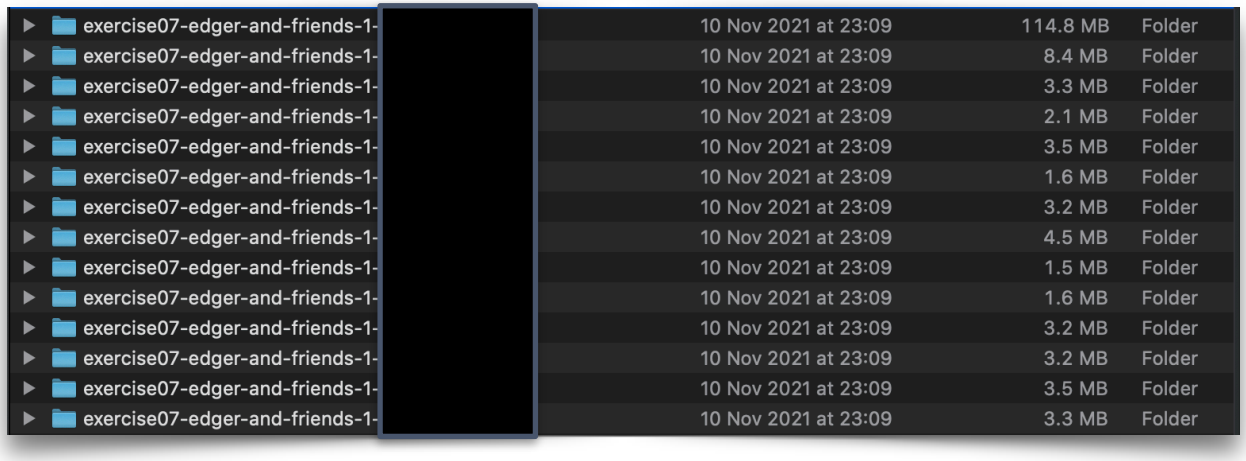
Exercise 7 notes

- no need to check in *data* to your assignment (GitHub) repo



Exercise 7 notes

```
> (samples <- read.table("samples.txt", header=TRUE,
+                        row.names=5, stringsAsFactors=FALSE))
                  rep condition libtype shortname           countfile
CG8144_RNAi-1.bam  T1         T      SE     T1.SE CG8144_RNAi-1.count
CG8144_RNAi-3.bam  T3         T      PE     T3.PE CG8144_RNAi-3.count
CG8144_RNAi-4.bam  T4         T      PE     T4.PE CG8144_RNAi-4.count
Untreated-1.bam    C1         C      SE     C1.SE   Untreated-1.count
Untreated-6.bam    C6         C      SE     C6.SE   Untreated-6.count
Untreated-3.bam    C3         C      PE     C3.PE   Untreated-3.count
Untreated-4.bam    C4         C      PE     C4.PE   Untreated-4.count
> (mm <- model.matrix(~condition, data=samples))
                  (Intercept) conditionT
CG8144_RNAi-1.bam           1          1
CG8144_RNAi-3.bam           1          1
CG8144_RNAi-4.bam           1          1
Untreated-1.bam             1          0
Untreated-6.bam             1          0
Untreated-3.bam             1          0
Untreated-4.bam             1          0
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$condition
[1] "contr.treatment"
```

- can use a data.frame in `model.matrix`

- note the use of metadata, instead of manually making a vector

Exercise 7 notes

- remember ```` ```{r message=FALSE} ```` … to avoid all the startup messages

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

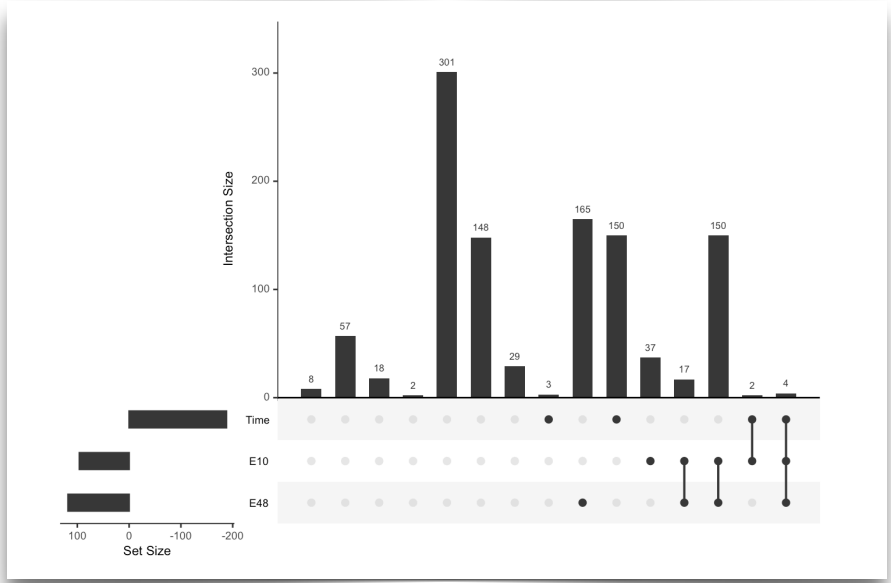
```
##
## Attaching package: 'BiocGenerics'
```

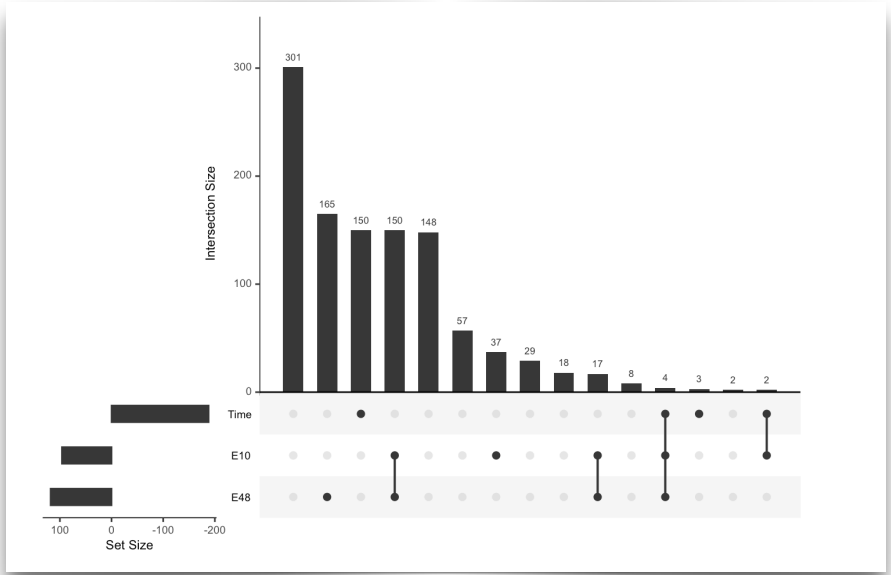
```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:limma':
##
##     plotMA
```

Exercise 7 notes

- unusual UpSet plots





RNA-seq data processing and analysis (with ARMOR)

STA426 – 15.11.2021

(many slides from Katharina Hembach)

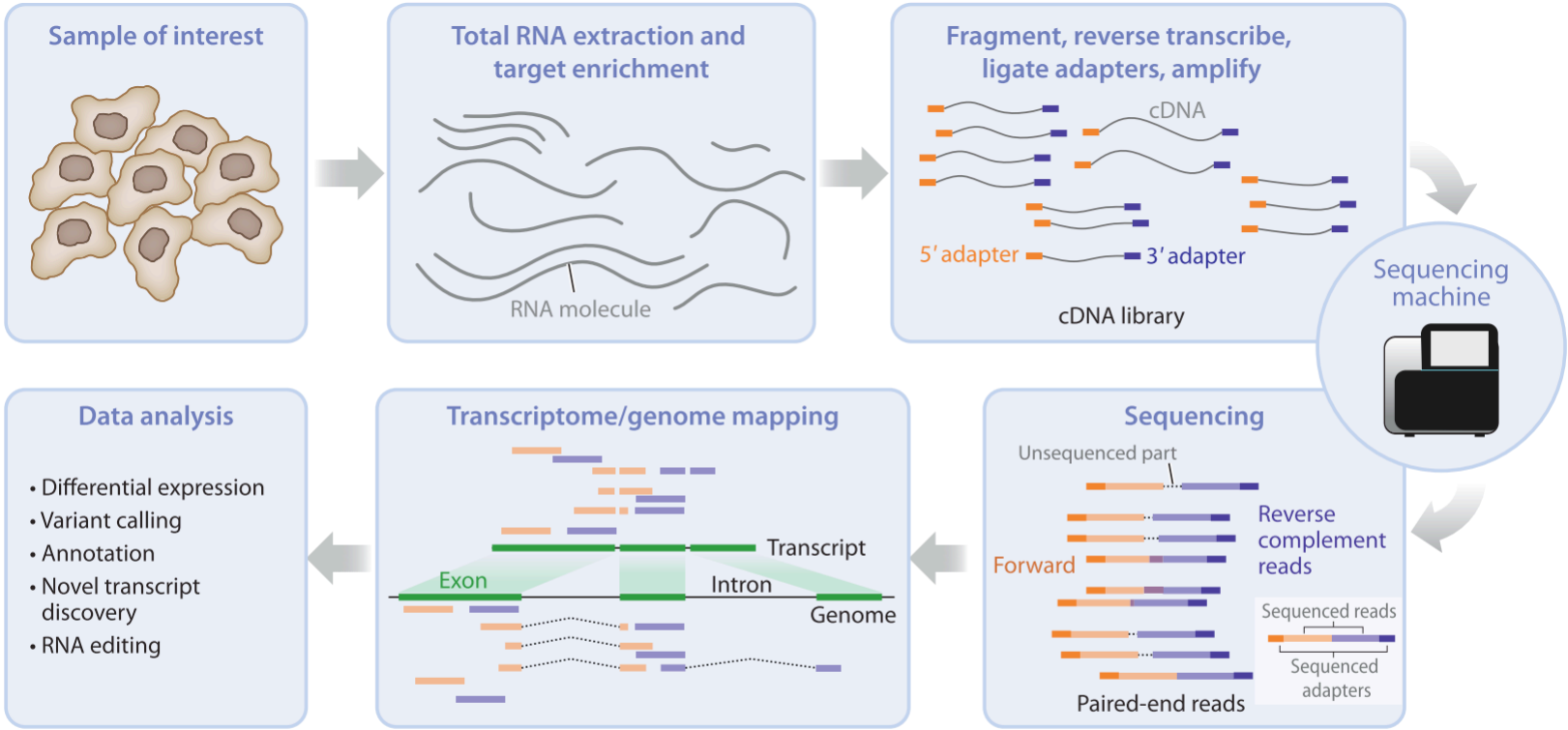
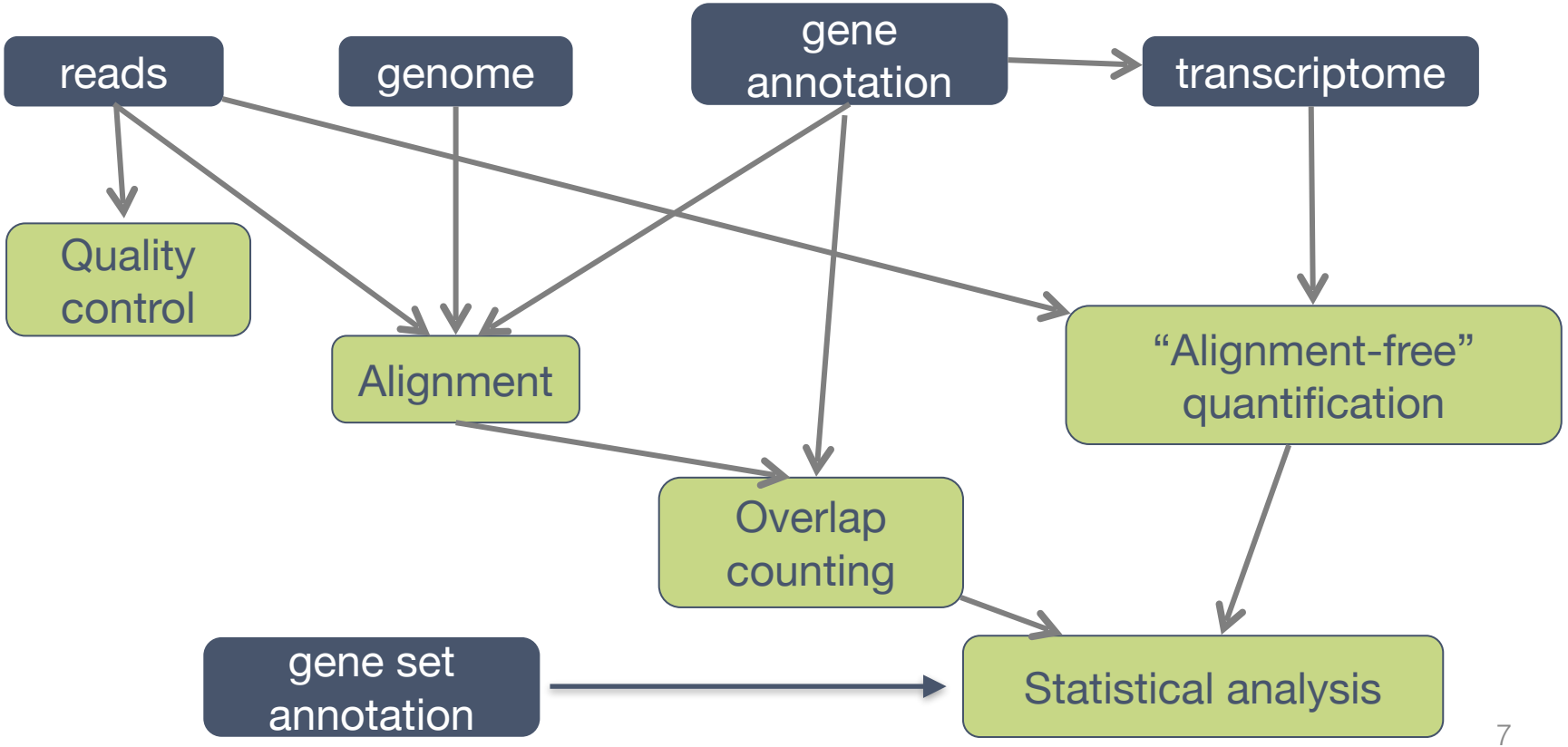


Figure 1

Overview of the experimental steps in an RNA sequencing (RNA-seq) protocol. The complementary DNA (cDNA) library is generated from isolated RNA targets and then sequenced, and the reads are mapped against a reference genome or transcriptome. Downstream data analysis depends on the goal of the experiment and can include, among other things, assessing differential expression, variant calling, or genome annotation.

Van den Berge, K., et al. (2019). RNA Sequencing Data: Hitchhiker's Guide to Expression Analysis. *Annual Review of Biomedical Data Science*

RNA-seq analysis pipeline



Ideally, you also (regarding of whether RNA-seq or another pipeline) ..

- perform "exploratory data analysis" at various points in a pipeline
- always check intermediate results and generate plots not just summaries! (e.g. FastQC)

ARMOR workflow

- **A**utomated = snakemake

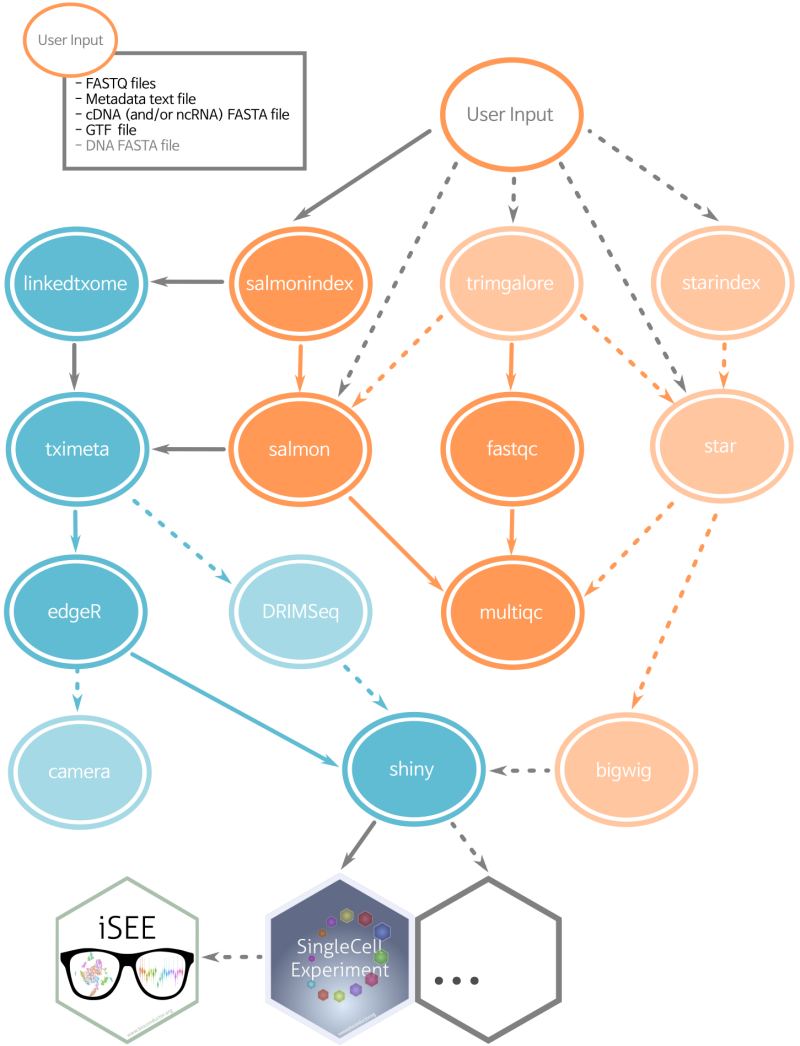
snakemake
A framework for reproducible data analysis

- **R**eproducible = conda and GitHub

CONDA GitHub Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

- **MO**dular = snakemake rules + configuration file

- **R**NA-seq



https://github.com/csoneson/ARMOR
Orjuela et al., G3 2019

Typical preprocessing of RNA-seq reads

1. Quality filtering & adapter trimming
   (remove reads with bad quality & adapter sequences)
2. Alignment to reference genome
3. Quantification of feature of interest
   (gene or transcript)

Quality control

- **FastQC** https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- **MultiQC** https://multiqc.info/
  - aggregates FastQC results from multiple samples, as well as Salmon and STAR output
- # reads, read length, read quality, GC content, % duplicated reads, adapter contamination, …
- Tools for quality filtering/adapter trimming:

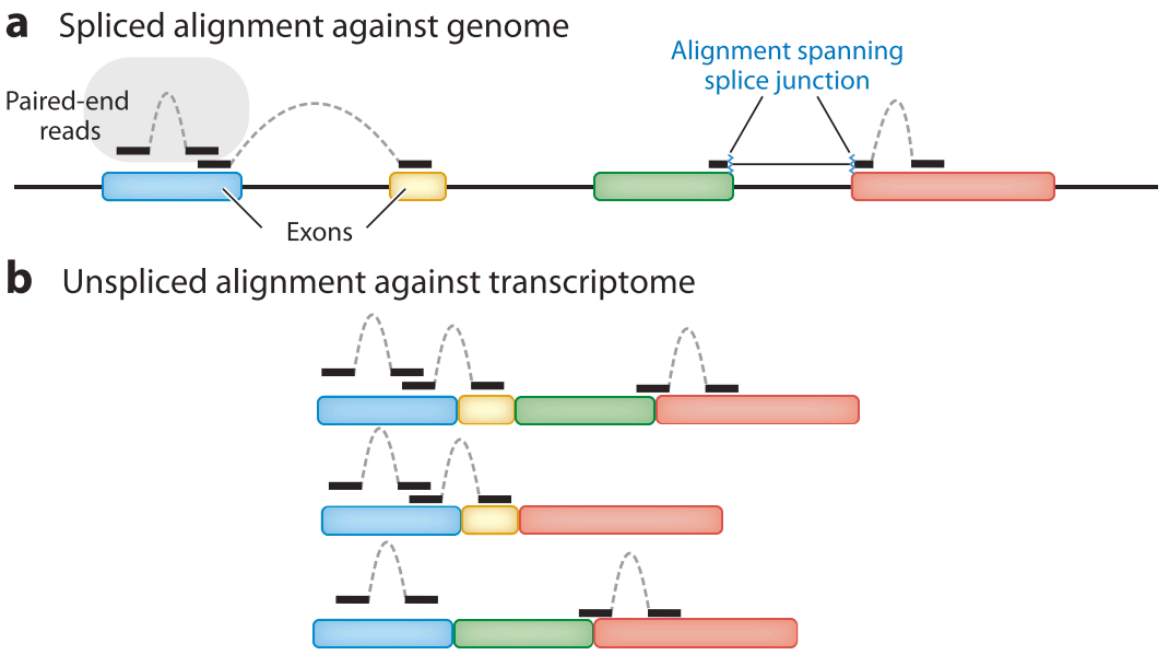
  cutadapt, TrimGalore!, Trimmomatic, FASTX-toolkit, …

Quality control

- Depending on FastQC result, you may need to fix (or troubleshoot) QC problems.
- Tools for quality filtering/adapter trimming:

  cutadapt, TrimGalore!, Trimmomatic, FASTX-toolkit, …

Alignment



STAR https://github.com/alexdobin/STAR

HISAT2 http://ccb.jhu.edu/software/hisat2/index.shtml

Salmon https://combine-lab.github.io/salmon/about/

kallisto https://pachterlab.github.io/kallisto/about

Van den Berge, K., et al. (2019). RNA Sequencing Data: Hitchhiker's Guide to Expression Analysis. *Annual Review of Biomedical Data Science*

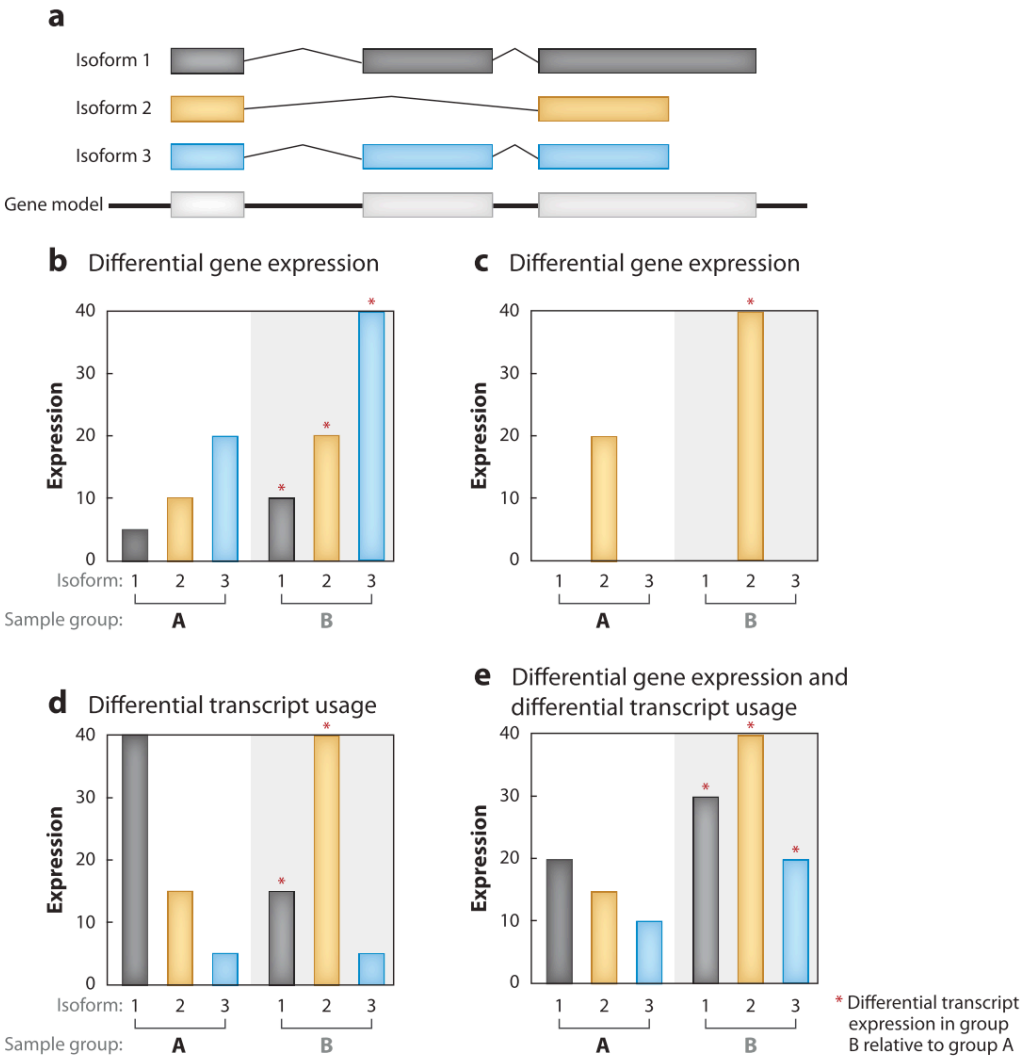
Alignment

Two possible approaches:

1. genome alignment: Find the most likely origin of each read on the genome.
   - create reference genome index for fast access
   - align reads to reference → BAM file
   - count number of reads overlapping with each feature (e.g. gene)
   - tools: STAR https://github.com/alexdobin/STAR or HISAT2 http://ccb.jhu.edu/software/hisat2/index.shtml

2. transcriptome mapping: “alignment-free” quantification
   - index reference transcriptome
   - quantify transcript abundance
   - summarise transcript counts to gene level (based on transcript to gene mapping)
   - tools: Salmon https://combine-lab.github.io/salmon/about/ or kallisto https://pachterlab.github.io/kallisto/about

Variants of differential expression



Van den Berge, K., et al. (2019). *Annual Review of Biomedical Data Science*

Statistical analysis

- **Differential gene expression:** Which genes change in expression in different genotypes, treatments, time points, …?

(edgeR http://bioconductor.org/packages/release/bioc/html/edgeR.html or DEseq2 http://bioconductor.org/packages/release/bioc/html/DESeq2.html)

- **Differential transcript usage**: Does the transcript composition of a given gene change?

(DRIMseq https://bioconductor.org/packages/release/bioc/html/DRIMSeq.html)

- **Gene set analysis**: are the DE genes enriched for a specific gene annotation category?

(`camera()` function from limma R package https://academic.oup.com/nar/article/40/17/e133/2411151)

HOW TO ORGANIZE YOUR SOFTWARE?

Many options for managing software environments

- Linux, Mac, Windows
- R has a packaging system, Python has a packaging system, but also many tools you either can download an executable version or the source code to compile into an executable
- Important to be conscious of the versions and environments that you are running; often with multi-package software environments, only certain versions
- install everything "locally" (manually) versus software environments (e.g., conda) versus containers (e.g., docker, singularity)
- no single correct way to manage software, but important to know that many options exist; to think about how you manage software in projects that you will run (e.g., over time when versions change)

https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html

- Open source package and environment management system for any programming language.
- quickly install, run and update packages and their dependencies
- packages are stored on different “channels” (locations)
- you need to specify the channel(s) when installing things
- bioconda is the channel for bioinformatics software

https://bioconda.github.io/

Conda environments

- you can manage packages/programs and their dependencies in environments
- no interaction with other environments
- easy to control package/language versions and avoid conflicts
- you can export an environment to a YAML file (https://yaml.org/spec/1.2/spec.html) and easily share it

→ reproducibility! (but beware ..)

Docker

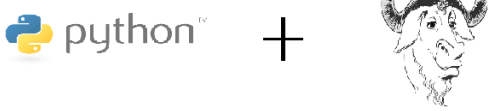
https://docs.docker.com/get-docker/

"*Docker is an open platform for developing, shipping, and running applications.* ***Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.*** *With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.*"

Docker


```
# list set of local images
docker image ls

# retrieve a docker image from dockerhub
docker pull markrobinsonuzh/sta426hs2021:latest

# if you build an image locally
# docker build
# docker push

# running containers
docker run -v /Users/mark/scratch:/home/rstudio/mnt -e \
  PASSWORD=bioc --cpus 4 --memory 16GB -p 8888:8787    \
  markrobinsonuzh/sta426hs2021:latest
docker run -it markrobinsonuzh/sta426hs2021:latest /bin/bash

# list running containers
docker ps

# login to running container
docker exec -it f9f60233082f /bin/bash
```

Snakemake



https://snakemake.readthedocs.io/en/stable/

- workflow management system

→ reproducible and scalable data analyses

- specify **rules** that describe how to create output files from input files
- file/rule dependencies are automatically determined
- rules can use shell commands, python code or external python/R scripts
- runs on laptops, clusters, the cloud without modifications
- you can automatically deploy required software with conda

What does it look like?

- Define Snakefile with rules

```
rule hello:
    input:
        "my_name.txt"
    output:
        "hello.txt"
    shell:
        "NAME=$(cat {input}); "
        "echo Hello $NAME! > {output}"
```

- Execute with

```
(snakemake) katharina@IMLS-NBM-KHE:~/Desktop$ snakemake hello --cores 1
```

Snakemake: useful commands

`--help` to get detailed help message

`--use-conda` to run rules in conda environments

`-n` dry run → only display what would be done but do not execute anything

`-p` print shell commands that will be executed

`-r` print reason for each executed rule

can be combined in `-npr`

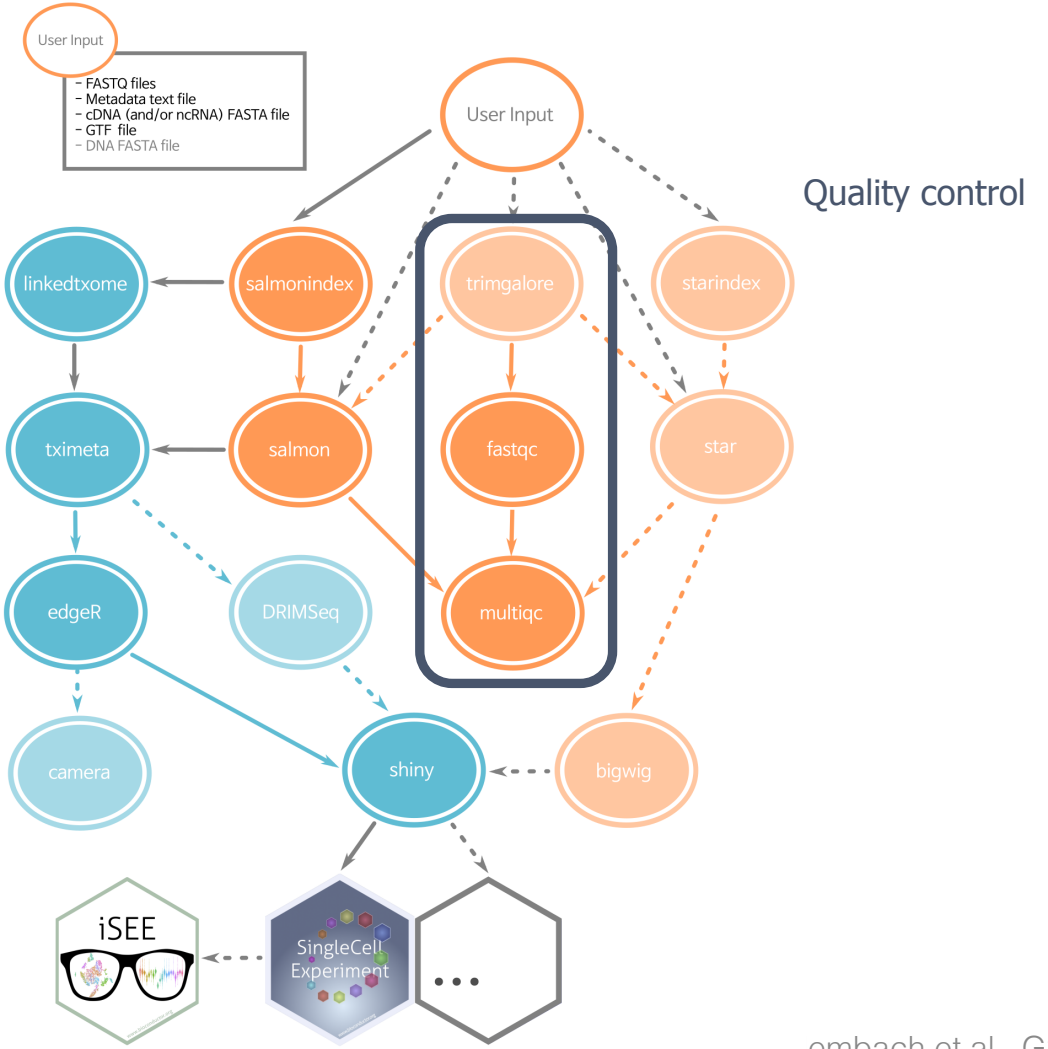
`-l` list all available rules

--cores to use at most this number of cores in parallel

--configfile path to configuration file (e.g. config.yaml)

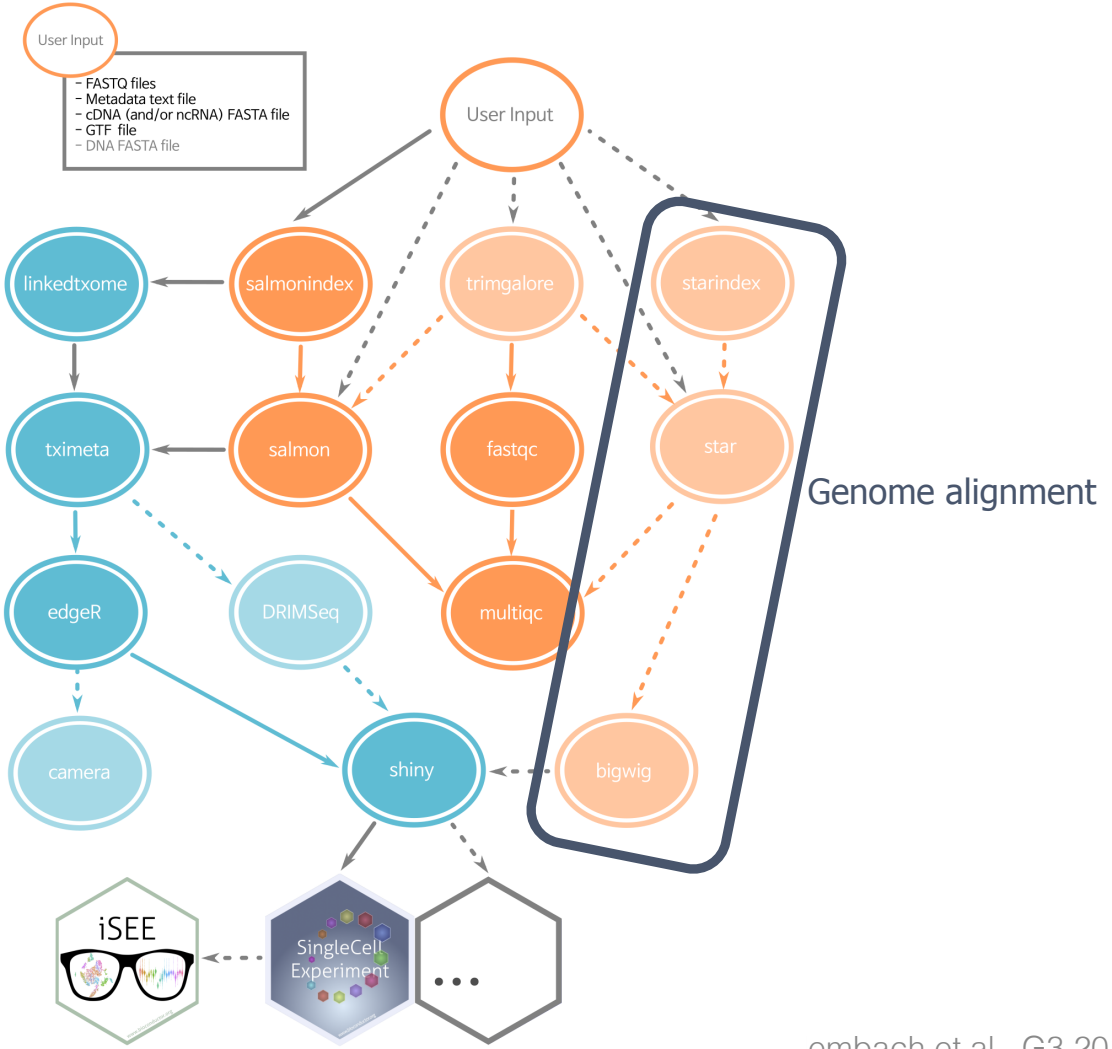
ARMOR WORKFLOW

ARMOR workflow



Srijuela, Huang, Frembach et al., G3 2019

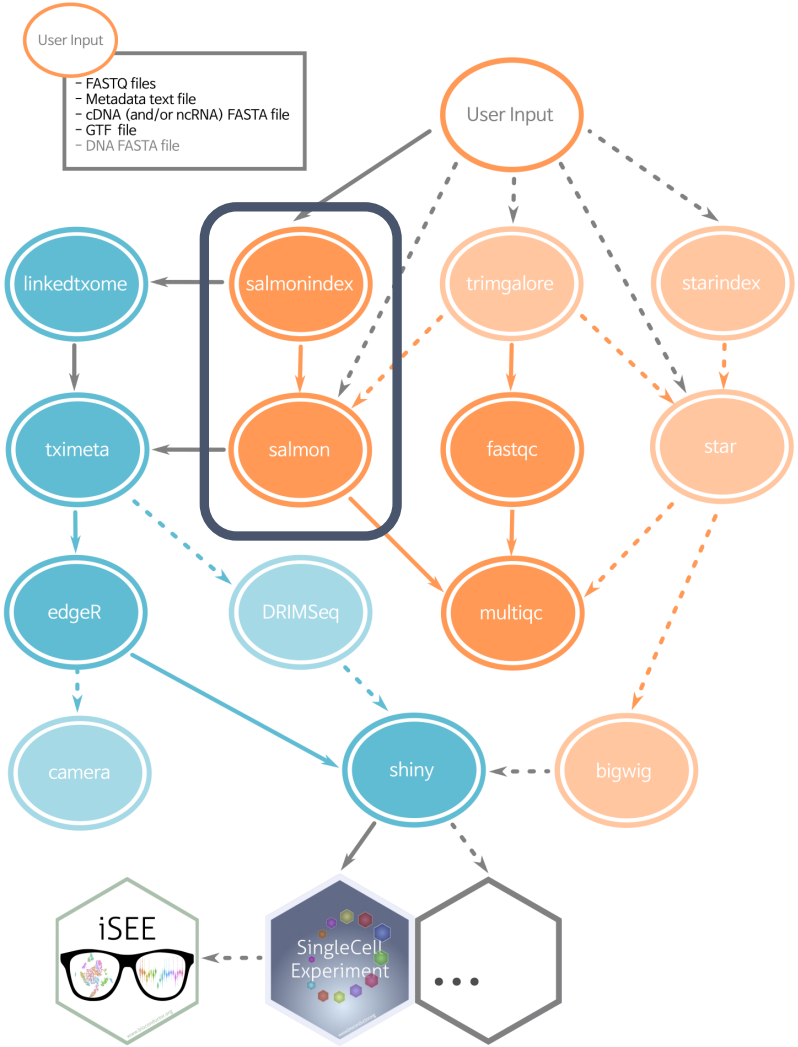
ARMOR workflow



Srijuela, Huang, Rembach et al., G3 2019

ARMOR workflow

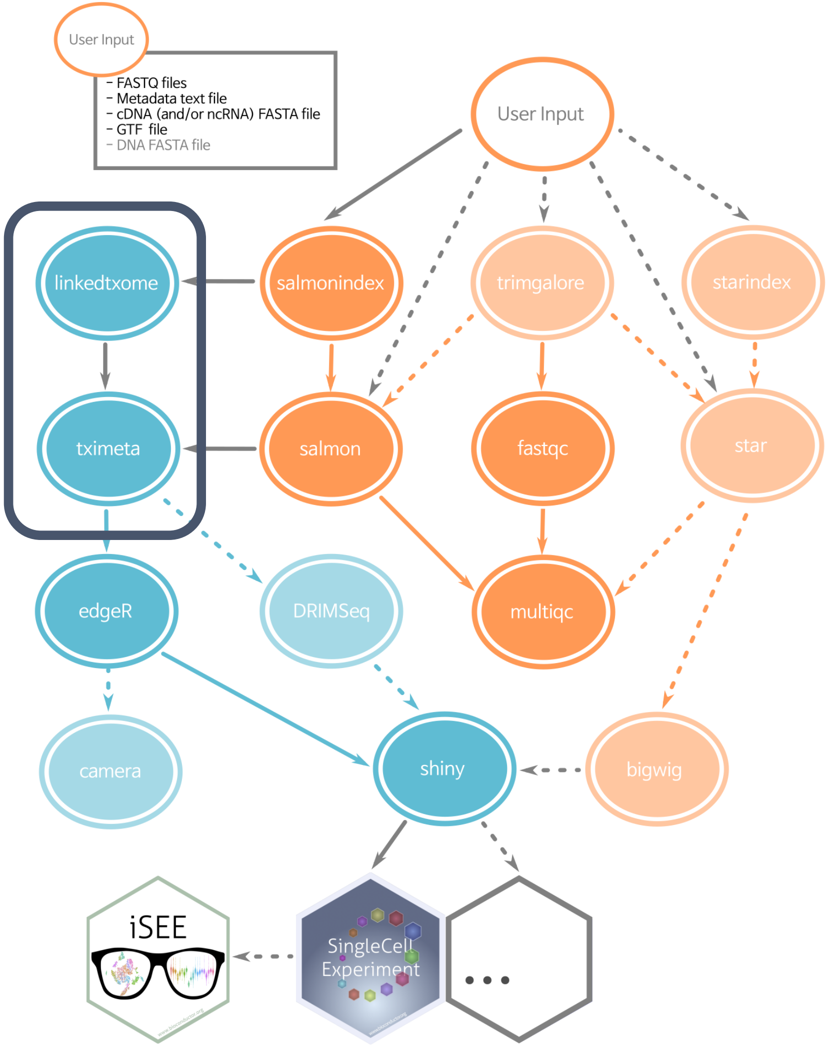
Transcript abundance estimation



Srivastava, Malik, Prjibelski, Hoang, Yorukoglu, Robertson, Patro, Sheffield, Kingsford, Love, Steinbach et al., G3 2019

ARMOR workflow

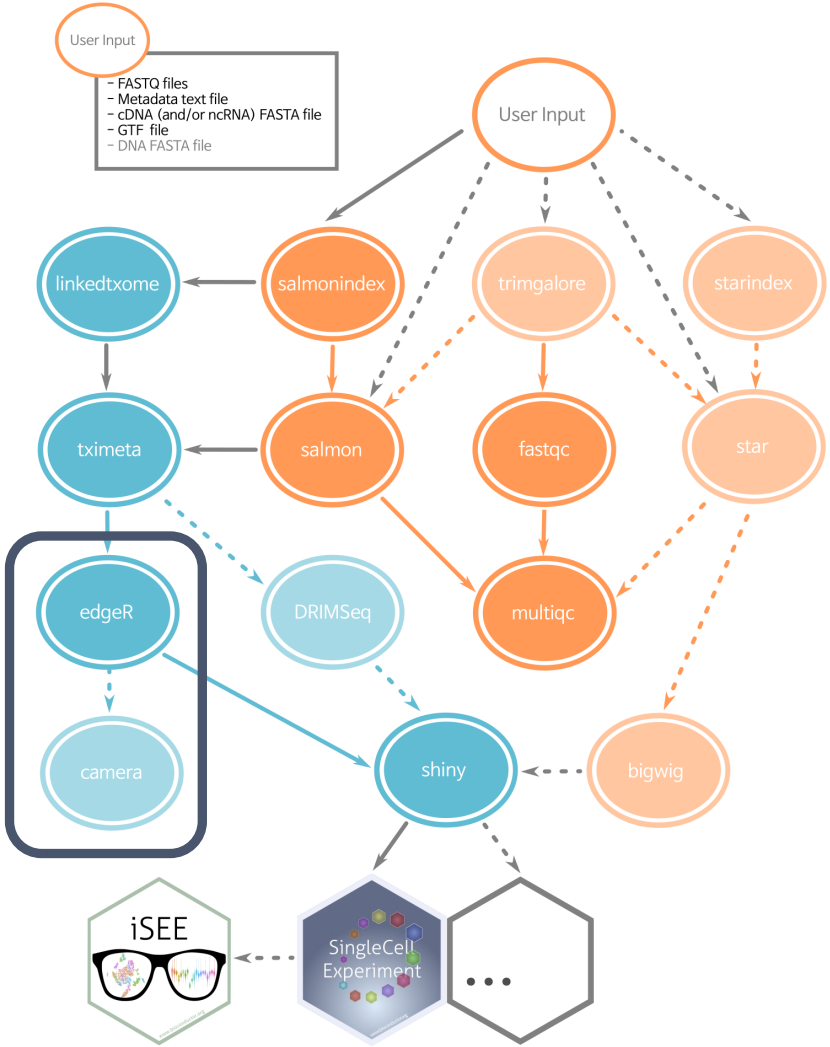
Data import into R



Orjuela, Huang, Hembach et al., G3 2019

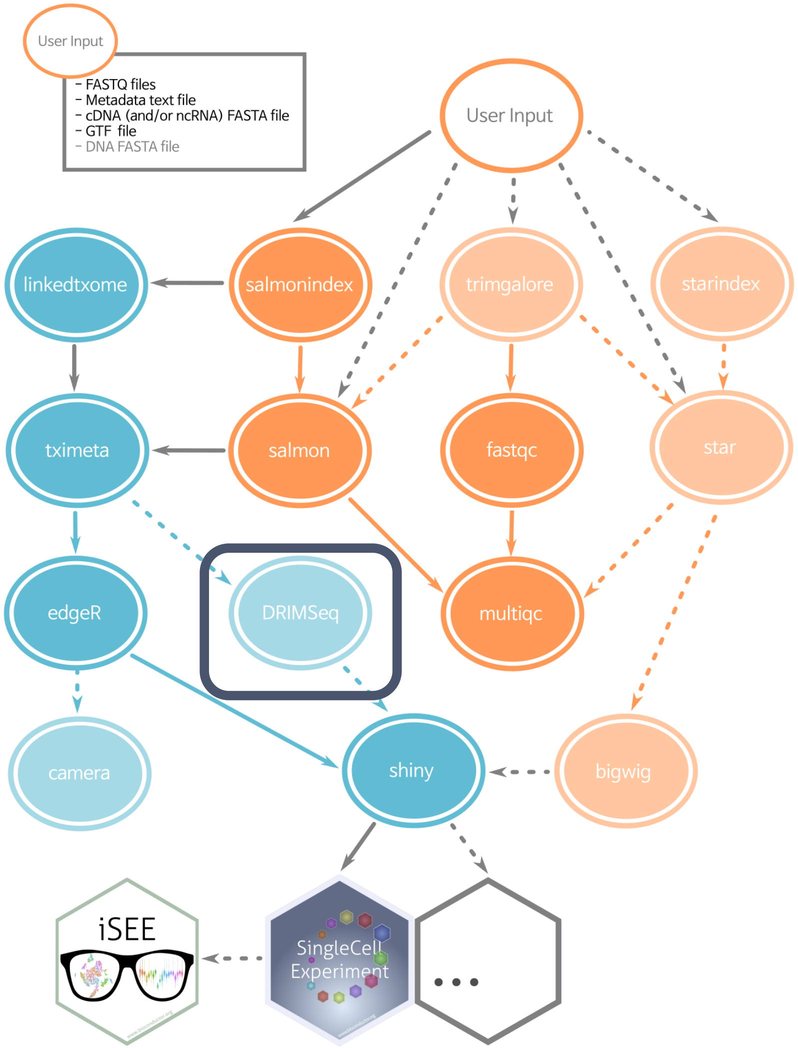
ARMOR workflow

Differential gene expression analysis & gene set enrichment



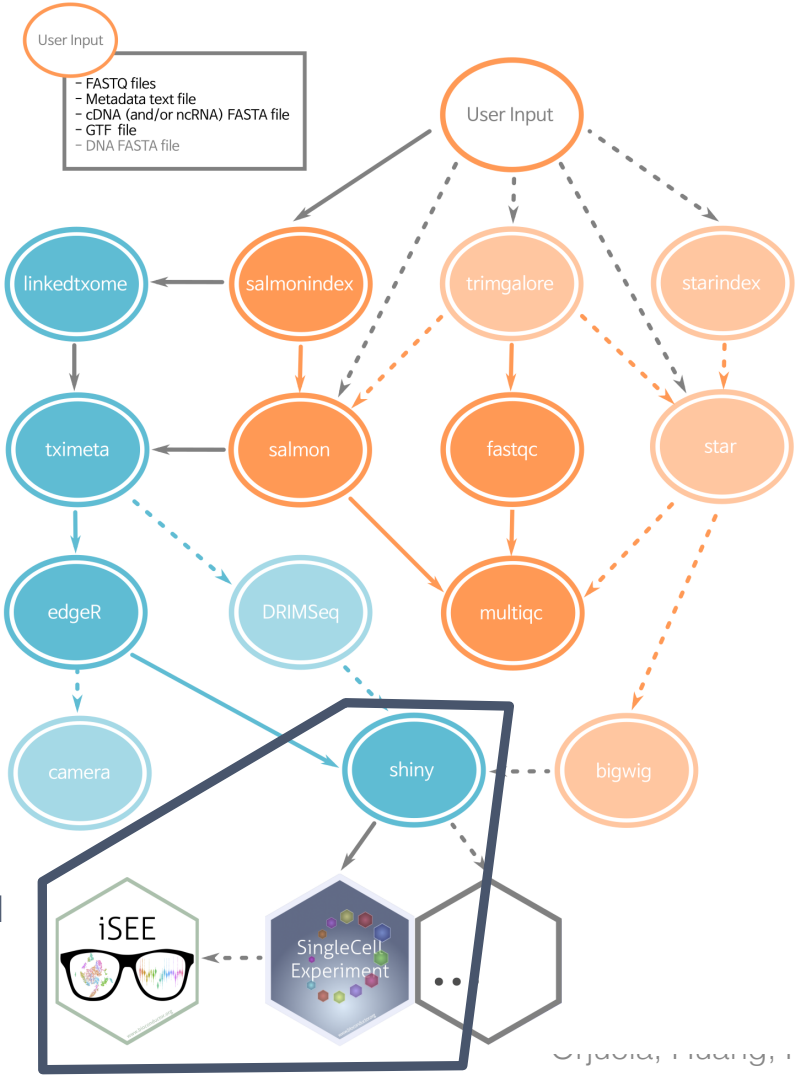
ARMOR workflow

Differential transcript usage



Orjuela, Huang, Hembach et al., G3 2019

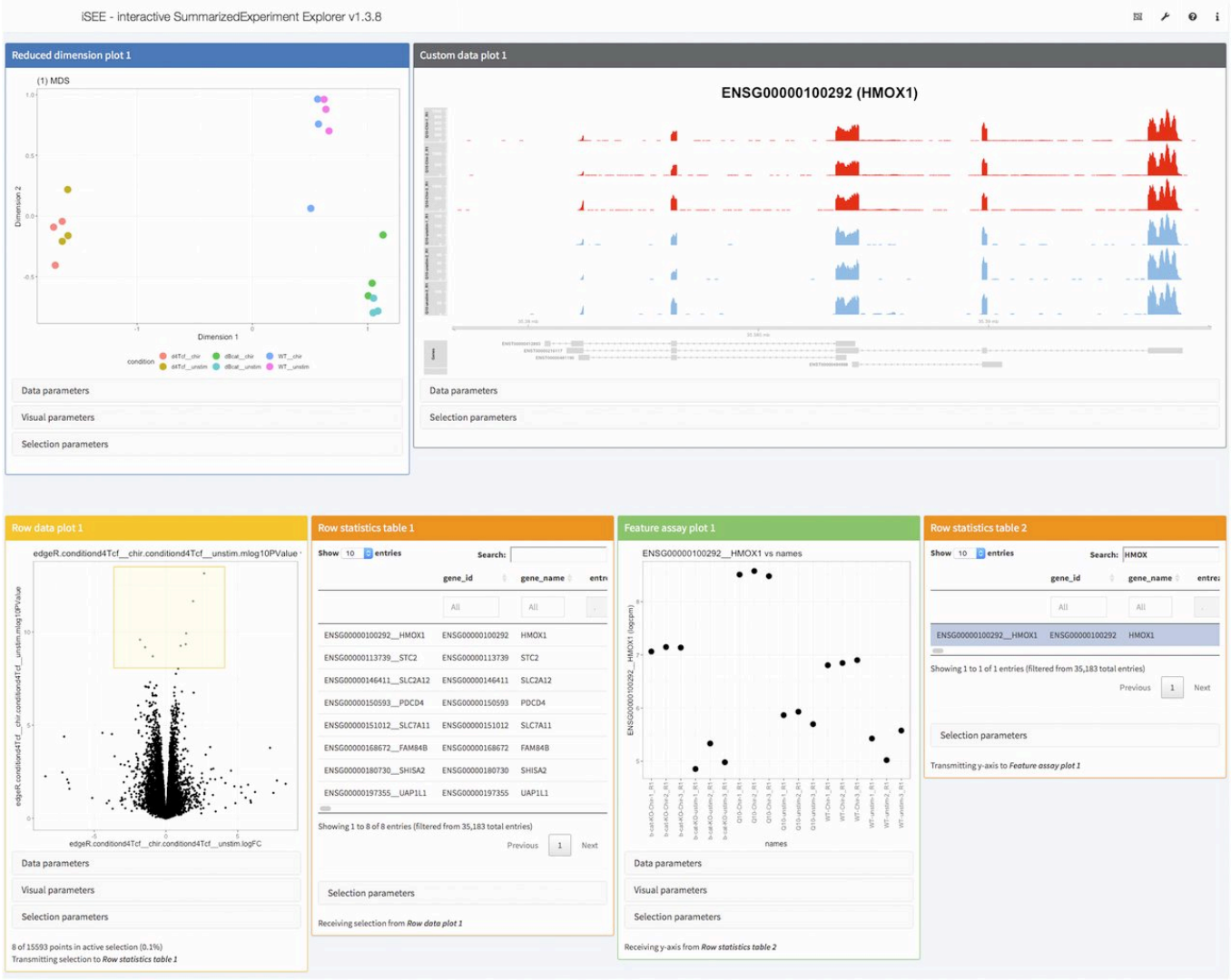
ARMOR workflow



Data export and visualisation

ARMOR

- **A**utomated **R**eproducible **MO**dular **R**NA-seq
- https://github.com/csoneson/ARMOR
- https://www.g3journal.org/content/9/7/2089
- Snakemake workflow
- reproducible, automated, partially contained
- mix of command line tools and R
- Snakefile, configuration file and R scripts
- all software can be installed in conda environments
- visualization with iSEE R package → shiny app
- can be extended by adding rules to the Snakefile



iSEE visualization of the ARMOR output

https://bioconductor.org/packages/release/bioc/html/iSEE.html

Read coverage in integrative genome viewer (IGV)

- Visualize BAM/bigwig files and gene annotations (GTF) in IGV

- Start IGV and load files (BAM needs index)

ARMOR: useful commands

- snakemake –npr to see what snakemake will be executing
- snakemake setup to see if all required software is available
- snakemake checkinputs to see if your specified design and contrast matrix is valid