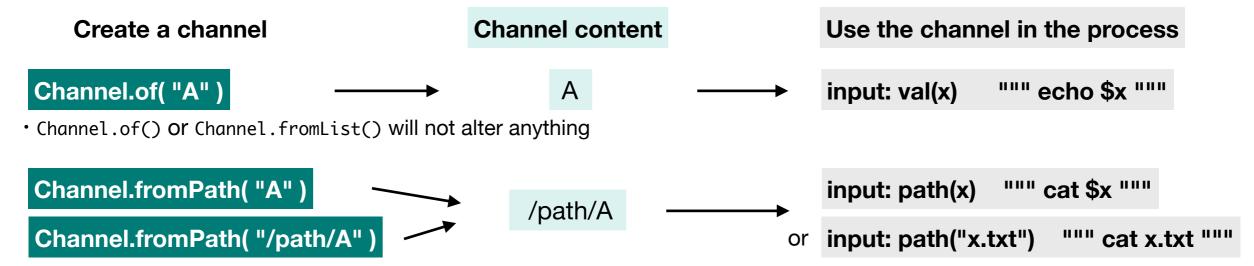
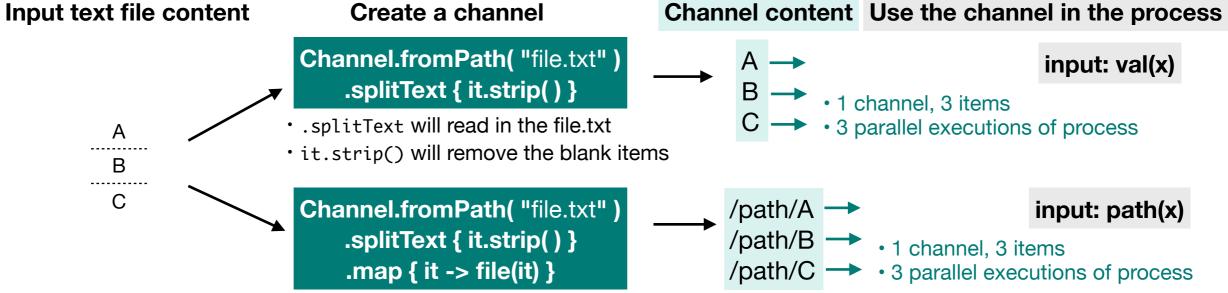
Nextflow cheatsheet: creating input channels



- If the full file path is absent, .fromPath() will prefix current folder as path
- · So the resulting channels always carry a full path

input: path("x.txt") will create a symlink to /path/A
in the working directory with the name "x.txt"

Creating input channels from a text file



· file() adds current folder as path unless there is already full path in the item

:		
A /path/A.bam /path/A.bam.bai Cl	nannel.fromPath("file.tsv")	F A / 11 /A 1
B /path/B.bam /path/B.bam.bai	splitCsv(sep: "\t")	[A,/path/A.b
	map { row -> [row[0], file(row[1]), file(row[2])] }	[B,/path/B.b
C /path/C.bam /path/C.bam.bai	inap (Tow > [Tow[o], mo(row[1]), mo(row[2])]]	[C,/path/C.b

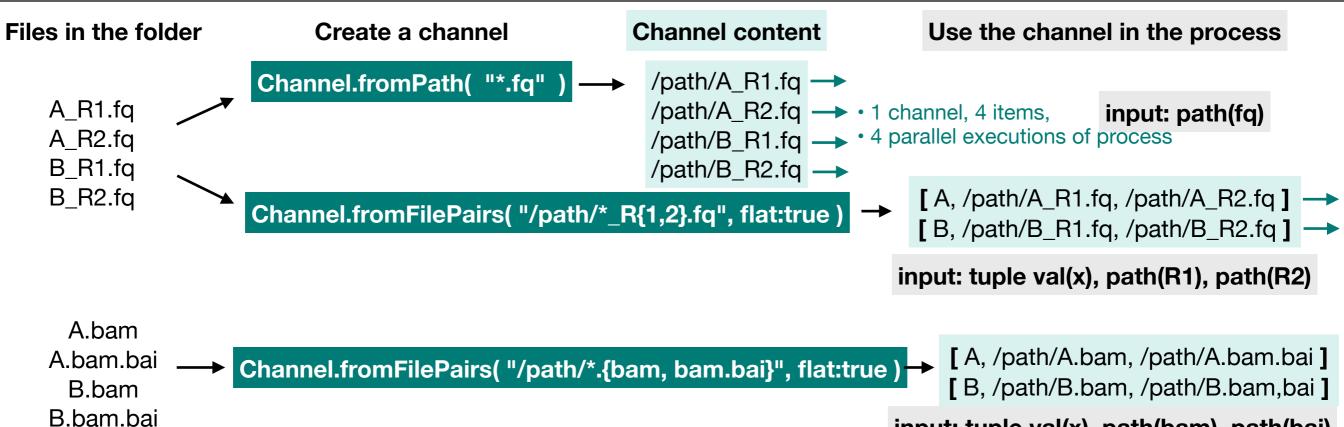
 .map{ } is very useful to select columns and specify channel structure. Here it converts the row to a tuple [A, /path/A.bam, /path/A.bam.bai] ➤ [B, /path/B.bam, /path/B.bam.bai] ➤ [C, /path/C.bam.bai] ➤

input: tuple val(x), path(bam), path(bai)

strain	bam
Α	A.bam
В	B.bam
С	C.bam

- A path is added with params.bam_path
- If "params.bam_path" doesn't add a full path, file() will.
- [x, y] is the same as tuple(x, y)

Creating input channels from a list of files



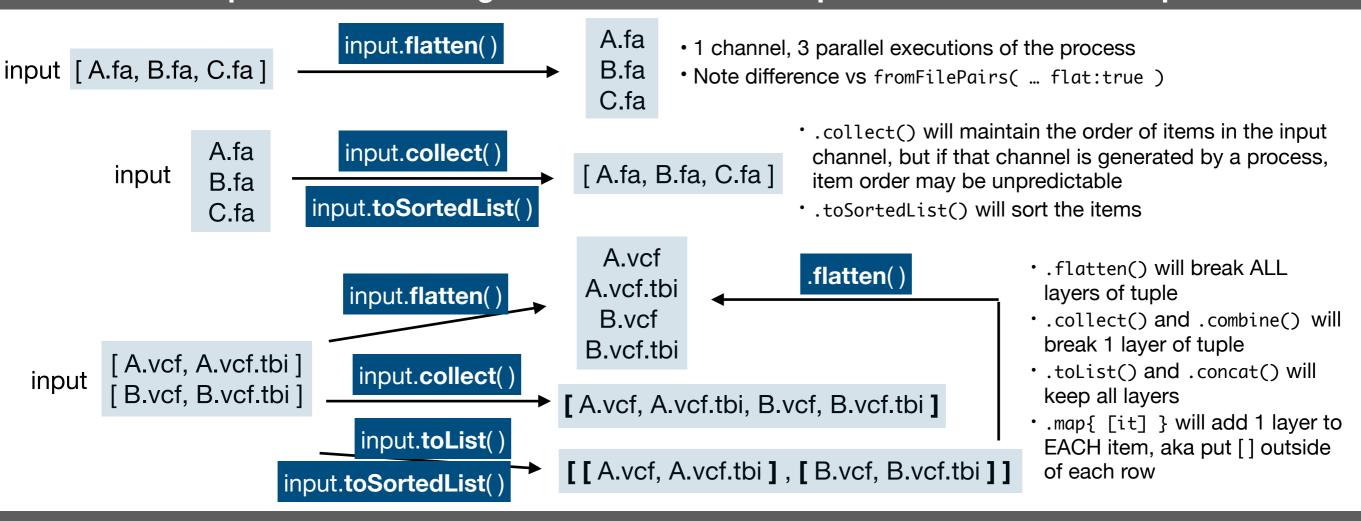
[•] First item "A" came from stripping the path and common pattern ".{bam, bam.bai}" as specified in .Channel.fromFilePairs

input: tuple val(x), path(bam), path(bai)

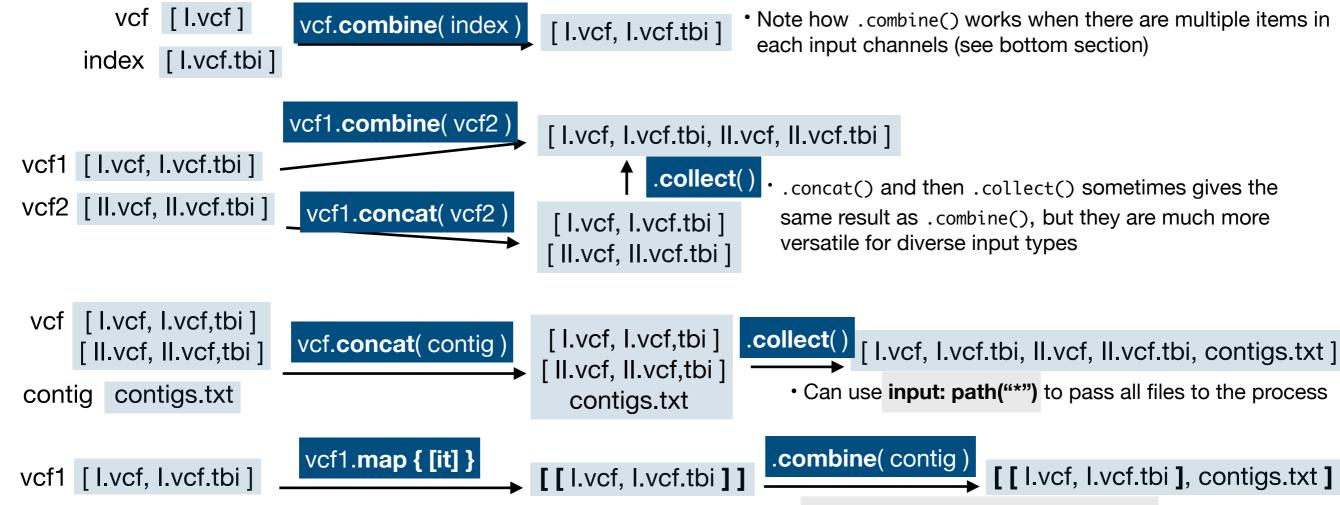
Nextflow cheatsheet: combine inputs into 1 channel

examples of 1 channel: A.fa Each item (row) has 1 execution of process Channel.of("A.fa", "B.fa", "C.fa") • 1 channel, 3 items B.fa • 3 parallel executions of the process C.fa → Channel.of(["A.fa", "B.fa", "C.fa"]) [A.fa, B.fa, C.fa] -• 1 channel, 1 item, 1 execution Channel.of(["l.vcf", "l.vcf.tbi"], [I.vcf, I.vcf.tbi] • 1 channel, 3 items ["II.vcf", "II.vcf.tbi"], [II.vcf, II.vcf.tbi] • 3 parallel executions of process ["III.vcf", "III.vcf.tbi"]) • [] indicates a "set" "tuple" "ArrayList" [III.vcf, III.vcf.tbi]

• With 1 input channel: change number of items and parallel execution of the process



With multiple input channels: combine them and change number of items



• This is especially useful when number of items in vcf1 could vary. In process: input: tuple path("*"), path(contig)

counting starts from 0 vcf A.vcf [A.vcf, chr1] [A, 1, I]vcf.combine(contig input1.combine(input2,by:0) [A, 1] B.vcf [A.vcf, chr2] input1 [B, 2, m] [B.vcf, chr1] [B, 2] All combinations [B, 2, n] chr1 contig [B.vcf, chr2] chr2 [A, I] input2 [B, m] [A, 1, I]input2.collect { it[1] } [B, n] input1.join(input2) [I, m, n] [B, 2, m]

Other useful cases

Nextflow 22.04.5 https://github.com/danrlu/nextflow_cheatsheet "it" represents an item, which is used throughout the cheatsheet .join() will keep only the first match