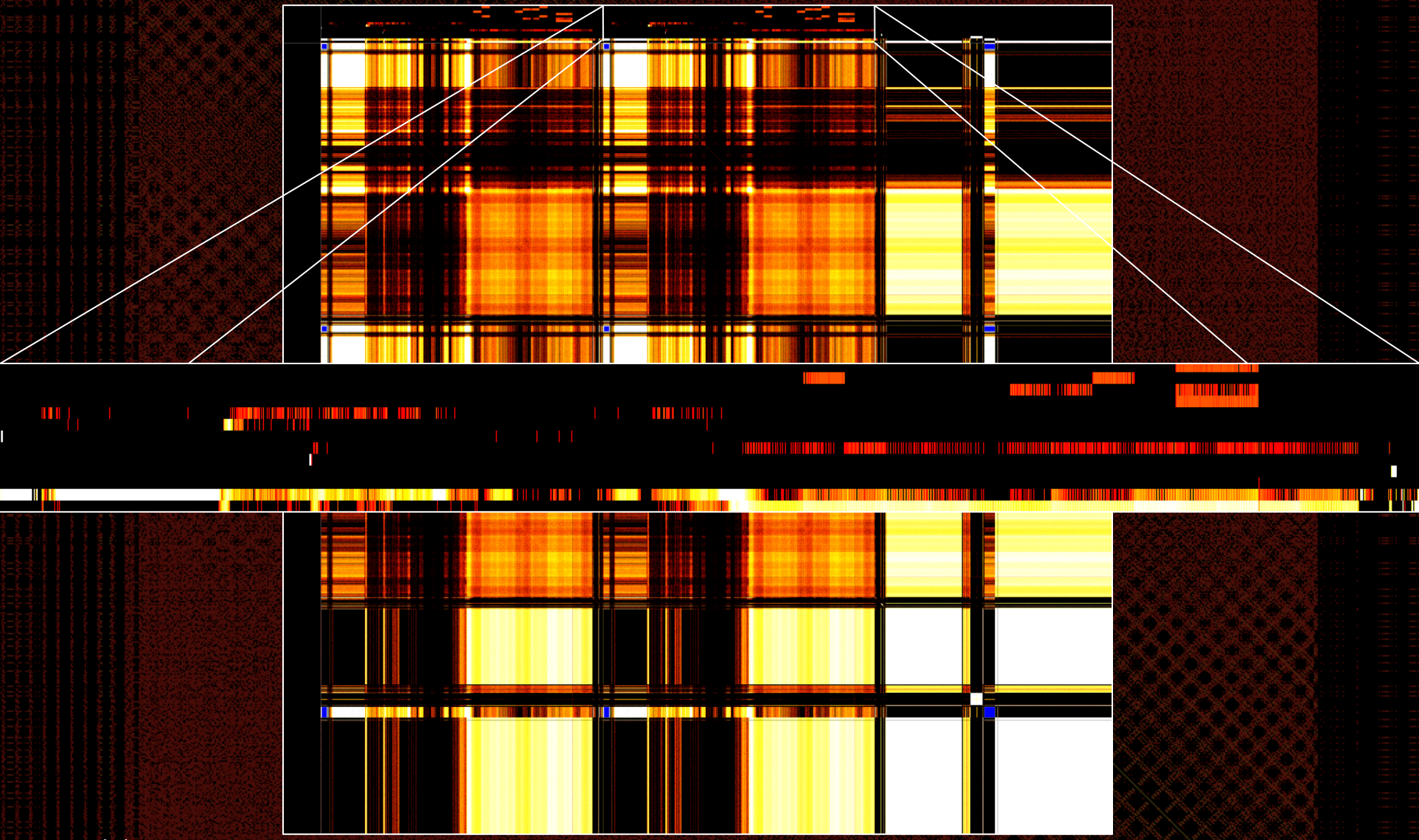




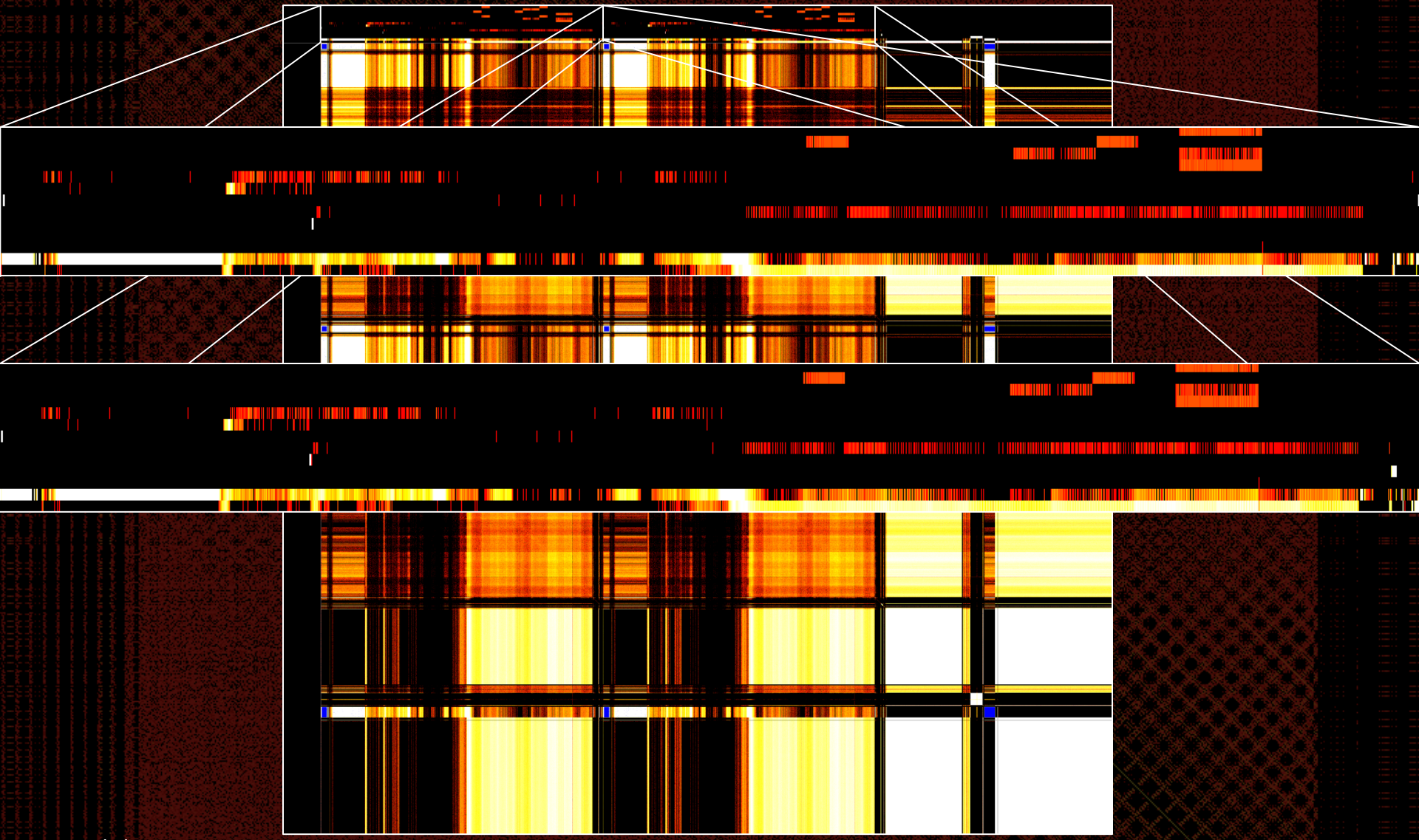
Visualizing our bug: Density Plots



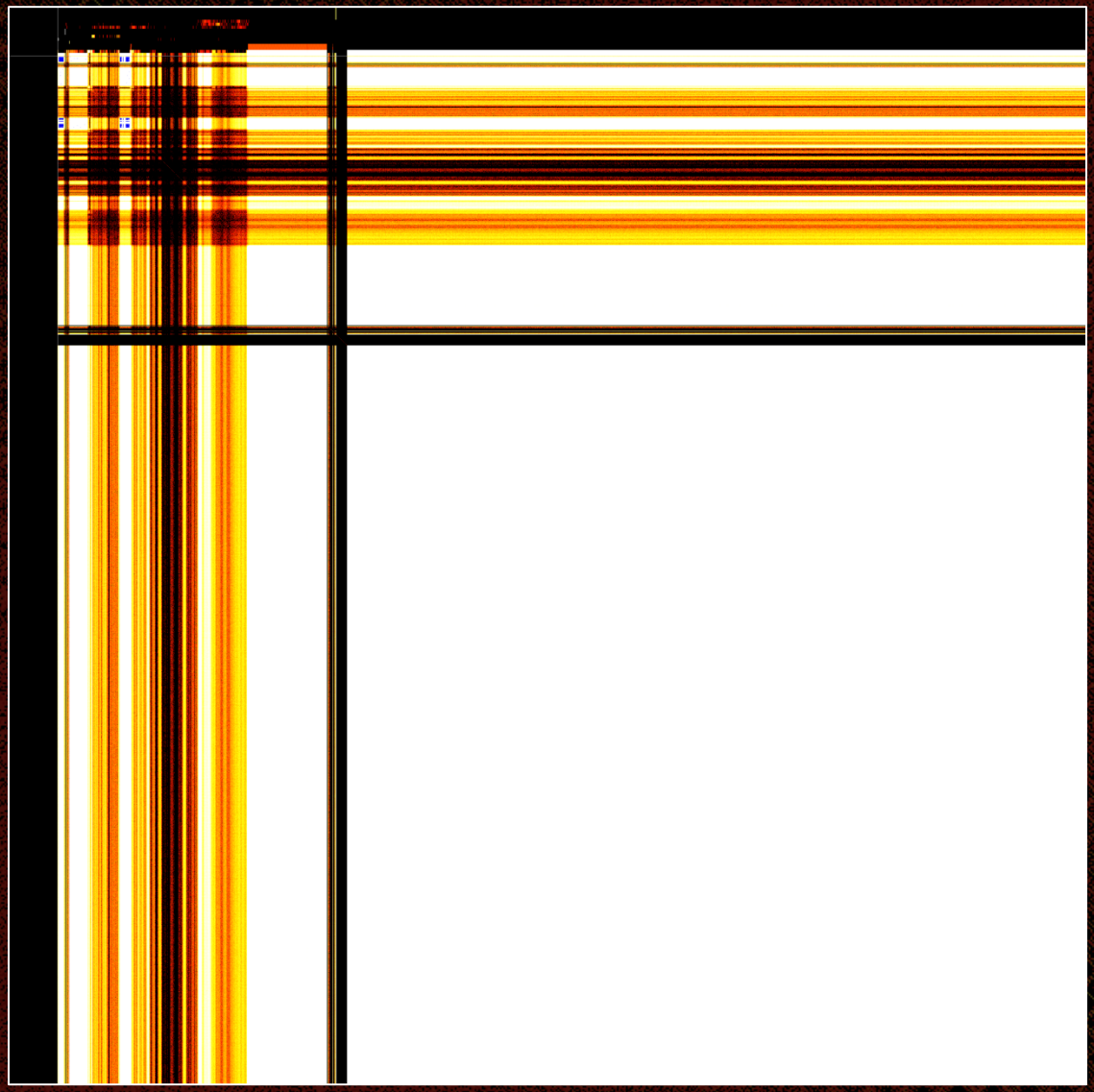
Visualizing our bug: Density Plots



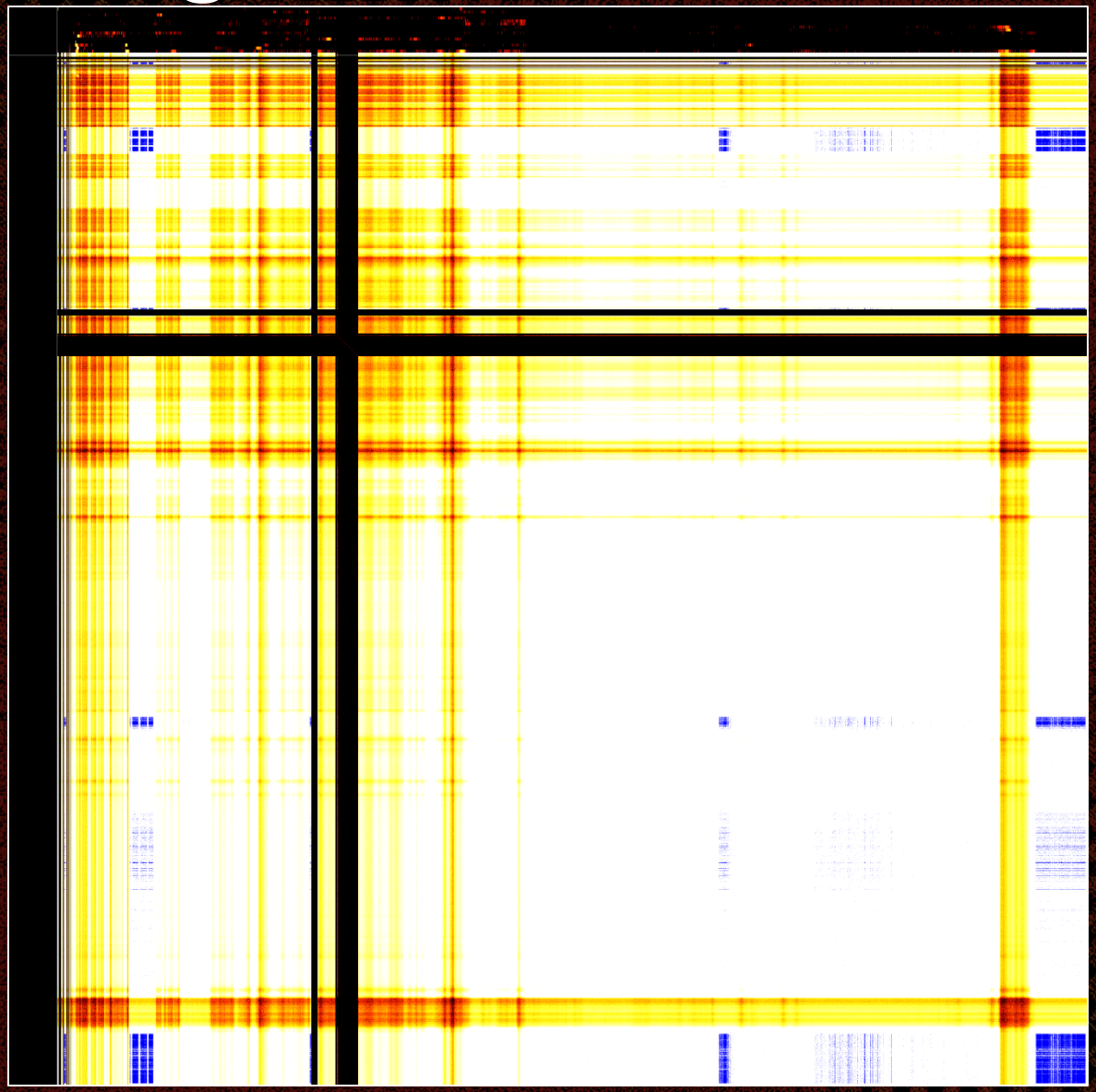
Visualizing our bug: Density Plots



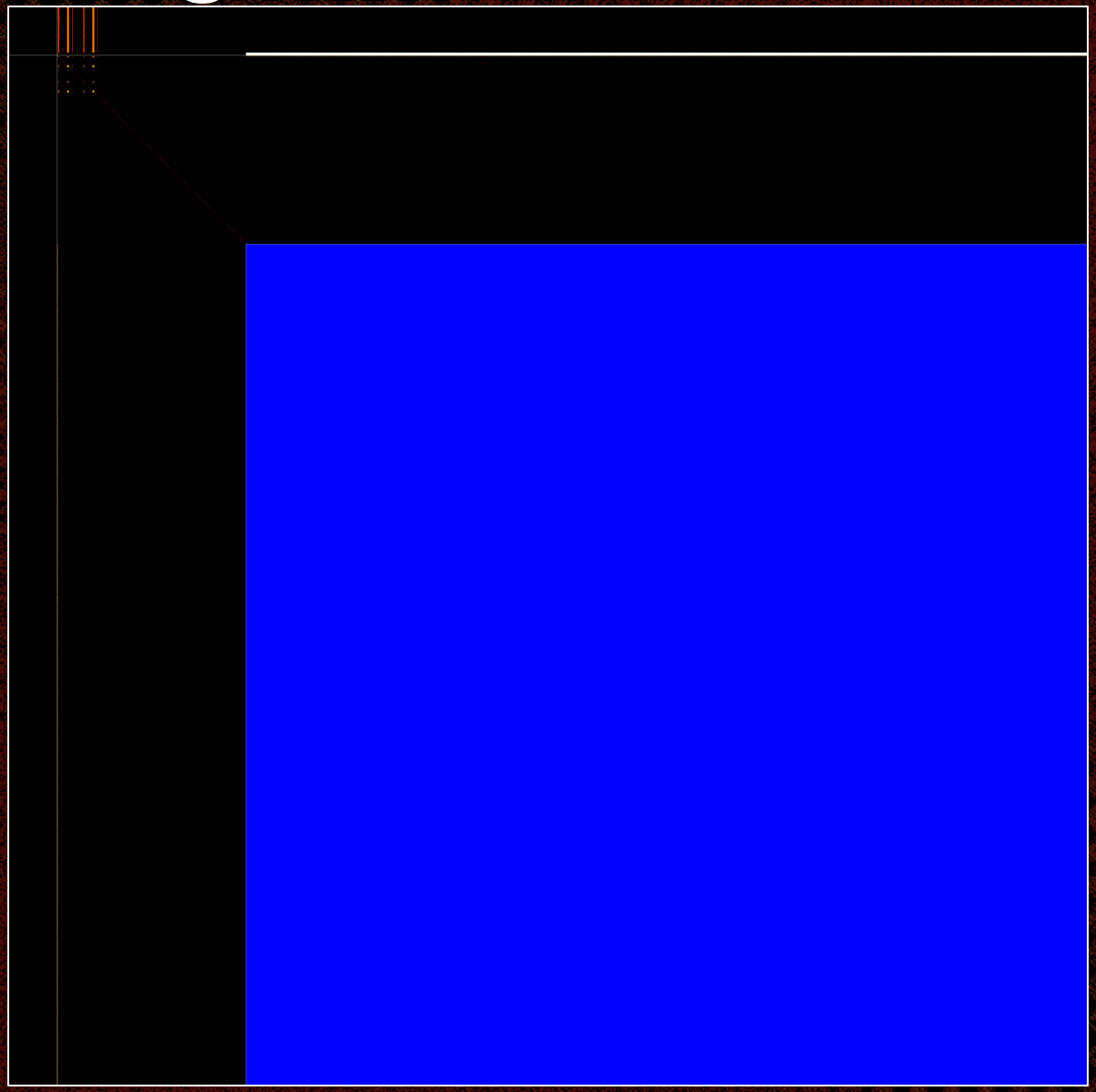
Visualizing our bug: Non-Buggy Plot



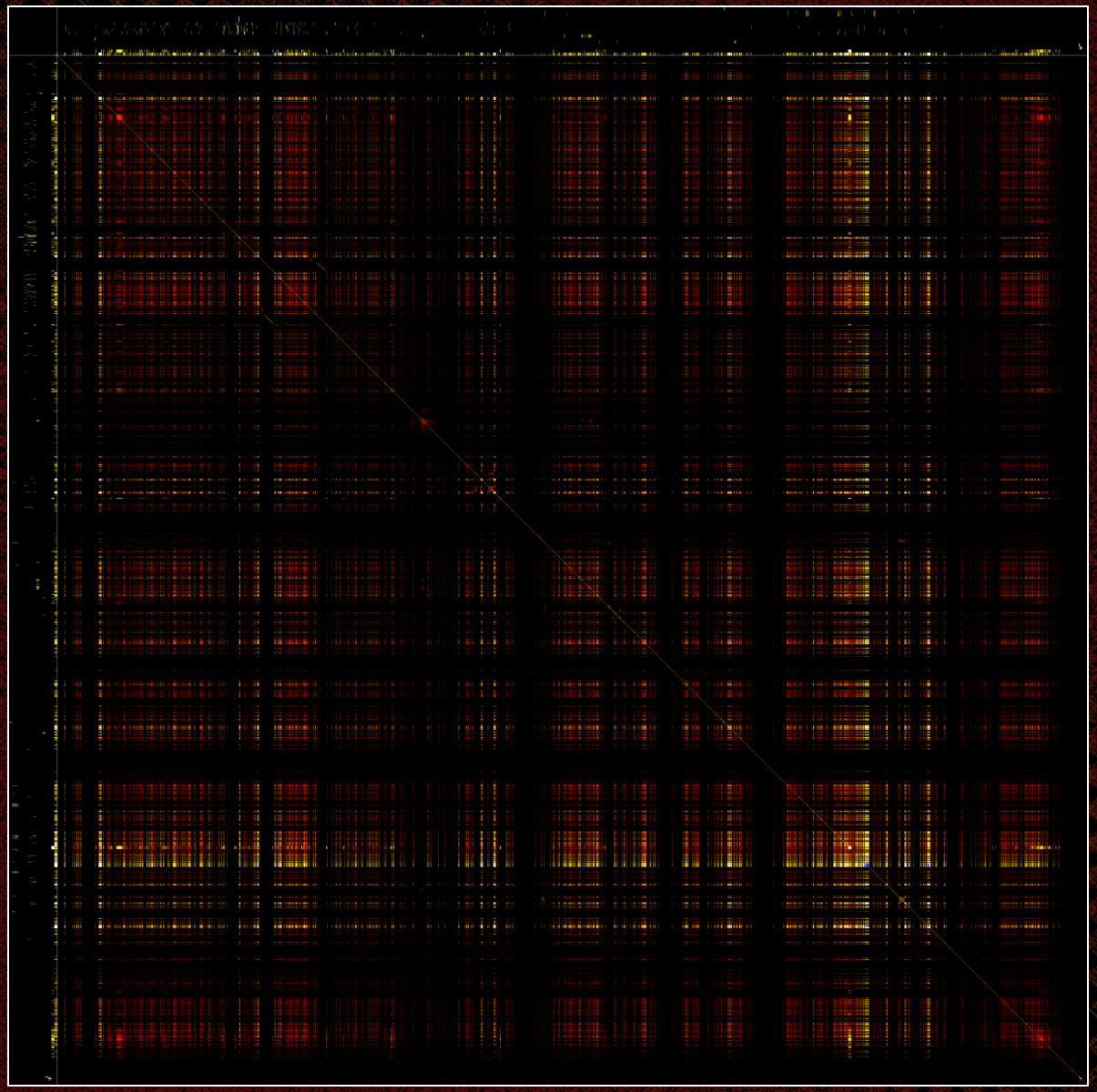
Using the Hibernation File



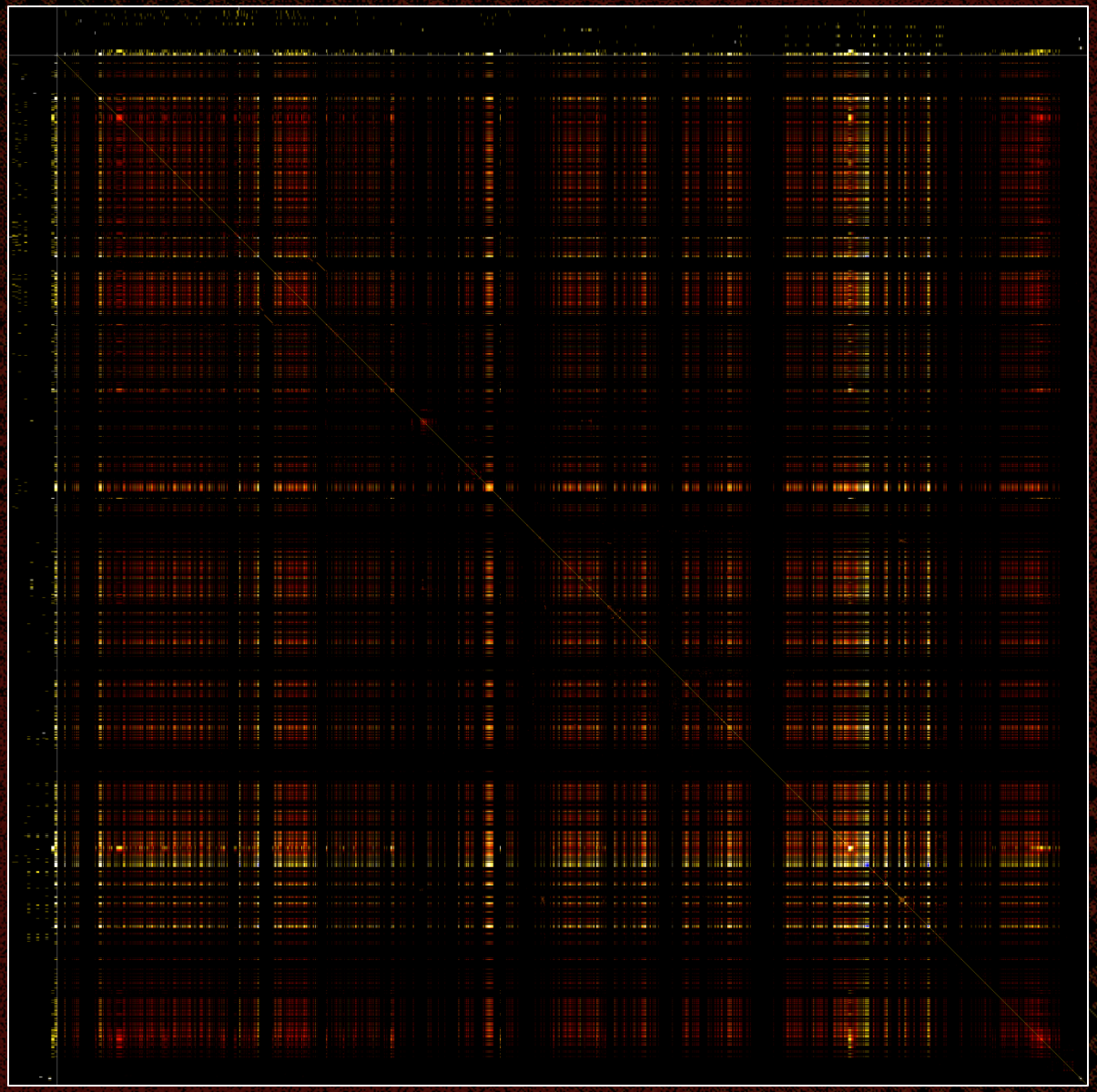
Using the Hibernation File



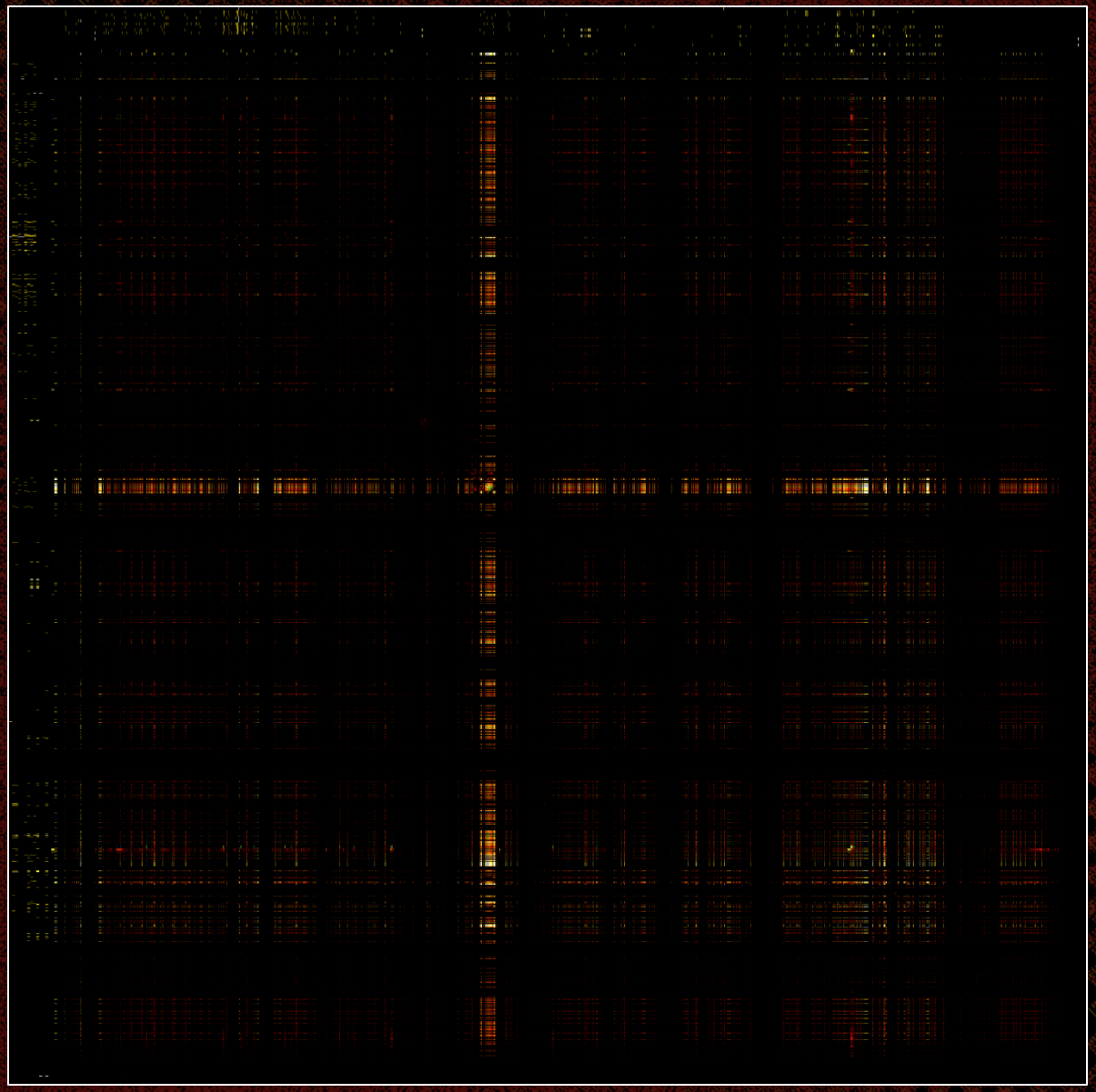
Comparing OS X /dev/mem to MMR



Comparing OS X /dev/mem to MMR

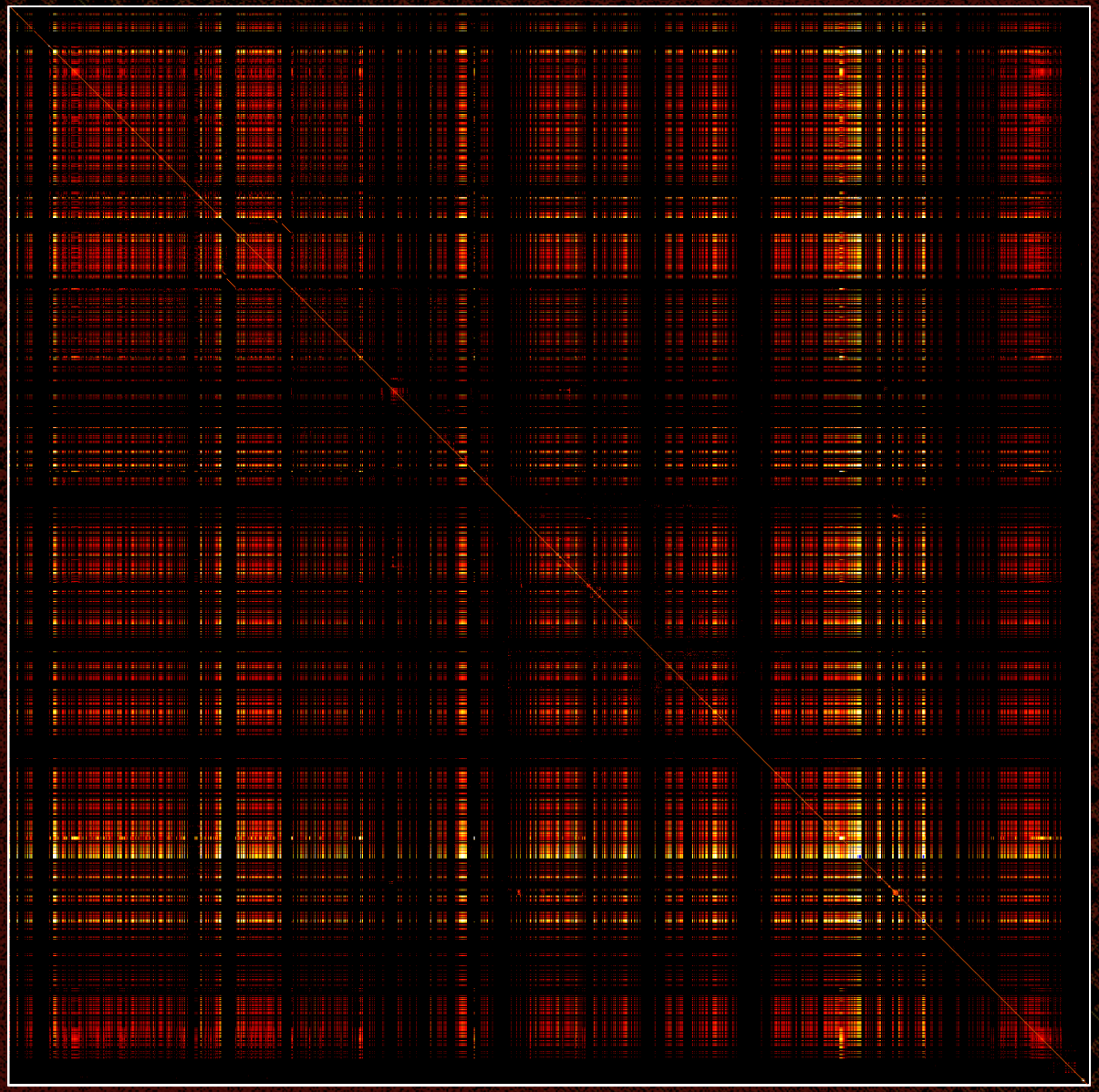


Comparing OS X /dev/mem to MMR

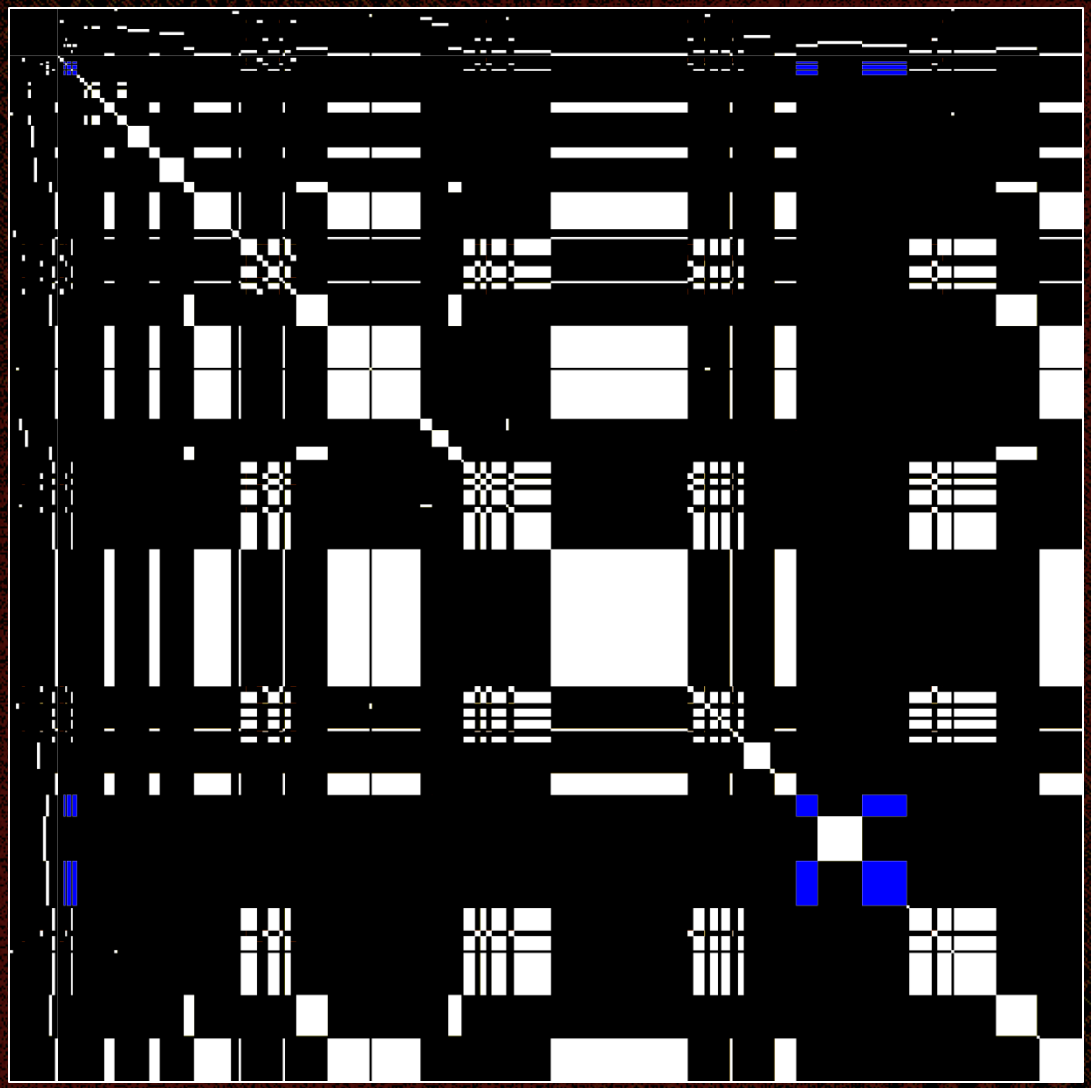


Comparing OS X /dev/mem to MMR

Comparing OS X /dev/mem to MMR

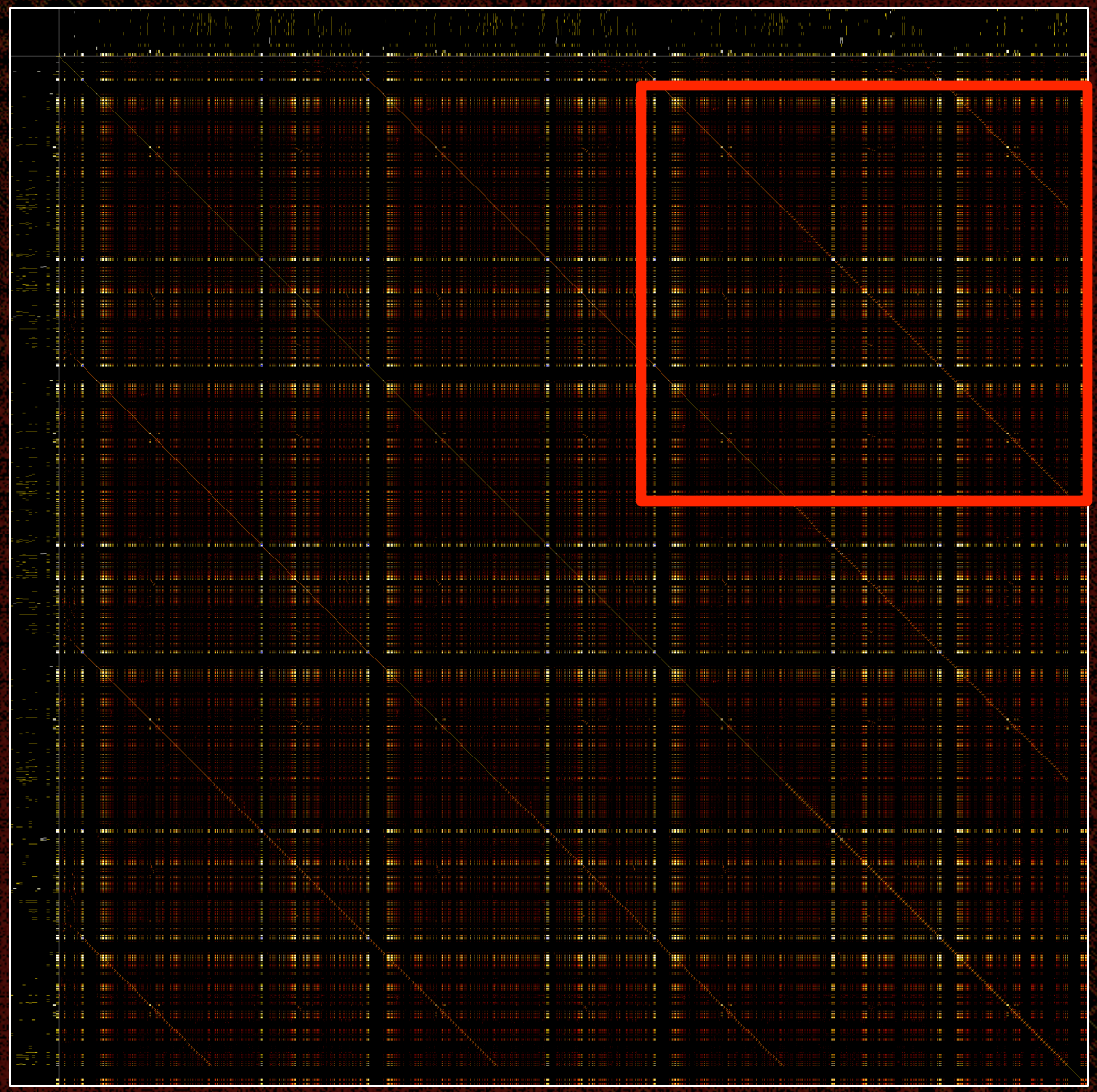


Apple's /dev/mem
(on Intel)

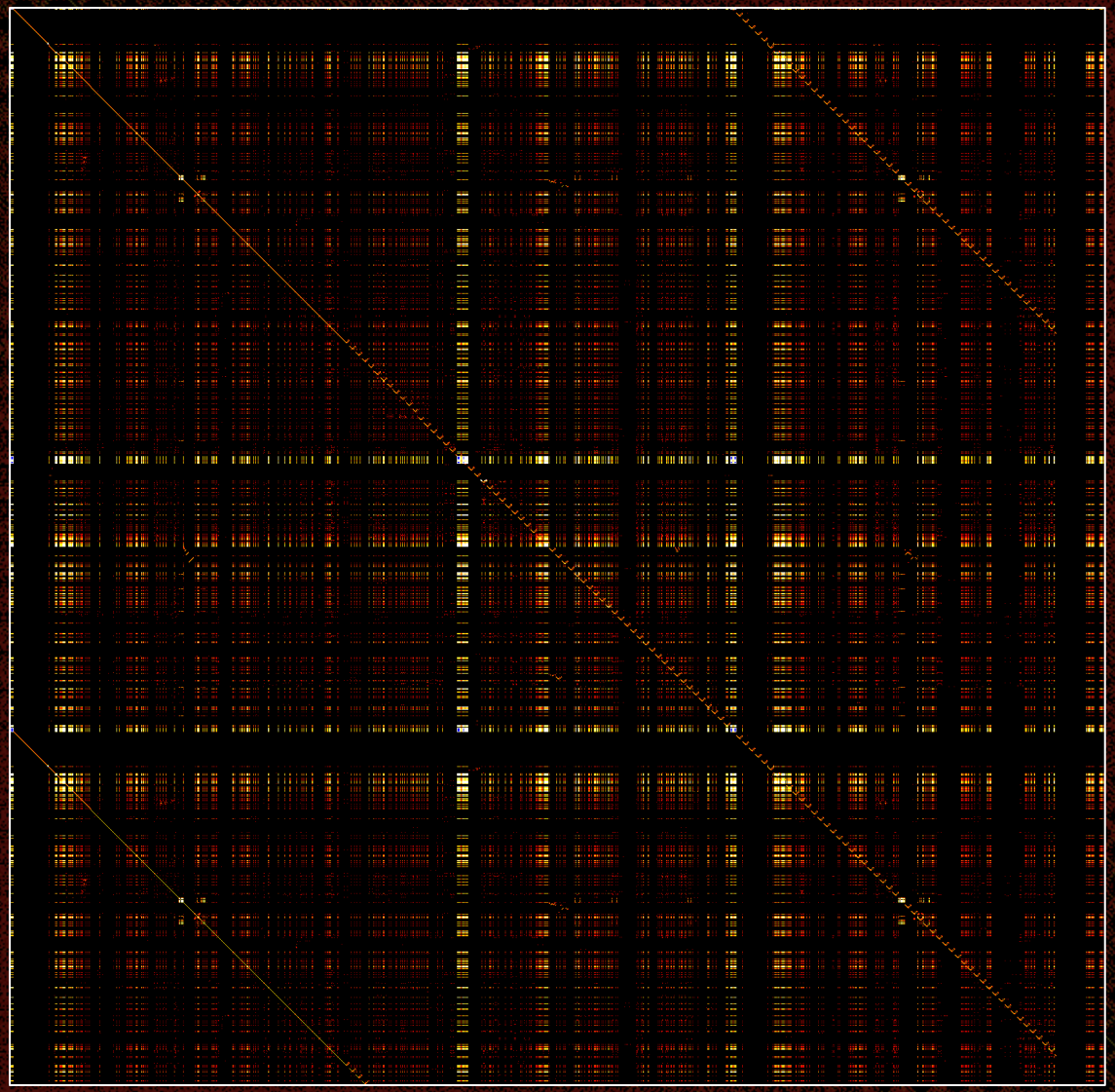


OS X **dd** to Disk

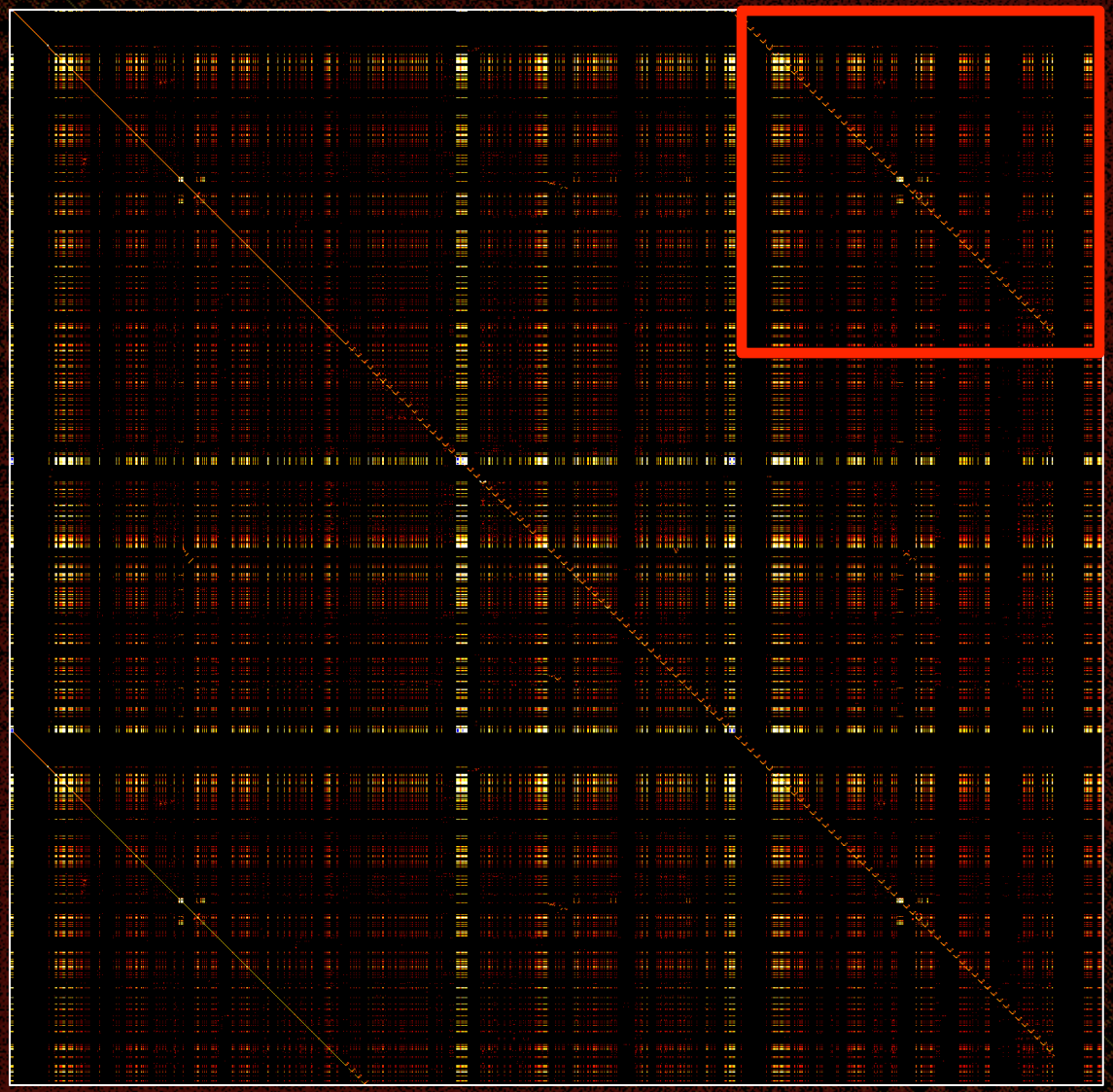
OS X **dd** to Disk



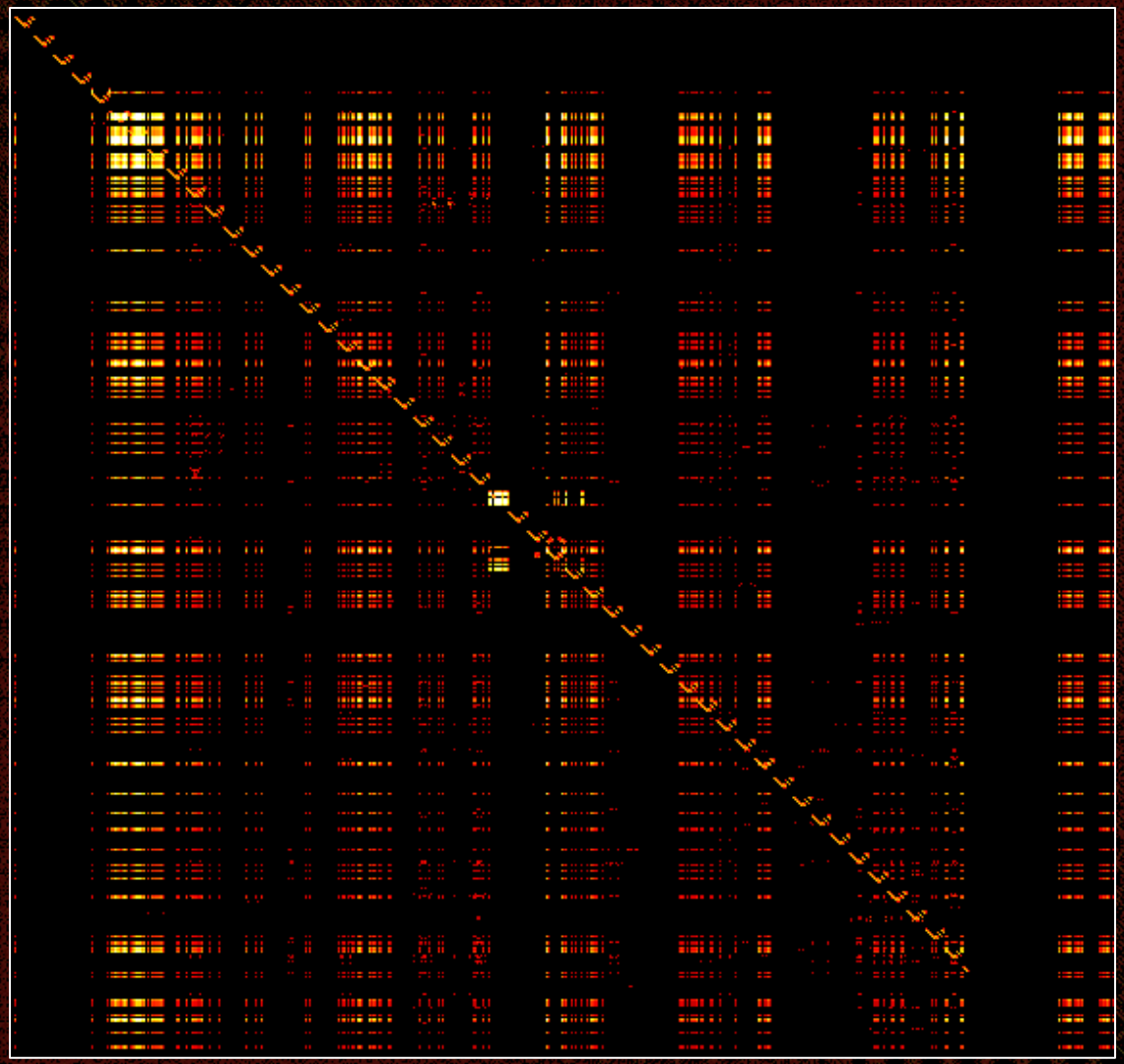
OS X **dd** to Disk



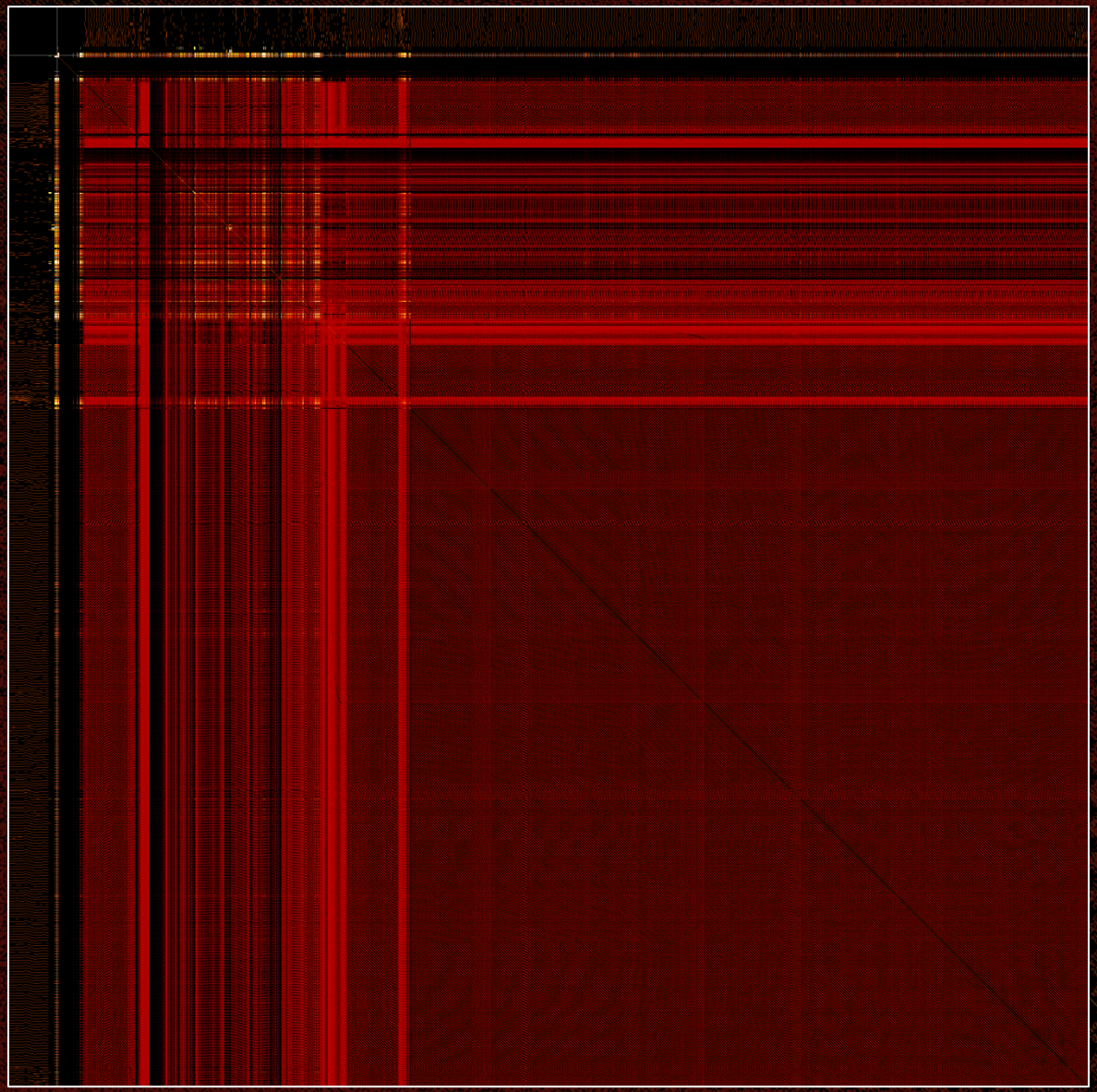
OS X **dd** to Disk



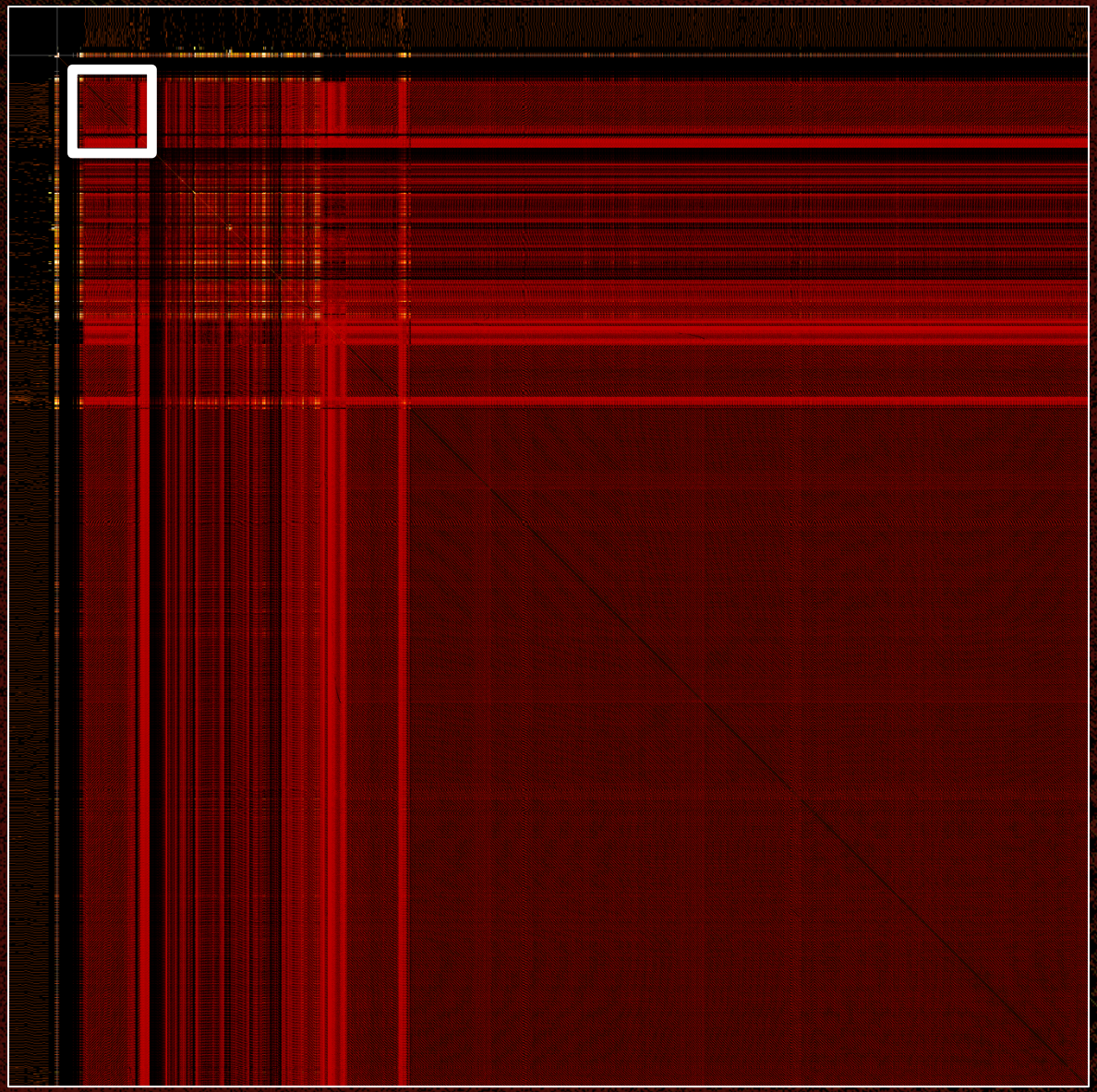
OS X **dd** to Disk



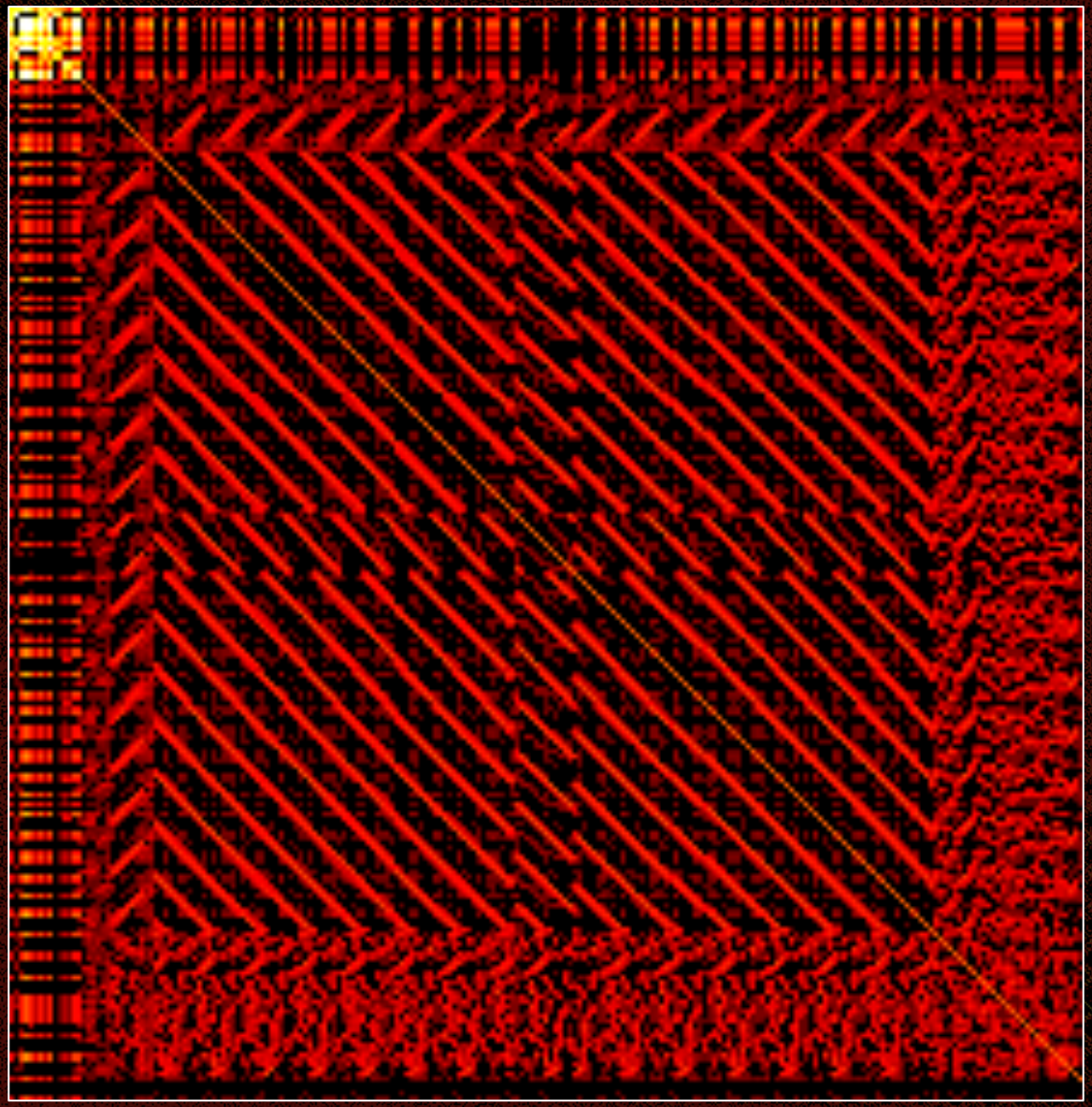
String Injection



String Injection



String Injection



Conclusion

- Implemented a physical memory imager for OS X
- Established metrics for imaging:
  - Speed, completeness, accuracy, non-interference
- With no ground truth, testing is difficult
  - Tools produce different results
- Visualization can help:
  - dotplots
  - density plots
- OS X:
  - Don't record dd to disk
  - Don't rely on the hibernation file
  - Mac memory reader available at: www.cybermarshal.com/index.php/cyber-marshal-utilities/mac-memory-reader

Future Work

- Mac Memory Reader for Lion (OS X 10.7)
  - Released July 28th
- Analysis
  - Matthieu Suiche – Blackhat Paper: Mac OS X Physical Memory Analysis
  - Kyeong-Sik Lee - Volafox
  - Our own tools are forthcoming

Questions

(How was the background image generated?)