

Exercise 7 notes

- purely cosmetic: use ``` `{r message=FALSE} ...` to avoid all the startup messages

- in quarto:

```
`` `{r}
```

```
#| label: #fig-atrip-hist  
#| echo: false  
#| error: false  
#| fig-cap: "atrip Histogram and Dens  
#| fig-width: 2  
#| fig-height: 3
```

Load packages

```
1 library("limma")  
2 library("ggplot2")  
3 library("dplyr")
```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

filter, lag

The following objects are masked from 'package:base':

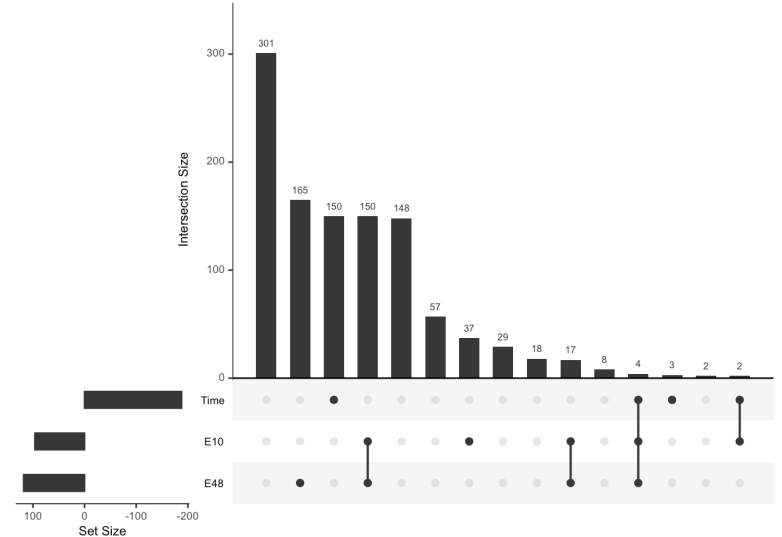
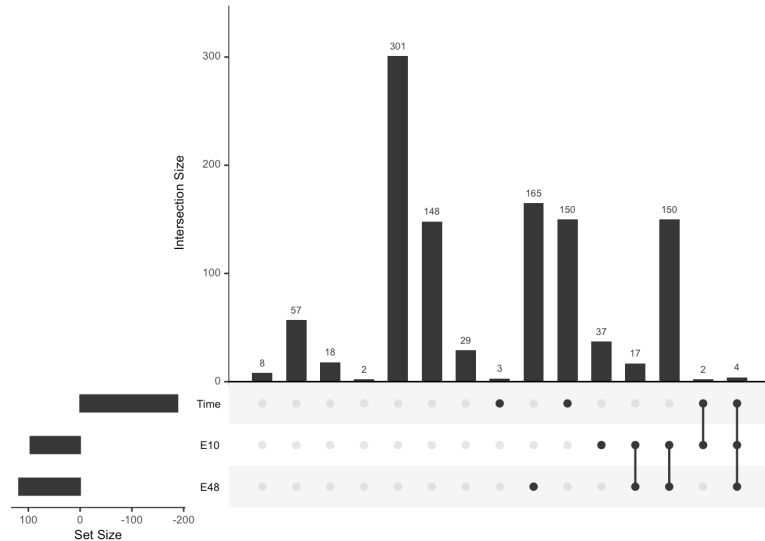
intersect, setdiff, setequal, union

```
1 library("tidyr")
```



Exercise 7 notes

- unusual UpSet plots



RNA-seq data processing and analysis (with ARMOR)



(many slides from Katharina Hembach)

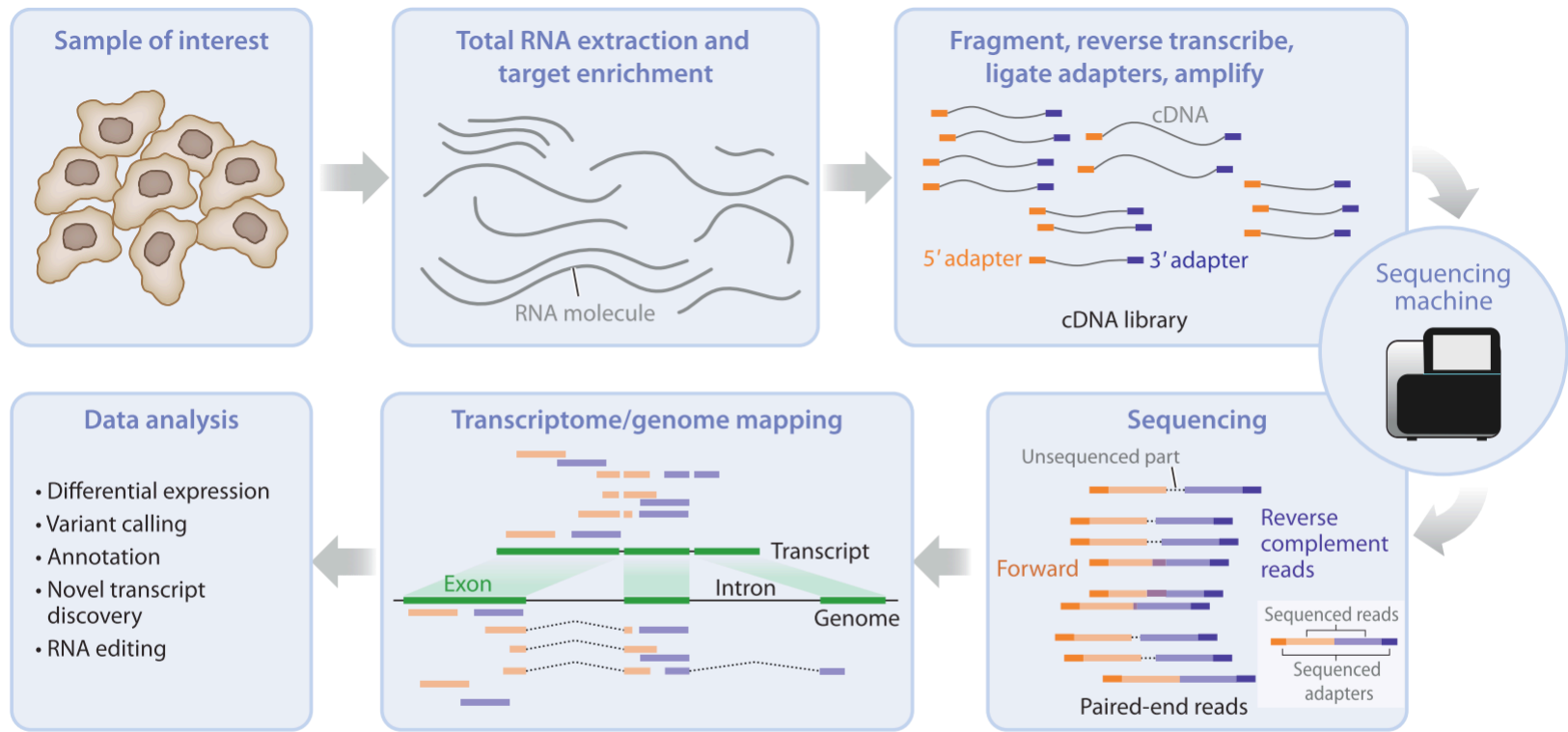


Figure 1

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.

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

- Automated = snakemake

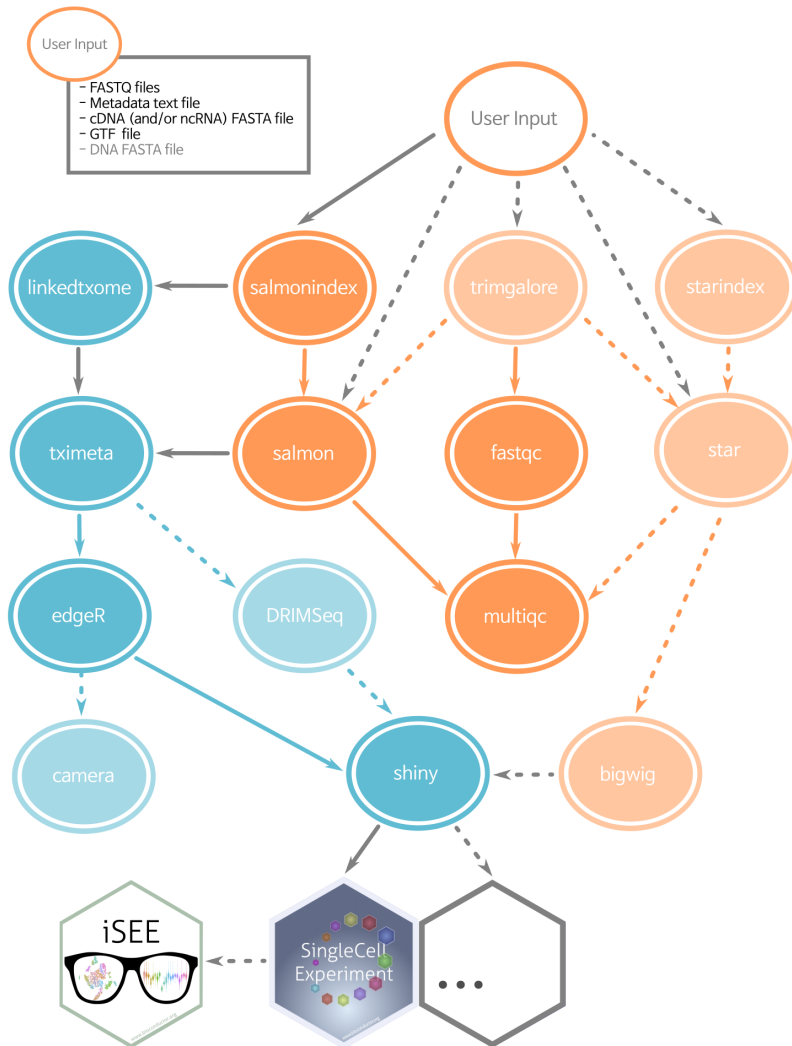


- Reproducible = conda and GitHub



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

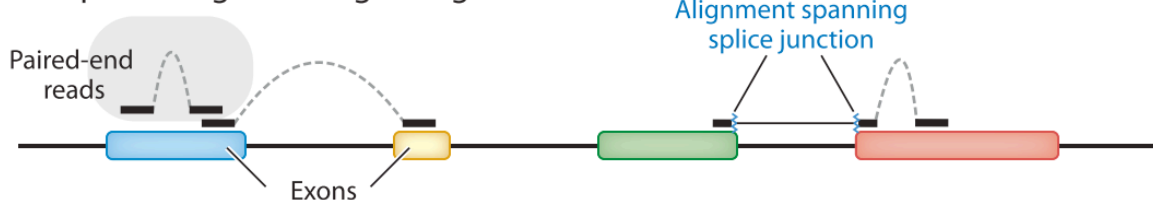
- **FastQC** <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
- **MultiQC** <https://multiqc.info/>
 - aggregates FastQC results from multiple samples, as well as Salmon and STAR output
- # reads, read length, read quality, GC content, % duplicated reads, adapter contamination, ...
- Tools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

Quality control

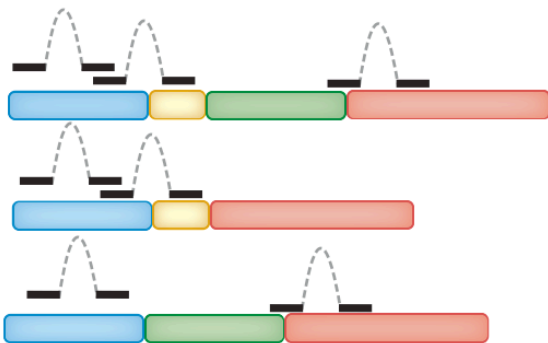
- Depending on FastQC result, you may need to fix (or troubleshoot) QC problems.
- Tools for quality filtering/adapter trimming:
cutadapt, TrimGalore!, Trimmomatic, FASTX-toolkit, ...

Alignment

a Spliced alignment against genome



b Unspliced alignment against transcriptome



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

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

Salmon

<https://combine-lab.github.io/salmon/about/>

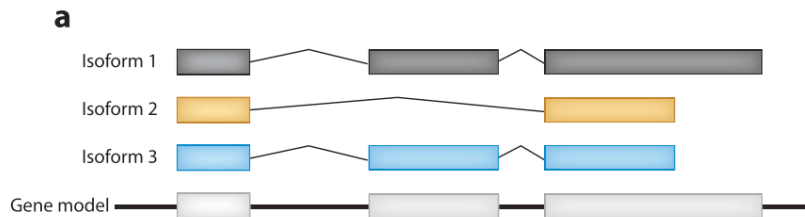
kallisto <https://pachterlab.github.io/kallisto/about>

Alignment

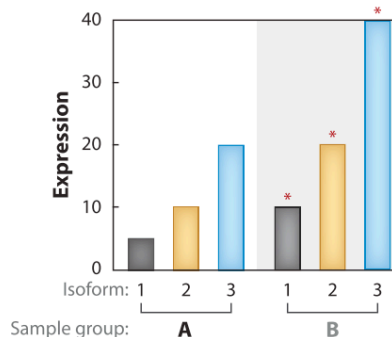
Two possible approaches:

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>

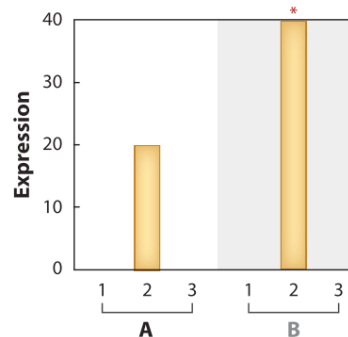
Variants of differential expression



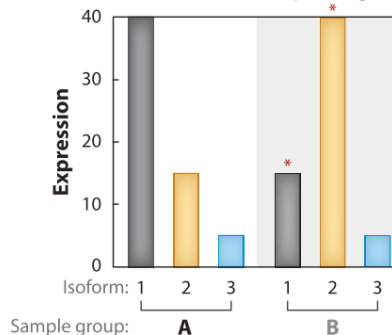
b Differential gene expression



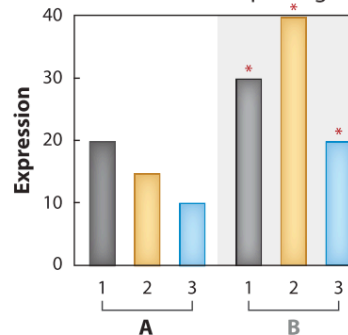
c Differential gene expression



d Differential transcript usage



e Differential gene expression and differential transcript usage



* Differential transcript expression in group B relative to group A

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 Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.



Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.



Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

Docker



```
# list set of local images
```

```
docker image ls
```

```
# retrieve a docker image from dockerhub
```

```
docker pull markrobinsonuzh/docker-for-sta426hs2023: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/docker-for-sta426hs2023:latest
```

```
docker run -it markrobinsonuzh/docker-for-sta426hs2023: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

```
1 rule hello:
2     input:
3         "my_name.txt"
4     output:
5         "hello.txt"
6     shell:
7         "NAME=$(cat {input}); "
8         "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)

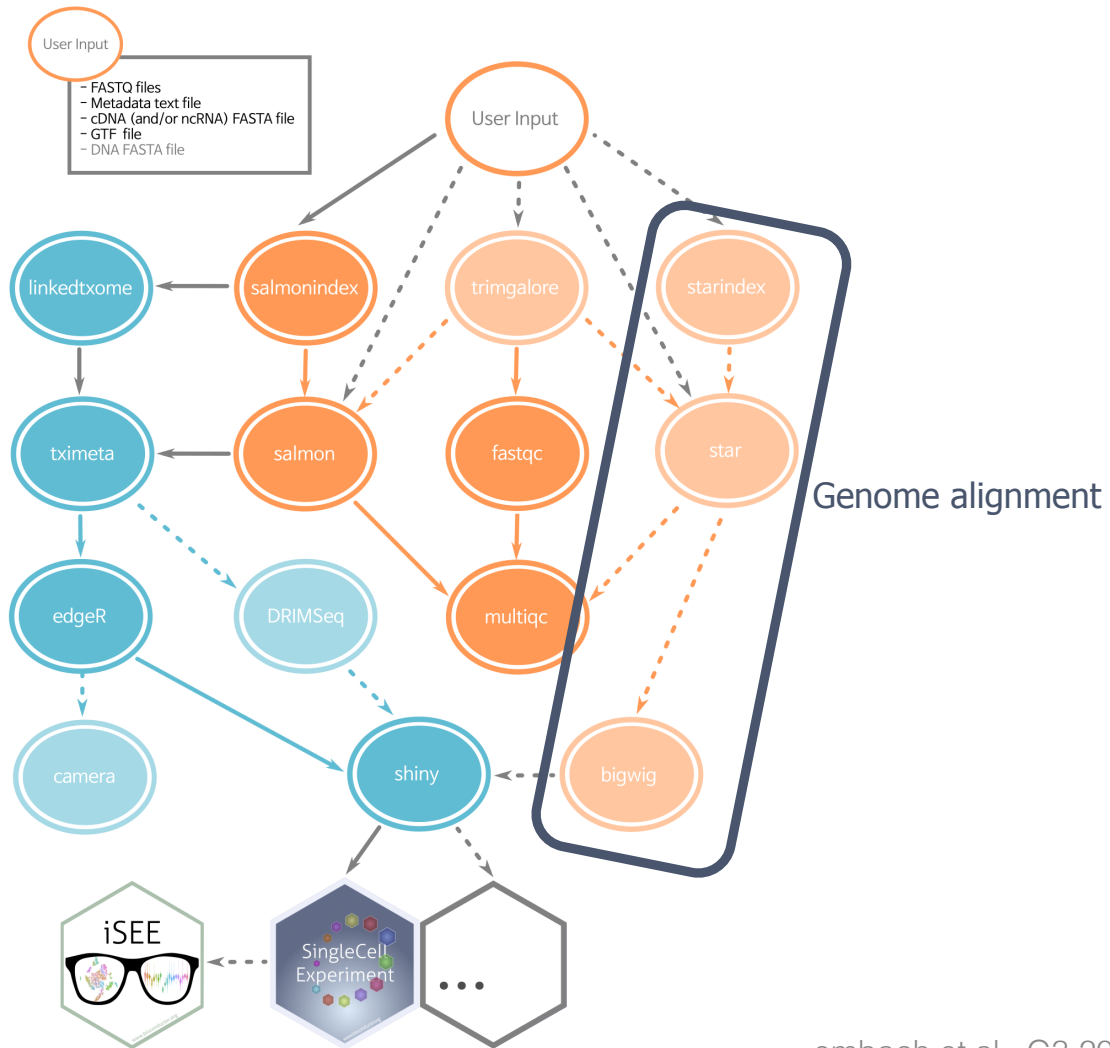
ARMOR WORKFLOW







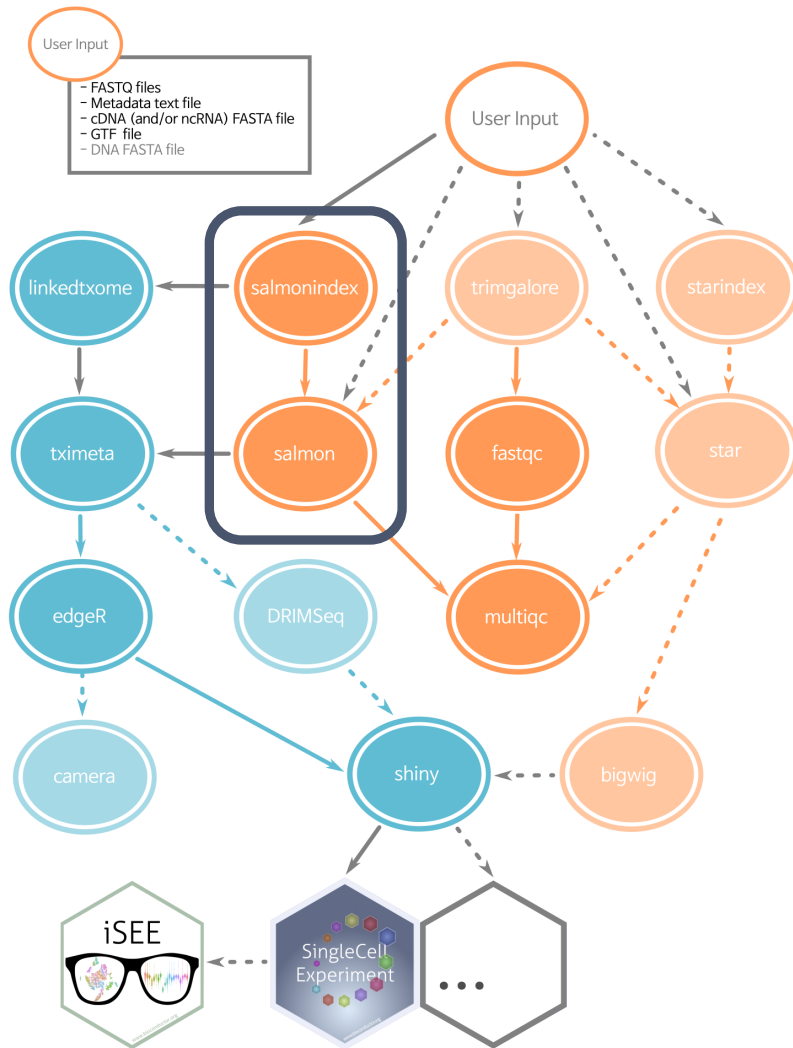
ARMOR workflow





ARMOR workflow

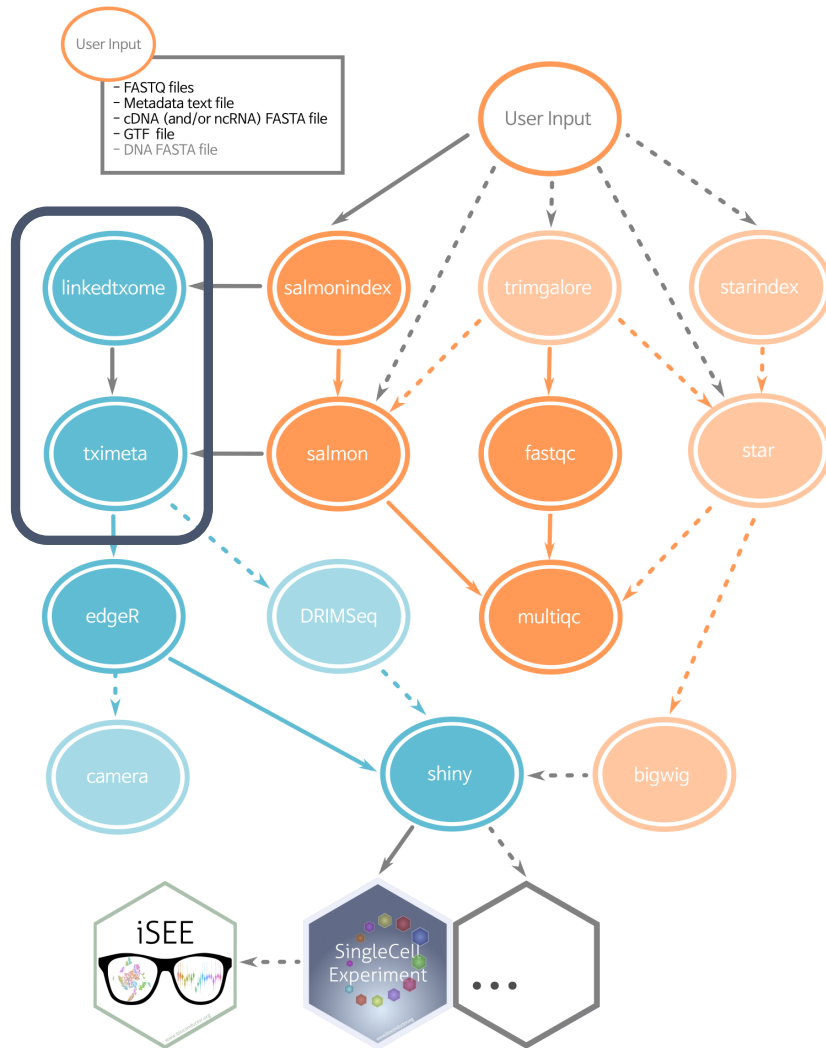
Transcript
abundance
estimation





ARMOR workflow

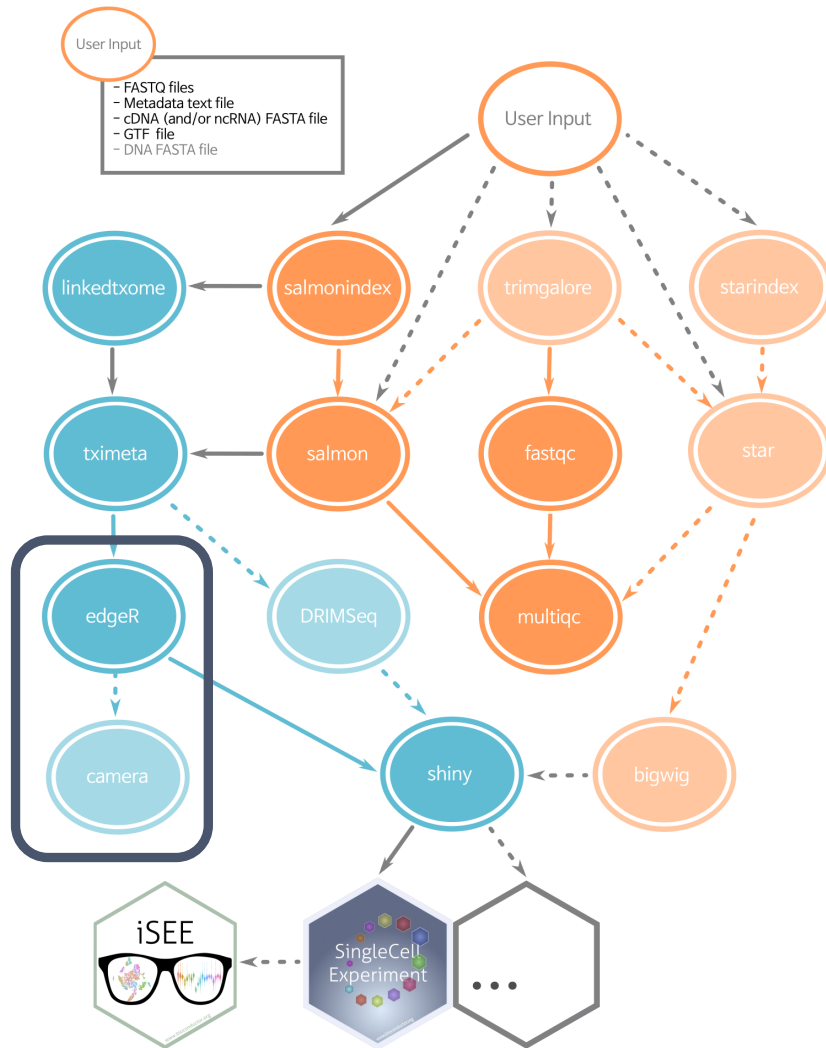
Data import
into R





ARMOR workflow

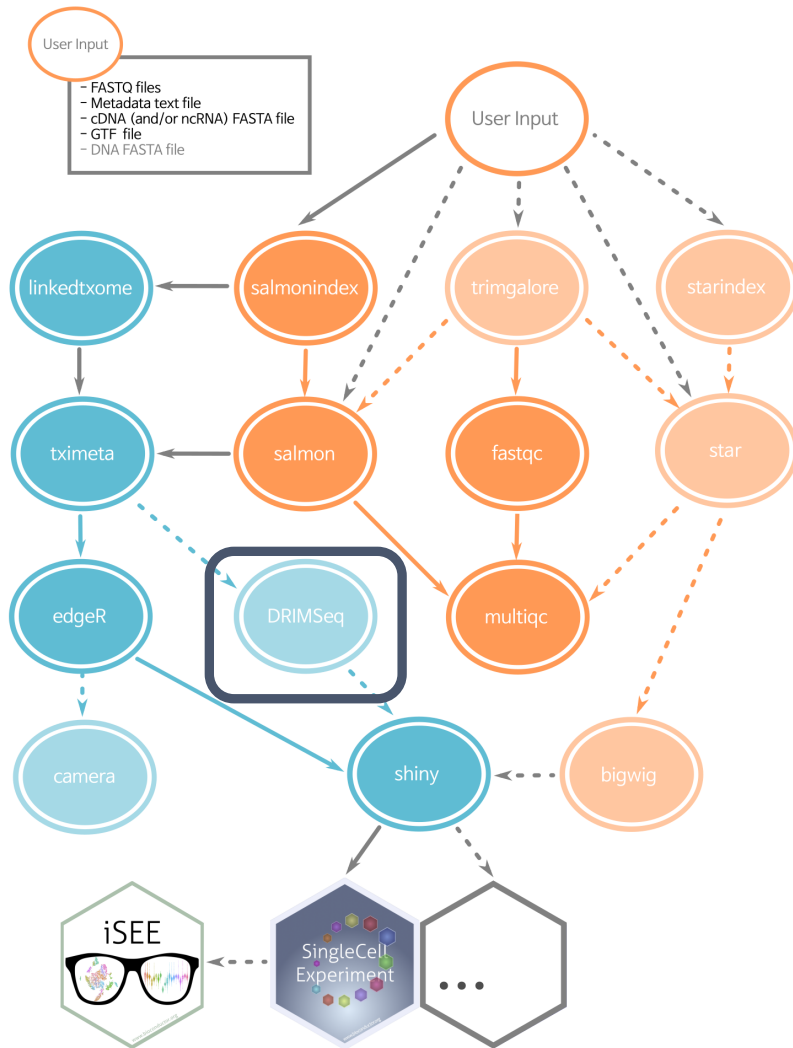
Differential gene
expression analysis
&
gene set enrichment





ARMOR workflow

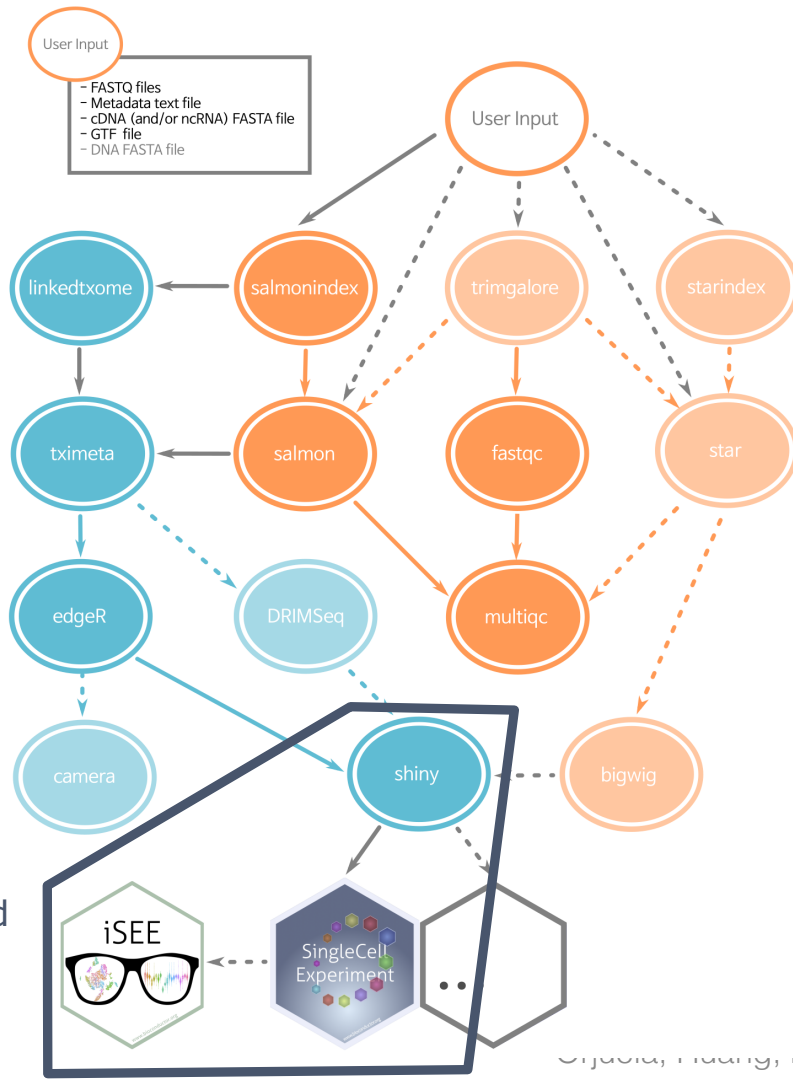
Differential transcript usage





ARMOR workflow

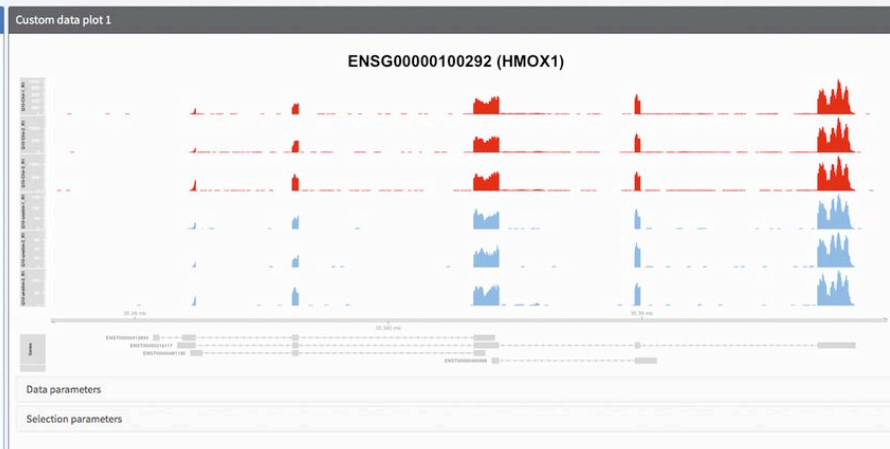
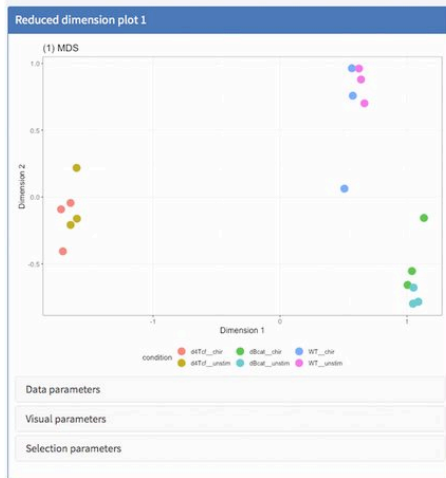
Data export and
visualisation



ARMOR

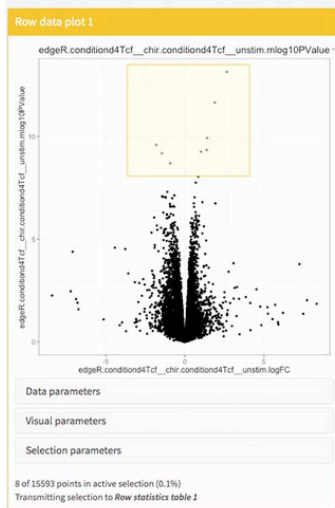


- **A**utomated **R**eproducible **M**odular **R**NA-seq
- <https://github.com/csoneson/ARMOR>
- <https://www.g3journal.org/content/9/7/2089>
- Snakemake workflow
- reproducible, automated, partially contained
- mix of command line tools and R
- Snakefile, configuration file and R scripts
- all software 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](https://bioconductor.org/packages/release/bioc/html/iSEE.html)



Row statistics table 1

Show: 10 entries

Search:

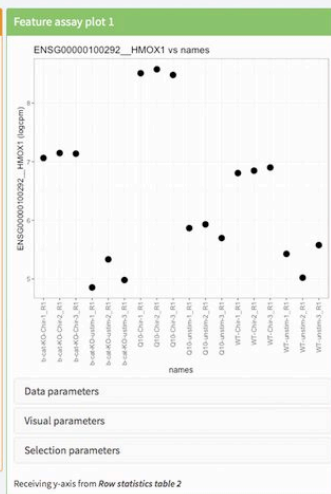
gene_id	gene_name	entri
ENSG00000100292_HMOX1	ENSG00000100292	HMOX1
ENSG00000113739_STC2	ENSG00000113739	STC2
ENSG00000146411_SLC2A12	ENSG00000146411	SLC2A12
ENSG00000150593_PDCD4	ENSG00000150593	PDCD4
ENSG00000151012_SLC7A11	ENSG00000151012	SLC7A11
ENSG00000168672_FAM84B	ENSG00000168672	FAM84B
ENSG00000180730_SHISA2	ENSG00000180730	SHISA2
ENSG00000197355_UAP1L1	ENSG00000197355	UAP1L1

Showing 1 to 8 of 8 entries (filtered from 35,183 total entries)

Previous 1 Next

Selection parameters

Receiving selection from Row data plot 1



Row statistics table 2

Show: 10 entries

Search: HMOX1

gene_id	gene_name	entri
ENSG00000100292_HMOX1	ENSG00000100292	HMOX1

Showing 1 to 1 of 1 entries (filtered from 35,183 total entries)

Previous 1 Next

Selection parameters

Transmitting y-axis to Feature assay plot 1

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