A framework for pipeline benchmarking and its application to scRNAseq clustering

Pierre-Luc Germain$^{1,2}$, Anthony Sonrel$^1$, Mark D. Robinson$^1$

1. Lab of Statistical Bioinformatics (Robinson group), University of Zürich
2. D-HEST Institute for Neuroscience, ETH Zürich
A simple scRNAseq clustering pipeline

1. Filtering
2. Normalization
3. Feature selection
4. Dimensionality reduction
5. Clustering
We define a pipeline as, minimally, a list of functions to be successively applied on the output of the previous one

```r
pipeline <- PipelineDefinition(
  list(
    step1= function(x, s1_method, param1){
      ## processing, e.g.:
      x <- some_function(x, s1_method, param1)
      return(x)
    },
    step2= function(x, s2_method, param2, param3){
      ## processing, e.g.:
      get(s2_method)(x, param2, param3)
    },
    ...,
    stepN= function(x, param4){
      ## processing
    }
  )
)
```
We define a pipeline as, minimally, a list of functions to be successively applied on the output of the previous one.

```r
pipeline <- PipelineDefinition(
  list(
    step1=function(x, s1_method, param1){
      ## processing, e.g.:
      x <- some_function(x, s1_method, param1)
      return(x)
    },
    step2=function(x, s2_method, param2, param3){
      ## processing, e.g.:
      get(s2_method)(x, param2, param3)
    },
    ...
    stepN=function(x, param4){
      ## processing
    }
  )
)
```

Optionally, the `PipelineDefinition` can include `evaluation` functions for some steps.

```r
stepFn(pipeline, step="step1", type="evaluation") <- function(x){
  # produce some evaluation metric based on 'x'
}
```
We define a pipeline as, minimally, a list of functions to be successively applied on the output of the previous one.

```r
pipeline <- PipelineDefinition(
  list(
    step1=function(x, s1_method, param1){
      ## processing, e.g.:
      x <- some_function(x, s1_method, param1)
      return(x)
    },
    step2=function(x, s2_method, param2, param3){
      ## processing, e.g.:
      get(s2_method)(x, param2, param3)
    },
    ...,
    stepN=function(x, param4){
      ## processing
    }
  )
)

stepFn(pipeline, step="step1", type="evaluation") <- function(x){
  # produce some evaluation metric based on 'x'
}
```

Optionally, the `PipelineDefinition` can include evaluation functions for some steps.

```
data <- data.frame(
  dataset  = c("dataset 1"),
  s1_method = c("function_A", "function_B"),
  param1   = c(5, 5),
  metric1  = c("...", "...")
)
```
We define a pipeline as, minimally, a list of functions to be successively applied on the output of the previous one

```r
pipeline <- PipelineDefinition(
  list(
    step1=function(x, s1_method, param1){
      ## processing, e.g.:
      x <- some_function(x, s1_method, param1)
      return(x)
    },
    step2=function(x, s2_method, param2, param3){
      ## processing, e.g.:
      get(s2_method)(x, param2, param3)
    },
    ...
    stepN=function(x, param4){
      ## processing
    }
  )
)

alternatives = list(
  s1_method=c("function_A", "function_B"),
  param1=5,
  s2_method=c("function_C", "function_D"),
  param2=FALSE,
  param3=50,
  param4=FALSE
)

runPipeline(datasets, alternatives, pipDef=pipeline)
```
Basic Seurat-based *PipelineDefinition*:

```r
> pipeline <- scrna_seurat_pipeline()
> pipeline
A PipelineDefinition object with the following steps:
- `doublet(x, doubletMethod)` *
  Takes a SCE object with the `phenoid` colData column, passes it through the
  function `doubletMethod`, and outputs a filtered SCE.
- `filtering(x, filt)` *
  Takes a SCE object, passes it through the function `filt`, and outputs a
  filtered Seurat object.
- `normalization(x, norm)` *
  Passes the object through function `norm` to return the object with the
  normalized and scale data slots filled.
- `selection(x, sel, selnb)`
  Returns a seurat object with the VariableFeatures filled with `selnb` features
  using the function `sel`.
- `dimreduction(x, dr, maxdim)` *
  Returns a seurat object with the PCA reduction with up to `maxdim` components
  using the `dr` function.
- `clustering(x, clustmethod, dims, k, steps, resolution, min.size)` *
  Uses function `clustmethod` to return a named vector of cell clusters.
```

and its application to 9 datasets with true cell labels...
Filtering:

Tradeoff between classification accuracy and the proportion/bias of the excluded cells.
### Normalization & Scaling

Residual correlation with technical covariates, separability of the subpopulations (silhouette)

<table>
<thead>
<tr>
<th>Residual corr with log10_total_counts</th>
<th>Residual corr with log10_total_features</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (no scaling)</td>
<td>none</td>
</tr>
<tr>
<td>seurat (no scaling)</td>
<td>seurat (no scaling)</td>
</tr>
<tr>
<td>seurat.feat_mt_regress</td>
<td>seurat.feat_regress</td>
</tr>
<tr>
<td>scan (no scaling)</td>
<td>scan</td>
</tr>
<tr>
<td>stableG</td>
<td>stableG</td>
</tr>
<tr>
<td>stableG.nucleus</td>
<td>stableG.sum</td>
</tr>
<tr>
<td>seurat</td>
<td>seurat</td>
</tr>
<tr>
<td>seurat.mt_regress</td>
<td>seurat.mt_regress</td>
</tr>
<tr>
<td>sctransform</td>
<td>sctransform</td>
</tr>
</tbody>
</table>

Averaging silhouette width per subpopulation:

- sctransform (no scaling)
- scan (no scaling)
- seurat (no scaling)
- scan
- seurat
- stableG
- stableG.nucleus
- stableG.sum
- none (no scaling)
- seurat.feat_regress
- seurat.mt_regress

sctransform wins...
Feature selection:

Does the ranking / selection track genes whose variance is between subpopulations?
Estimating the number of dimensions
Denoising/Imputation

(A) Mean silhouette width

(B) ARI at true nb clusters, Mean precision, Mean recall

(with Anthony Sonrel)
Meet me at the (not-quite-up-to-date) **poster** for feedback/discussion!