https://github.com/tycho-kirchner/shournal

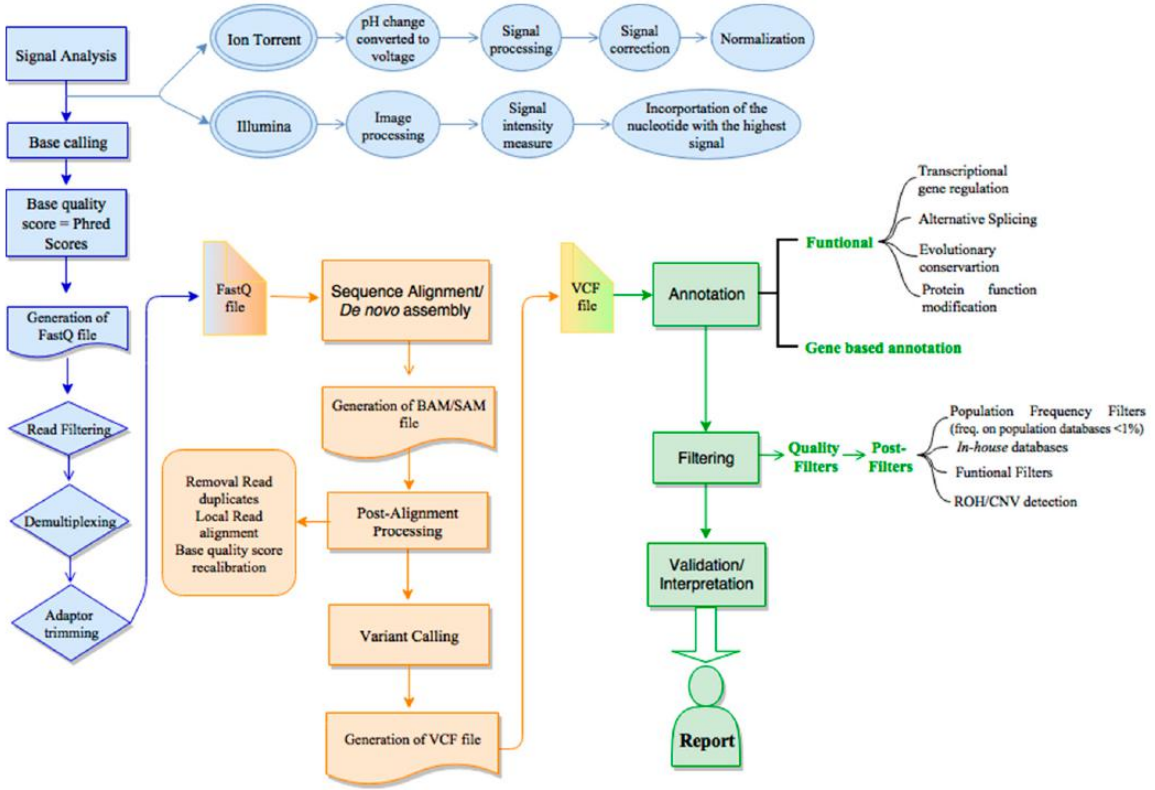
Leibniz Institute on Aging –
Fritz Lipmann Institute

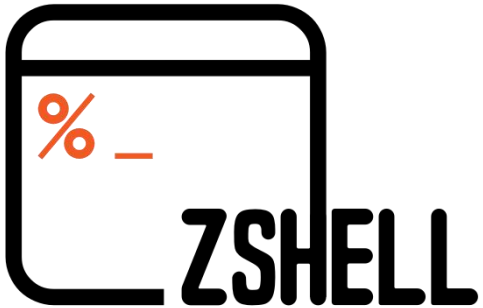
Bashing irreproducibility with shournal

Record shell-commands and used files with the *shell journal*.

Motivation

https://github.com/tycho-kirchner/shournal






```
$ sort ...
$ paste ...
$ ./preprocess.sh
$ ./pipeline.sh
```

Pereira R, Oliveira J, Sousa M. Bioinformatics and Computational Tools for Next-Generation Sequencing Analysis in Clinical Genetics. J Clin Med. 2020 Jan 3;9(1):132. doi: 10.3390/jcm9010132. PMID: 31947757; PMCID: PMC7019349.

Reproducibility?

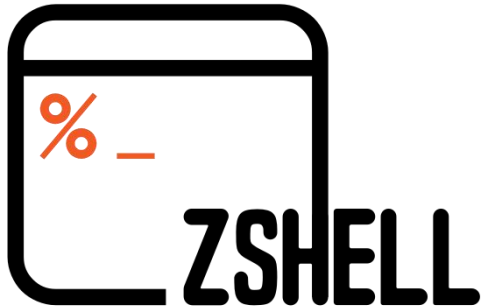
A shell workflow typically involves:

- Executing commands with a plethora of parameters
- Modifying scripts
- Editing configurations files
- (~~Documenting what was done~~)


```
$ bwa mem -t 32 -T 40 -a -Y path/to/index path/to/fastq 2> /dev/null | samtools
sort -@ 32 -u | samtools mpileup -A -Q 20 -x -B --no-output-ins --no-output-ins -
-no-output-del --no-output-del --no-output-ends -f chr.fa - > /path/to/pileup.tsv
```

Bash history to the rescue?

- Similar commands entered at different working directories
- Scripts or config files modified without version control
- Input files changed on disk




```
$ which-command-created --restore-scripts-and-config-files foo.tsv
```

Usage example

https://github.com/tycho-kirchner/shournal

```
$ SHOURNAL_ENABLE
$ cat demo.sh
#!/usr/bin/env bash
echo hi | tee out.log
$
```

Usage example

https://github.com/tycho-kirchner/shournal

```
$ SHOURNAL_ENABLE
$ cat demo.sh
#!/usr/bin/env bash
echo hi | tee out.log
$ ./demo.sh
hi
$
```

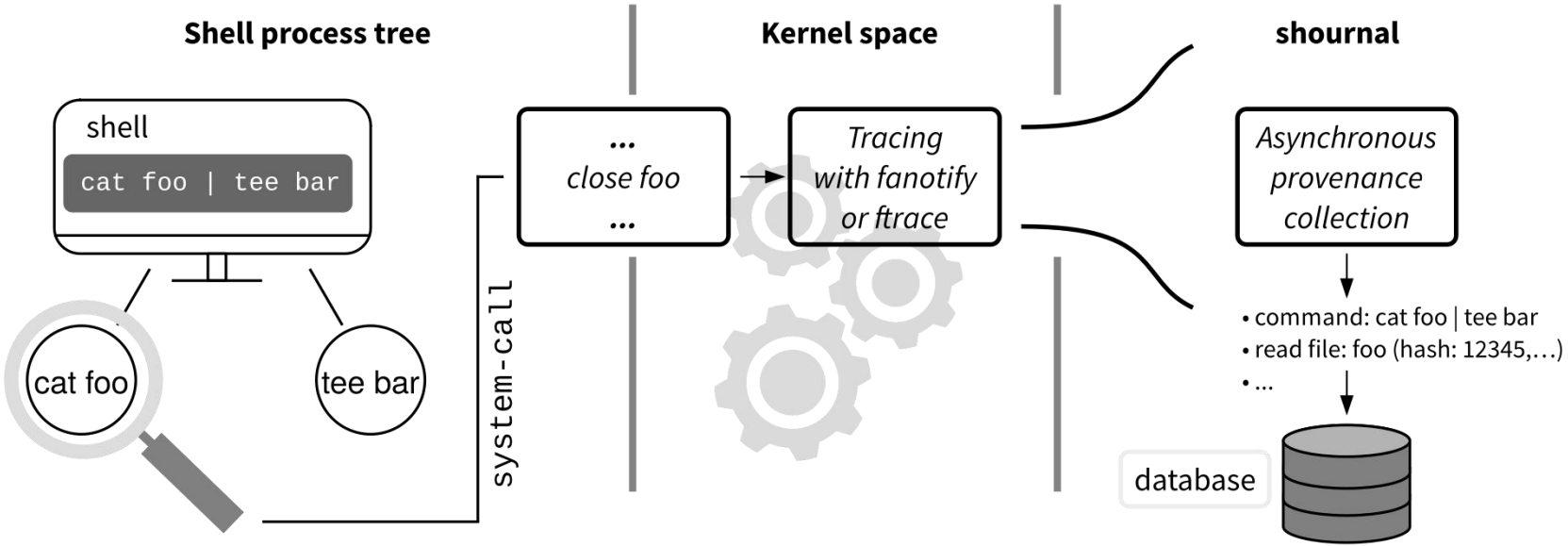
Usage example

https://github.com/tycho-kirchner/shournal

```
$ SHOURNAL_ENABLE
$ cat demo.sh
#!/usr/bin/env bash
echo hi | tee out.log
$ ./demo.sh
hi
$ shournal -q --wfile out.log
cmd-id 2 $?=0 2022-12-08 08:46 $ ./demo.sh
Working directory: /home/user
  1 written file:
     /home/user/out.log (3 bytes) Hash 24
  1 read file:
     /home/user/demo.sh (42 bytes) Hash 17
          #!/usr/bin/env bash
          echo hi | tee out.log
```

How it works

https://github.com/tycho-kirchner/shournal



Make use of the collected provenance

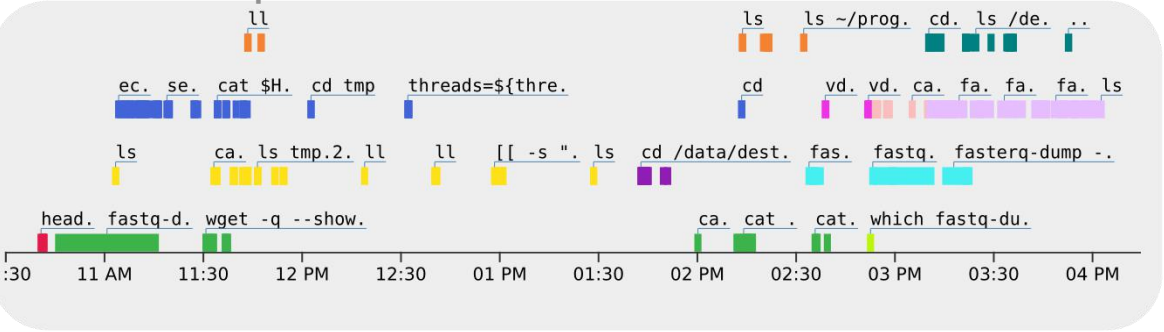
- Which commands created or modified a given file (by size and checksum or filename)
- Restore old scripts and configuration files without VCS (saved based on file suffix)
- Detect changes in input files
- Files used below a given path
- Files modified during a given period of time
- ...

```
$ shournal --query -cwd $PWD -cmdtxt ./pipeline.sh% --wpath $PWD/%
--command-end-date -between 2022-12-01 2022-12-31 --restore-rfiles
```

Make use of the collected provenance

https://github.com/tycho-kirchner/shournal

Export as standalone html document



snakemake

https://github.com/snakemake/shournal-to-snakemake

nextflow

...and others

ReproZip

The Reproducibility Packer!

CDE

CARE

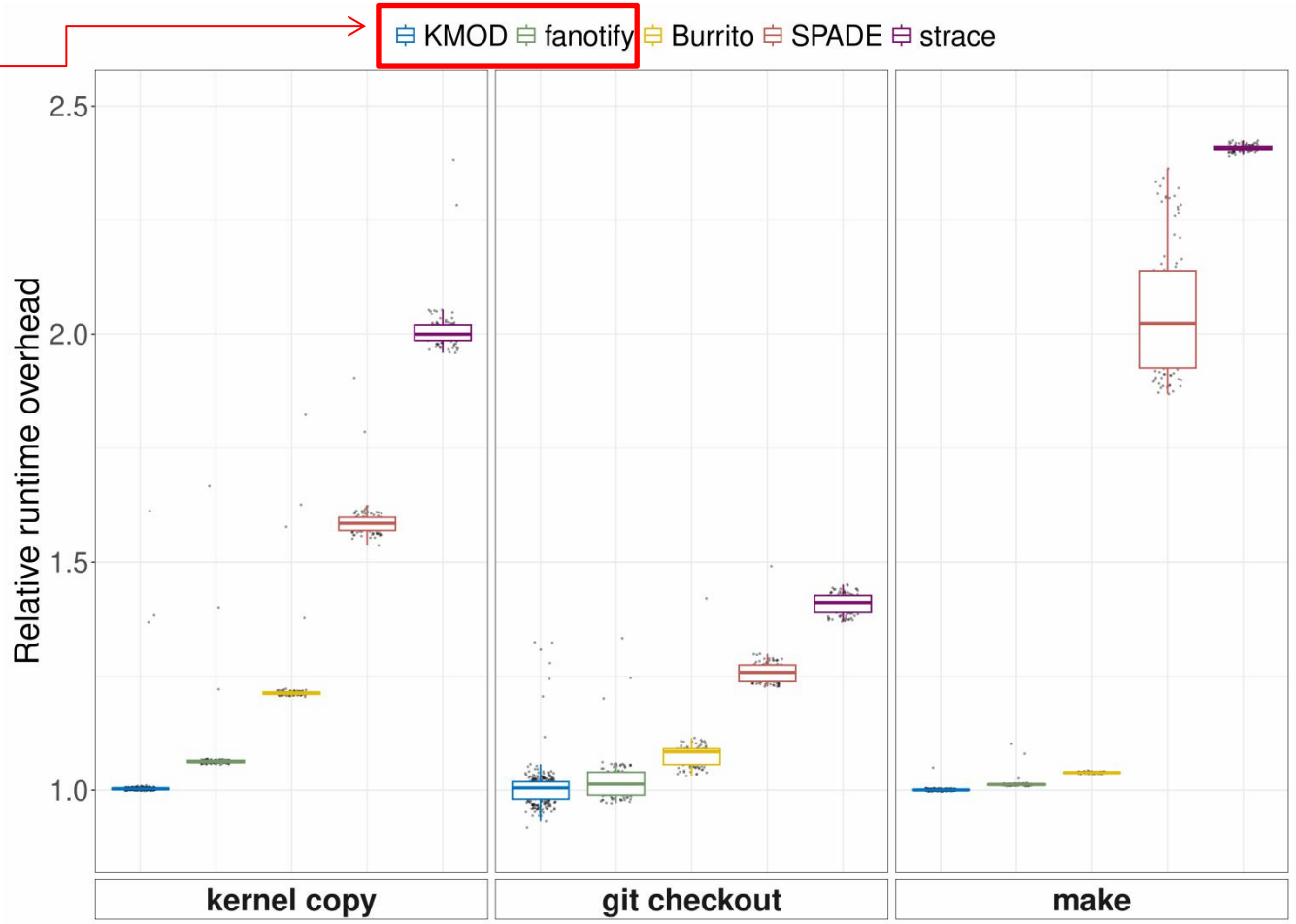
High runtime- and storage overhead.
Local reproducibility of the whole pipeline required.

Runtime-overhead

Comparison of shournal's tracing backends *KMOD* and *fanotify* with other tracing tools.

Only 0.6% overhead is achieved by:

- ftrace
- tracing only the *close* file operation
- primarly capturing metadata and **partial** checksums



Current limitations

- Binary executables are not tracked
- Remote execution not merged into parent command
- Files opened read-writable are treated as written files
- File-actions performed via IPC (interprocess communication)
- Extremely high file activity leads to gaps
- File-provenance, where file handles are shared with unobserved processes, is possibly lost
- Memory-mapped files are sometimes* not tracked
- The fanotify backend has additional minor limitations

* when the underlying file-descriptor is closed before the file is written

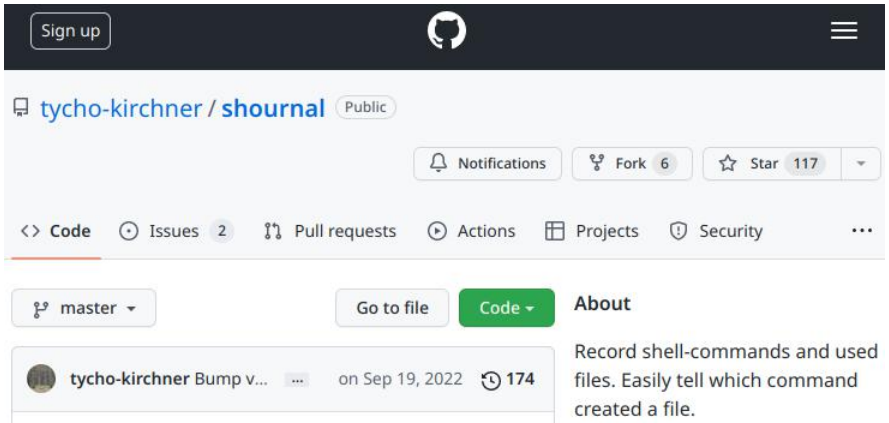
shournal enables you to ...

- Determine the provenance of files
- Restore old scripts and configuration files without VCS
- Detect changes in input files
- Summarize and resume projects quickly
- Narrow the gap to the creation of fully reproducible experiments with ReproZip, CDE, CARE or snakemake (and possibly others)
- Re-do ridiculous complicated commands

... with *low* overhead

Acknowledgments

Special thanks to:
Steve Hoffmann
Konstantin Riege



Storage overhead

The storage overhead depends on user configuration

- Which directories to monitor for file events (e.g. ignore /tmp)
- Which and how many files to store entirely (by suffix)
- Max. number of file events to store per command (backupscript!)
- cp-benchmark yields approx. 174 bytes per file event
- interface provided to delete unneeded entries (e.g. older than two years)