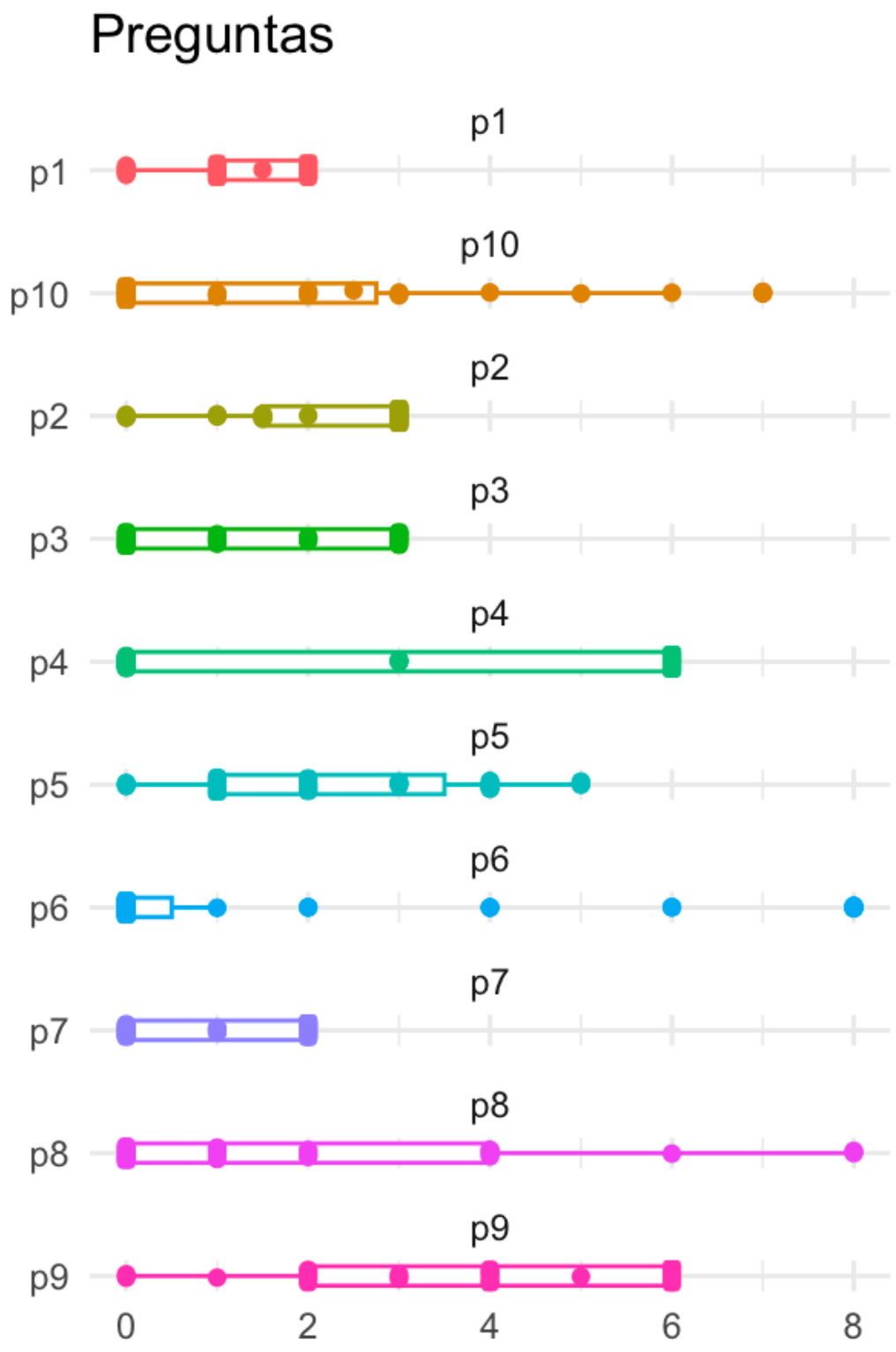
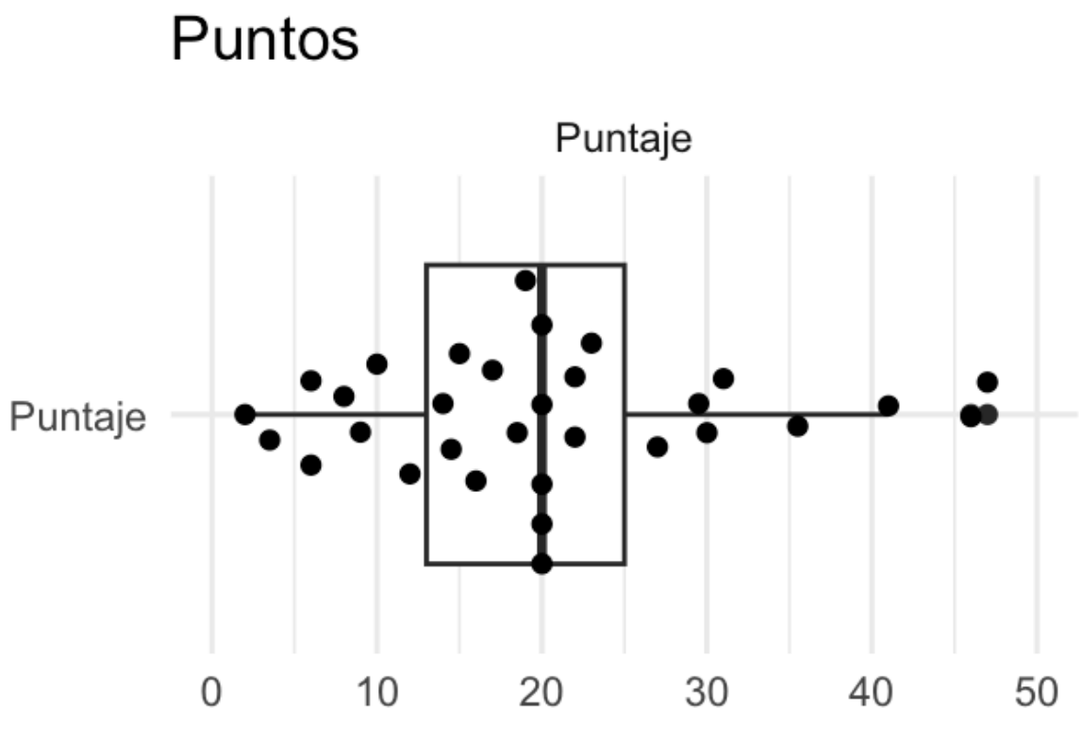
Ejemplos Hadoop y Nextflow

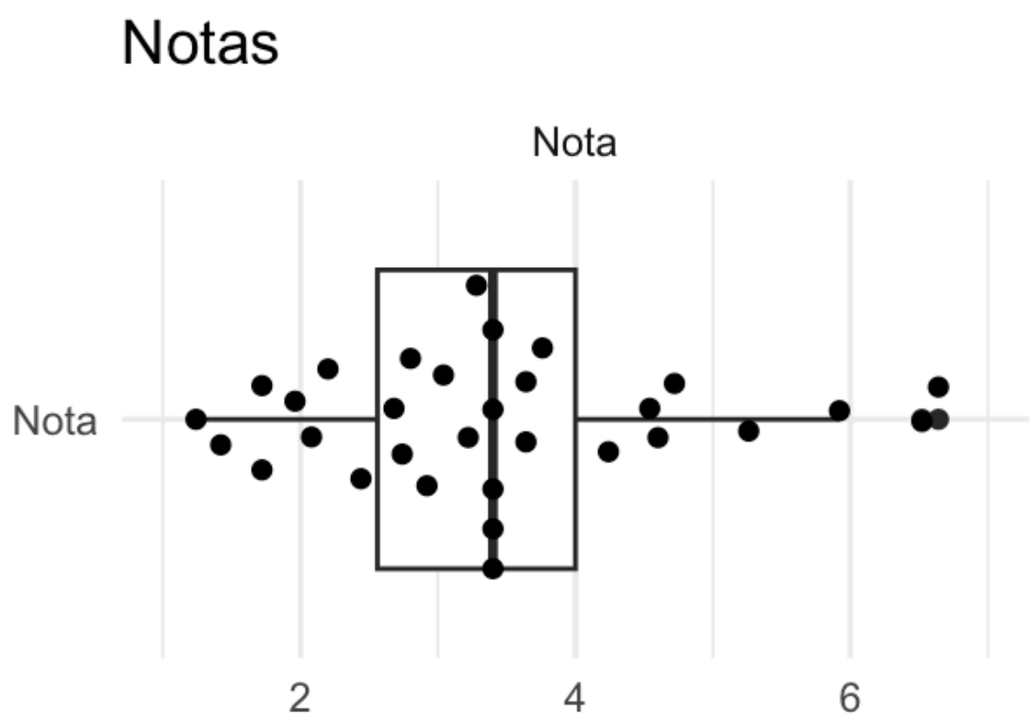
Alex Di Genova

23/10/2023

Notas Control 1



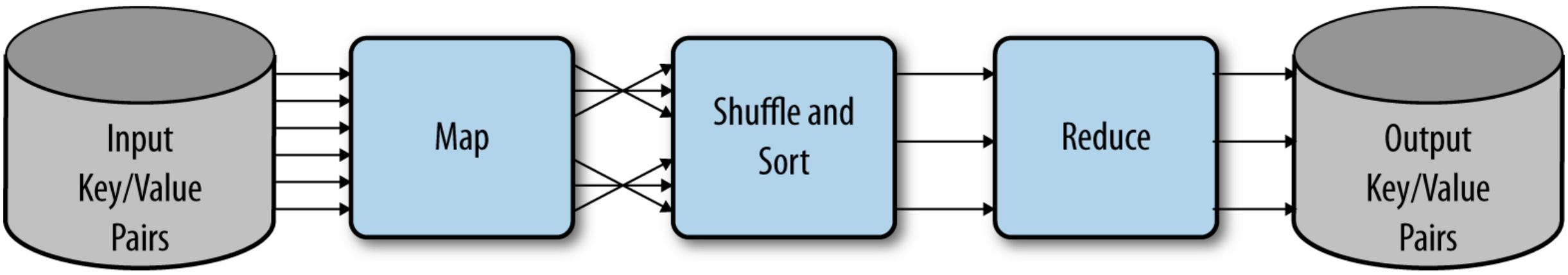





```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.240   2.560   3.400   3.417   4.000   6.640 
```

Hadoop

MapReduce: Un modelo de programación funcional



1. Los datos locales se cargan en un proceso de mapeo como pares clave/valor de HDFS.

2. Los mapeadores generan cero o más pares clave/valor, asignando valores calculados a una clave en particular.

3. Luego, estos pares se ordenan/barajan en función de la clave y luego se pasan a un reductor de modo que todos los valores de una clave estén disponibles.

4. Los reductores deben generar cero o más pares clave/valor finales, que son la salida.

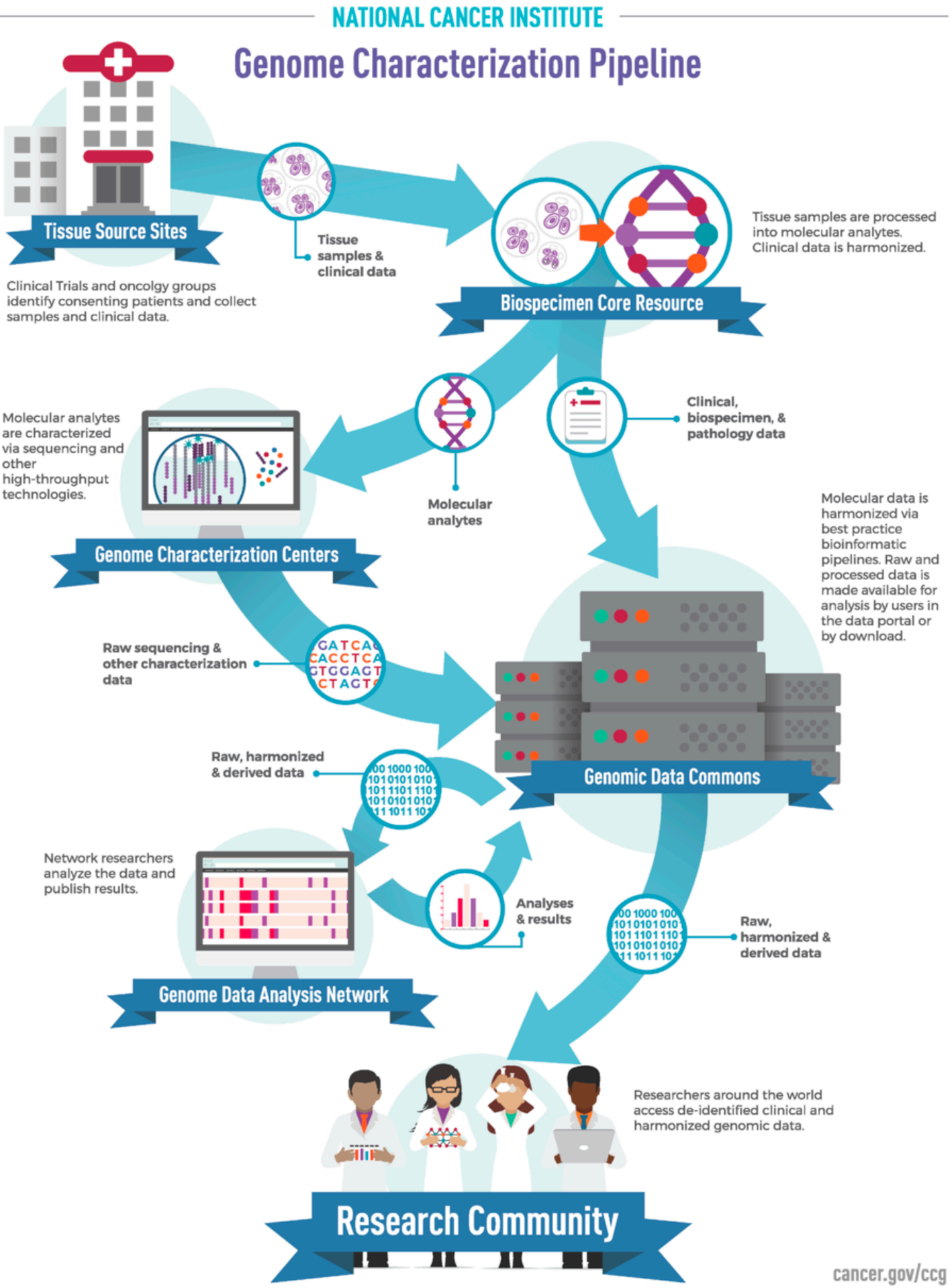
Hadoop
Map/Reduce

```
def map(dockey, line):
    for word in value.split():
        emit(word, 1)
```

```
def reduce(word, values):
    count = sum(value for value in values)
    emit(word, count)
```

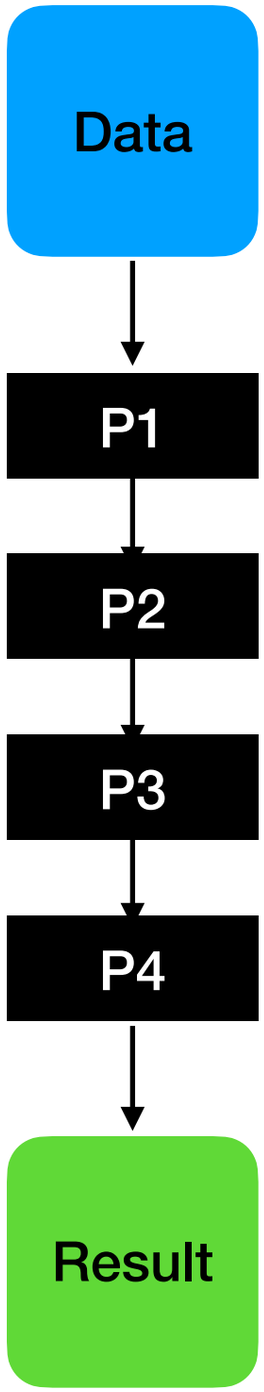
```
# Input to WordCount mappers

(27183, "The fast cat wears no hat.")
(31416, "The cat in the hat ran fast.")

# Mapper 1 output

("The", 1), ("fast", 1), ("cat", 1), ("wears", 1),
("no", 1), ("hat", 1),(".", 1)

# Mapper 2 output

("The", 1), ("cat", 1), ("in", 1), ("the", 1),
("hat", 1), ("ran", 1),("fast", 1),(".", 1)


# Input to WordCount reducers
# This data was computed by shuffle and sort

(., [1, 1])
("cat", [1, 1])
("fast", [1, 1])
("hat", [1, 1])
("in", [1])
("no", [1])
("ran", [1])
("the", [1])
("wears", [1])
("The", [1, 1])

# Output by all WordCount reducers

(., 2)
("cat", 2)
("fast", 2)
("hat", 2)
("in", 1)
("no", 1)
```

https://github.com/adigenova/uohpmd/blob/main/code/Haddop_nextflow.ipynb

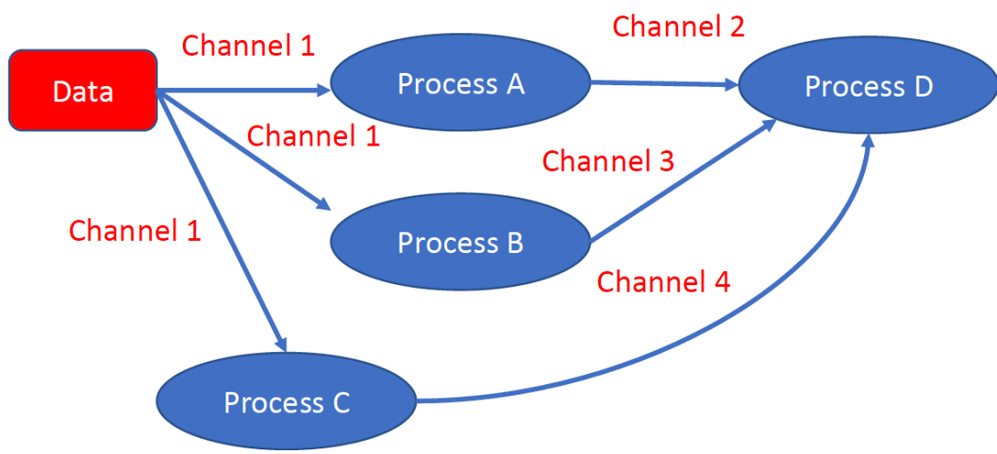


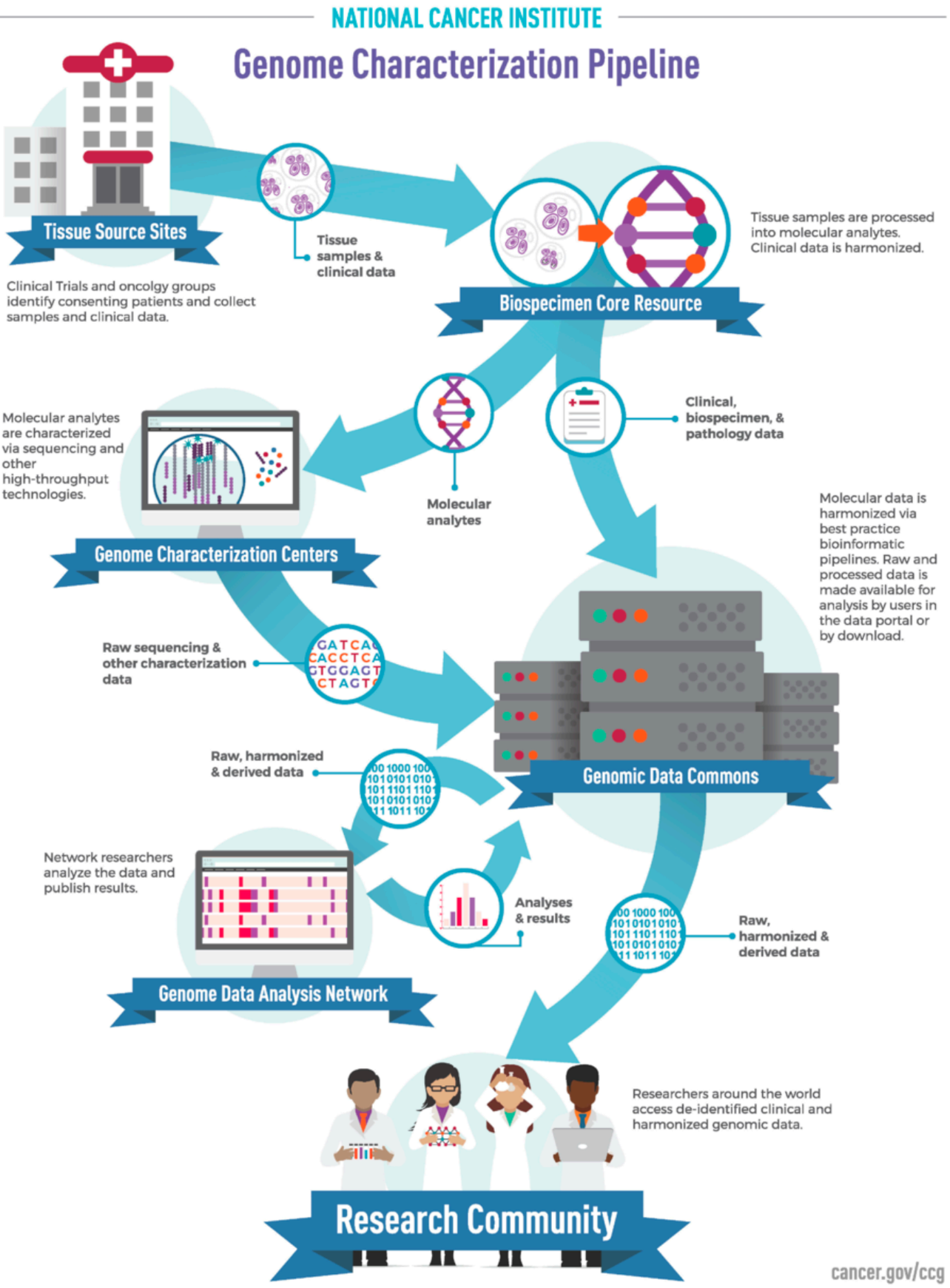
Nextflow

Genómica



Workflow





Nextflow

Genómica

Requisitos de pipelines para proyectos genómicos a gran escala

- **Reproducible**: los resultados genómicos deben ser totalmente reproducibles
- **Escalable**: Facil ejecucion en cluster HPC o cloud.
- **Portátil**: puede ejecutarse en varias infraestructuras (diferente sistema operativo, cloud)
- **Manejar la heterogeneidad**: funciona con dependencias de software en conflicto y varios requisitos de recursos.

Nextflow

Primer pipeline

Procesos

```
//pipeline_01.nf
nextflow.enable.dsl=2

process INDEX {
    input:
      path transcriptome
    output:
      path 'index'
    script:
      """
      salmon index -t $transcriptome -i index
      """
}

 process QUANT {
    input:
      each  path(index)
      tuple(val(pair_id), path(reads))
    output:
      path pair_id
    script:
      """
      salmon quant --threads $task.cpus --libType=U \
          -I $index \
          -1 ${reads[0]} -2 ${reads[1]} -o $pair_id
      """
}

//Definición del pipeline
workflow {
     //canales de entrada
    transcriptome_ch =
     channel.fromPath('transcriptome/*.fa.gz',checkIfExists: true)
    read_pairs_ch =
     channel.fromFilePairs('reads/*_{1,2}.fq.gz',checkIfExists: true)
    //el proceso INDEX toma el calar transcriptome_ch
    index_ch = INDEX(transcriptome_ch)
    //el proceso QUANT toma dos canales como argumentos: Indice y las lecturas
    QUANT( index_ch, read_pairs_ch ).view()
}
```

https://github.com/adigenova/uohpmd/blob/main/code/NextflowI.ipynb

Preguntas?