flowr

streamlining workflows

speed up analysis using a computing cluster

why bother?

➡when one needs to wrangle a lot of data

➡and there are multiple steps involved

➡esp. when some of the steps can be further broken down and processed in parallel

➡use a computing cluster, submit a web of jobs

✓ Effectively process a **multi-step pipeline**, spawning it across the computing cluster

✓ **Reproducible** and **transparent**, with cleanly structured execution logs

✓ **Track** and **re-run** flows

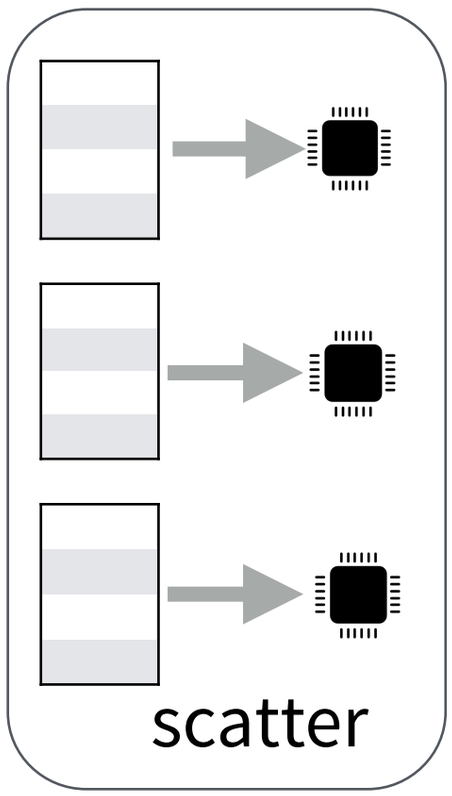
✓ **Lean** and **Portable**, with easy installation

✓ Run the same pipeline in the cloud (using star cluster) OR a local machine

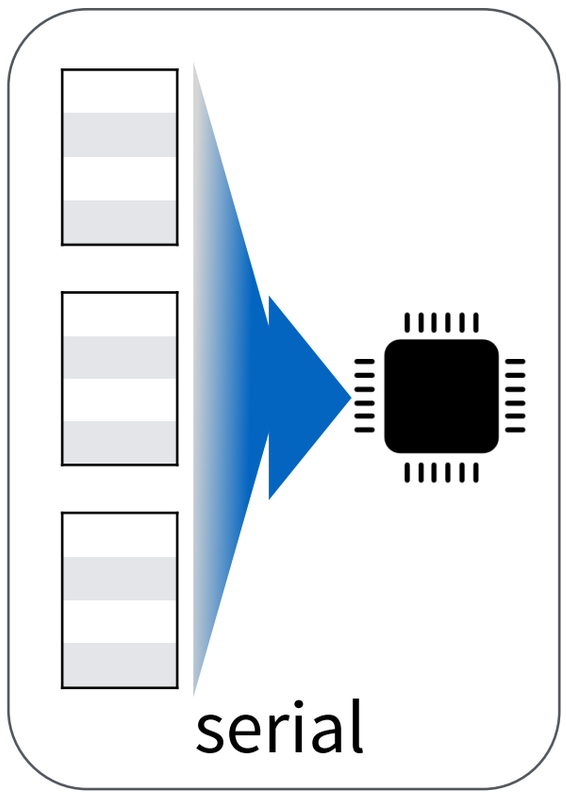
✓ Supports **multiple cluster computing platforms** (torque, lsf, sge, slurm …)

five simple terms, defining all relationships

submission types



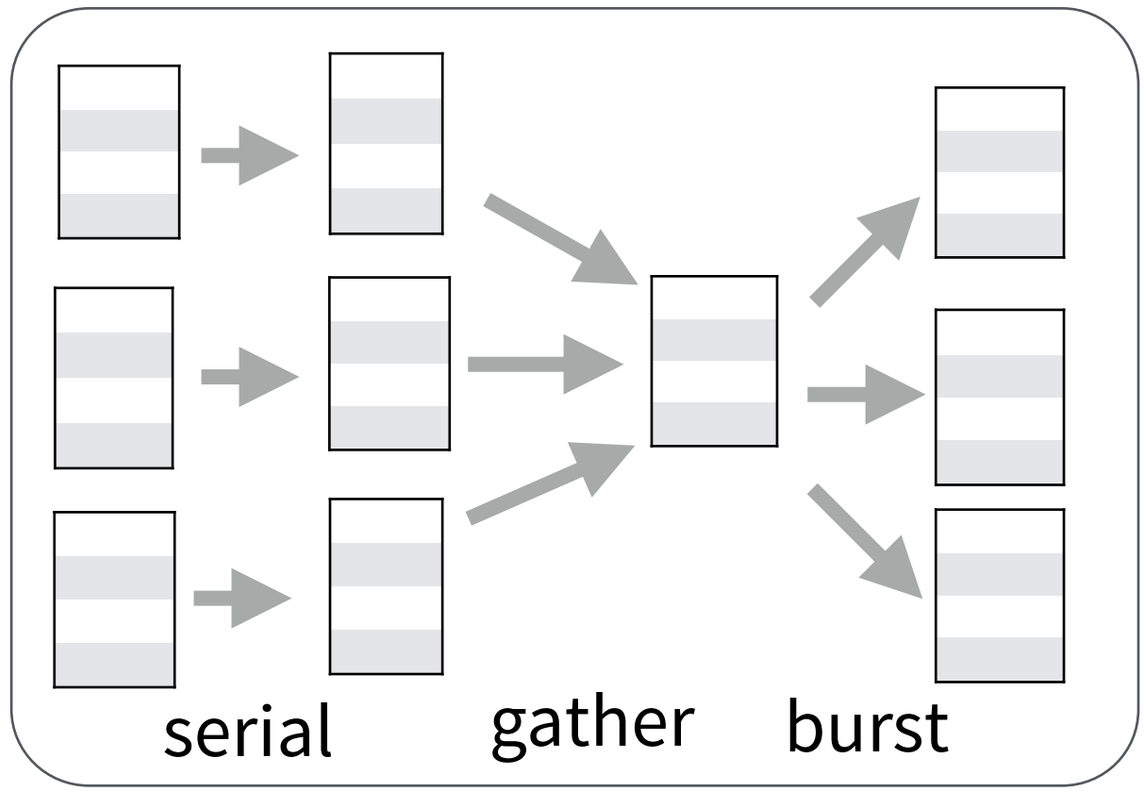
all in parallel



sequentially

decide how pieces of a single step are processed

dependency types



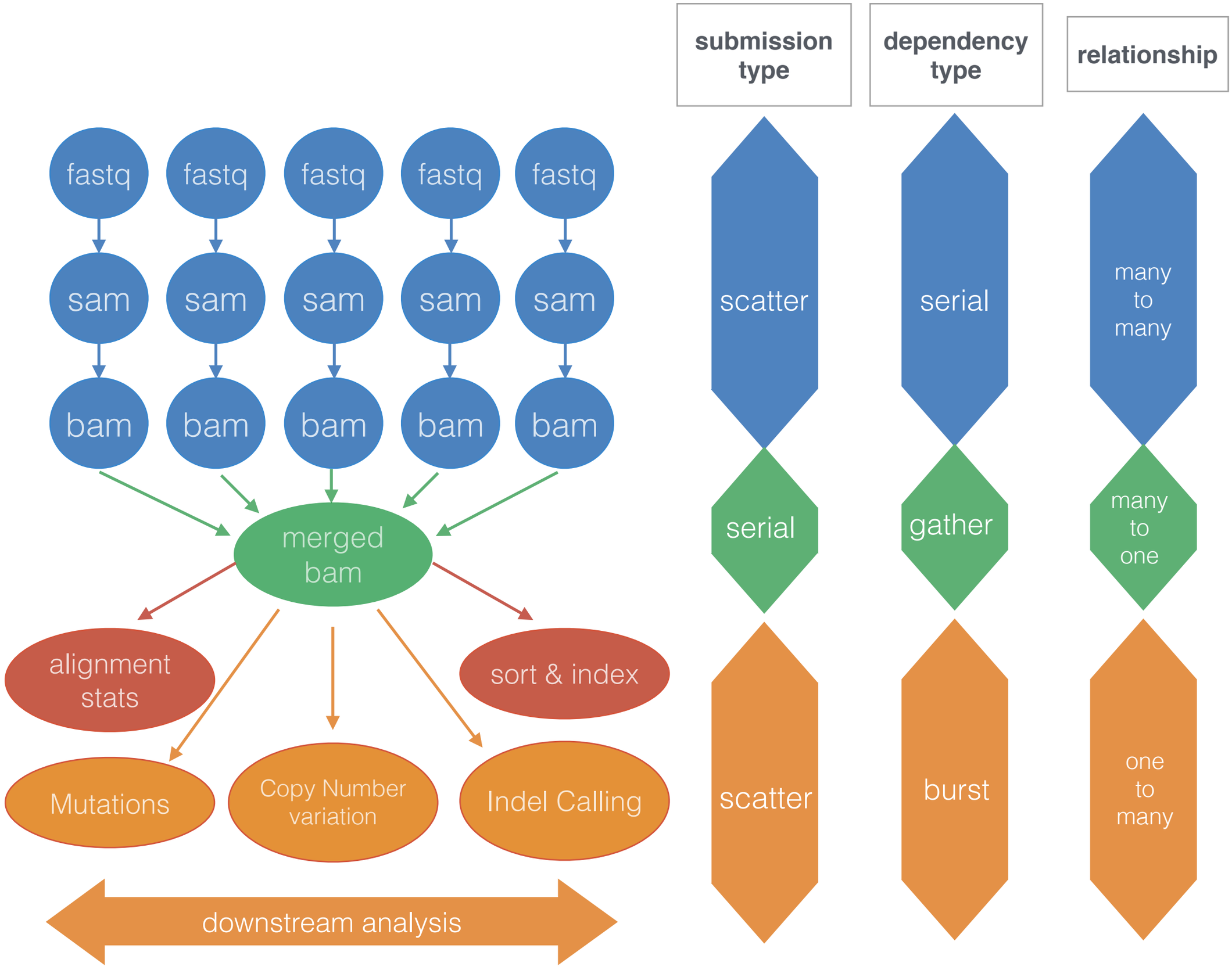
one-to-many

many-to-one

many-to-many

decide the relationship b/w steps

Using a genomics example flow, with flowr concepts



a simple pipeline, where

- ★ we would sleep for a few seconds
- ★ create a few small files
- ★ merge those files
- ★ get the size of the resulting merged file

simple pipeline in bash

say Hello to the world

wait for a few seconds...

create two small files

merge the two files

check the size of the resulting file

```
echo 'Hello World !'

sleep 5
sleep 5

cat $RANDOM > tmp1
cat $RANDOM > tmp2

cat tmp1 tmp2 > tmp

du -sh tmp
```

wrap bash commands into R

say Hello to the world

wait for a few seconds…

create two small files

merge the two files

check the size of the resulting file

```
hello='echo Hello World !'

sleep=c('sleep 5', 'sleep 5')

tmp=c('cat $RANDOM > tmp1',
      'cat $RANDOM > tmp2')

merge='cat tmp1 tmp2 > tmp'

size='du -sh tmp'
```

create a table of all commands

```
hello='echo Hello World !'

sleep=c('sleep 5', 'sleep 5')

tmp=c('cat $RANDOM > tmp1',
      'cat $RANDOM > tmp2')

merge='cat tmp1 tmp2 > tmp'

size='du -sh tmp'
```

```
library(flowr)
lst = list(hello=hello,
           sleep=sleep,
           tmp=tmp,
           merge=merge,
           size=size)

flowmat = to_flowmat(lst, "samp1")
```

create a **named** list

create a table

```
|samplename |jobname |cmd                  |
|:----------|:-------|:--------------------|
|samp1      |hello   |echo Hello World !   |
|samp1      |sleep   |sleep 5              |
|samp1      |sleep   |sleep 5              |
|samp1      |tmp     |cat $RANDOM > tmp1   |
|samp1      |tmp     |cat $RANDOM > tmp2   |
|samp1      |merge   |cat tmp1 tmp2 > tmp  |
|samp1      |size    |du -sh tmp           |
```

a simple tab-delim table

connect the dots…

flow definition decides the sequence of steps

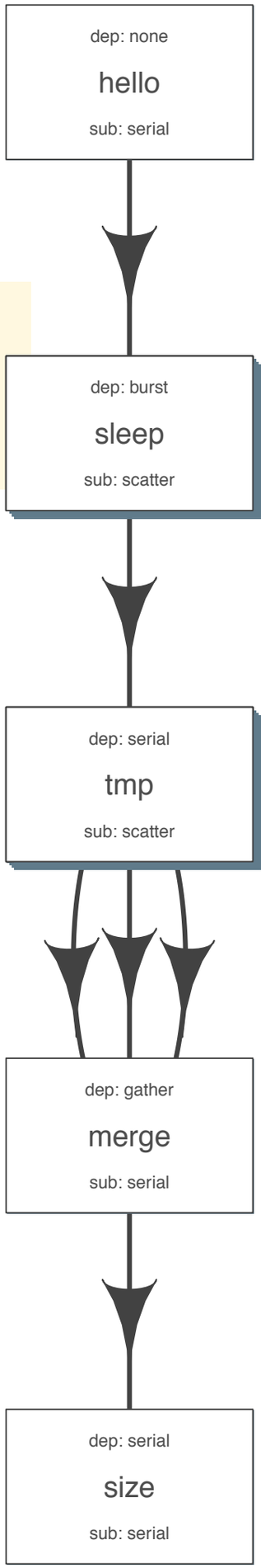
create a flow definition

```
flowdef = to_flowdef(flowmat,
        sub_type = c("serial", "scatter", "scatter", "serial", "serial"),
        dep_type = c("none", "burst", "serial", "gather", "serial"),
        platform = "local")
```

jobname	sub_type	prev_jobs	dep_type	cpu
hello	serial	none	none	1
sleep	scatter	hello	burst	1
tmp	scatter	sleep	serial	1
merge	serial	tmp	gather	1
size	serial	merge	serial	1

a simple tab-delim table

```
plot_flow(flowdef)
```



stitch a flow…

use a flow mat and flow def, to create a flow object

flow mat

samplename	jobname	cmd
samp1	hello	echo Hello World !
samp1	sleep	sleep 5
samp1	sleep	sleep 5
samp1	tmp	cat $RANDOM > tmp1
samp1	tmp	cat $RANDOM > tmp2
samp1	merge	cat tmp1 tmp2 > tmp
samp1	size	du -sh tmp

+

flow def

jobname	sub_type	prev_jobs	dep_type	cpu
hello	serial	none	none	1
sleep	scatter	hello	burst	1
tmp	scatter	sleep	serial	1
merge	serial	tmp	gather	1
size	serial	merge	serial	1

stitch & submit to the cluster (cloud or server)

```
fobj = to_flow(flowmat, flowdef, execute = TRUE)
```

```
Flow is being processed. Track it from R/Terminal using:
flowr status x=~/flowr/runs/flowname-samp1-20151005-16-01-38-M8WniKJo
OR from R using:
status(x='~/flowr/runs/flowname-samp1-20151005-16-01-38-M8WniKJo')
```

```
Working on: hello
  |=====                                            |  25%
Working on: sleep
  |==========                                       |  50%
Working on: merge
  |===============================================| 100%
Working on: size
```

submit a flow, then...

`status()`

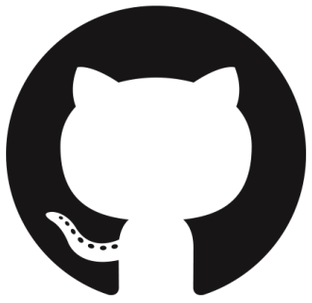
monitor the status of a single flow
OR multiple flows

`kill()`

kill all the associated jobs, of one or many flows

`rerun()`

One can rerun the flow from a intermediate step



github.com/sahilseth/flowr

complete documentation: docs.flowr.space

email: sahil.seth@me.com

Extra details

Flow mat

samplename	jobname	cmd
sample1	A	sleep 2 && sleep 5;echo hello
sample1	A	sleep 13 && sleep 7;echo hello
sample1	B	head -c 100000 /dev/urandom > tmp1
sample1	B	head -c 100000 /dev/urandom > tmp1
sample1	C	cat tmp1 tmp2 tmp3 > merged
sample1	D	du -sh merged
sample1	D	ls merged

- use any language to create a flow mat (a tsv file)
- cmd column defines commands to run

Flow Definition

	Define Relationships			Resource Requirements				
jobname	submission type	previous job(s)	dependency type	queue	memory	time	cpu	platform
A	scatter	none	none	medium	163185	23:00	1	lsf
B	scatter	A	serial	medium	163185	23:00	1	lsf
C	serial	B	gather	medium	163185	23:00	1	lsf
D	scatter	C	burst	medium	163185	23:00	1	lsf

- creativily define relationships using submission and dependency types
- each row describes resources for **one** step, providing full flexibility