SBG-CGC: R api and the Common Worflow Language

Tiffany Delhomme

IARC course - analysing TCGA data in the cloud

28 feb 2018

International Agency for Research on Cancer
World Health Organization

Summary

Why learning CWL?

It is the language used by the SBG-CGC

Tools and workflows in the SBG-CGC:

- need to be **"coded"** in a specific language
  - This is the Common Workflow Language

Why learning CWL?

It is the language used by the SBG-CGC

Tools and workflows in the SBG-CGC:

- need to be **"coded"** in a specific language
  - This is the Common Workflow Language
- are described in a particular format, representing structured informations
  - This is JSON or YAML format (focus on JSON)

Why learning CWL?

It is the language used by the SBG-CGC

Tools and workflows in the SBG-CGC:

- need to be **"coded"** in a specific language
  - This is the Common Workflow Language
- are described in a particular format, representing structured informations
  - This is JSON or YAML format (focus on JSON)
- executed from JSON files are reproducible
  - there is a trace, we can have versions hosted on GitHub
  - the tool is portable into different environments

What is Common Workflow Language?

- CWL is a **specification** to describe command line **tools**
  - tools can be connected together to create **workflows**
  - tools and workflows are **portable** across multiple computing environments (reproducibility)

What is Common Workflow Language?

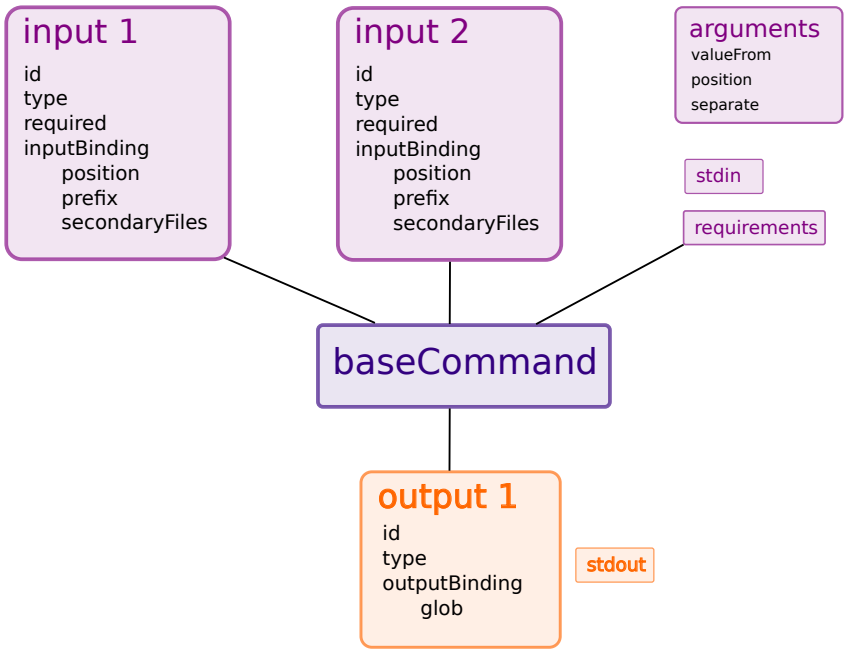
- CWL is a **specification** to describe command line **tools**
  - tools can be connected together to create **workflows**
  - tools and workflows are **portable** across multiple computing environments (reproducibility)
- CWL tasks are **isolated**
  - user must be **explicit** in term of inputs/outputs
  - benefit of explicitness and isolation are **flexibility**, **portability**, and **scalability**

How to define a tool with CWL?

- CWL documents consist of an array of objects represented using JSON or YAML syntax

How to define a tool with CWL?

- CWL documents consist of an array of objects represented using JSON or YAML syntax
- a tool is a set of objects, with their own properties



International Agency for Research on Cancer
World Health Organization

First example: extract header with samtools

JSON CWL document

CWL specification is here: http://www.commonwl.org/draft-2

JSON file is in GitHub: IARCbioinfo/SBG-CGC_course2018

```
"class": "CommandLineTool",
"baseCommand": "samtools view",
"label": "samtools_header",
"id": "samtools_header_json",
"sbg:contributors": [
        "tdelhomme"
],
```

First example: extract header with samtools

JSON CWL document

```
    "inputs": [
{
  "type": "File",
  "description": "BAM file to extract the header",
  "label": "input BAM file",
  "id": "#input_BAM",
  "required" : true,
  "inputBinding": {
    "secondaryFiles" : "^.bai",
    "separate": true,
    "sbg:cmdInclude": true,
    "position": 0,
    "prefix": ""
  }
},
```

First example: extract header with samtools

JSON CWL document

```
    {
  "type": "string",
  "id": "#output_file_name",
  "required" : true
  }
]
```

First example: extract header with samtools

JSON CWL document

```
    "outputs": [
  {
    "id": "#header_output",
    "type": "File",
    "outputBinding": {
      "glob": {
        "engine": "#cwl-js-engine",
        "class": "Expression",
        "script": "$job.inputs.output_file_name"
      }
    }
  }
],
```

First example: extract header with samtools

JSON CWL document

```
"stdout": {
  "engine": "#cwl-js-engine",
  "class": "Expression",
  "script": "$job.inputs.output_file_name"
}
```

First example: extract header with samtools

JSON CWL document

```
      "requirements": [
  {
    "requirements": [
      {
        "dockerPull": "rabix/js-engine",
        "class": "DockerRequirement"
      }
    ],
    "class": "ExpressionEngineRequirement",
    "id": "#cwl-js-engine"
  }
]
```

First example: extract header with samtools

JSON CWL document

```
"hints": [
    {
      "class": "sbg:CPURequirement",
      "value": 1
    },
    {
      "class": "sbg:MemRequirement",
      "value": 1000
    },
    {
      "dockerImageId": "",
      "class": "DockerRequirement",
      "dockerPull": "biocontainers/samtools"
    }
  ],
```

First example: extract header with samtools

JSON CWL document

```
"arguments": [
  {
    "separate": true,
    "position": -1,
    "valueFrom": "-H"
  }
]
```

sevenbridges-r R package

installation

- the **sevenbridges-r** is an R/Bioconductor package that provides an interface for the SBG-CGC public API

sevenbridges-r R package

installation

- the **sevenbridges-r** is an R/Bioconductor package that provides an interface for the SBG-CGC public API
- installation from bioconductor

```
source("https://bioconductor.org/biocLite.R")
biocLite("sevenbridges")
```

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

Proposed protocol:

- authentification with the token from CGC interface: **Auth()**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

Proposed protocol:

- authentification with the token from CGC interface: **Auth()**
- get the data in the R environment
  - get the project we will work on: **Auth()$project()**
  - get the files we we work on: **Auth()$project$file()**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

Proposed protocol:

- authentification with the token from CGC interface: **Auth()**
- get the data in the R environment
  - get the project we will work on: **Auth()$project()**
  - get the files we we work on: **Auth()$project$file()**
- create tool from JSON file (tool definition in CWL): **convert_app()**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

Proposed protocol:

- authentification with the token from CGC interface: **Auth()**
- get the data in the R environment
  - get the project we will work on: **Auth()$project()**
  - get the files we we work on: **Auth()$project$file()**
- create tool from JSON file (tool definition in CWL): **convert_app()**
- add the app to the project: **p$app_add()**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

Proposed protocol:

- authentification with the token from CGC interface: **Auth()**
- get the data in the R environment
  - get the project we will work on: **Auth()$project()**
  - get the files we we work on: **Auth()$project$file()**
- create tool from JSON file (tool definition in CWL): **convert_app()**
- add the app to the project: **p$app_add()**
- create a task **p$task_add()** and run it **tsk$run()**

sevenbridges-r R package

How would we use it?

- object-oriented design: **objects** and its **attributes** and **methods**

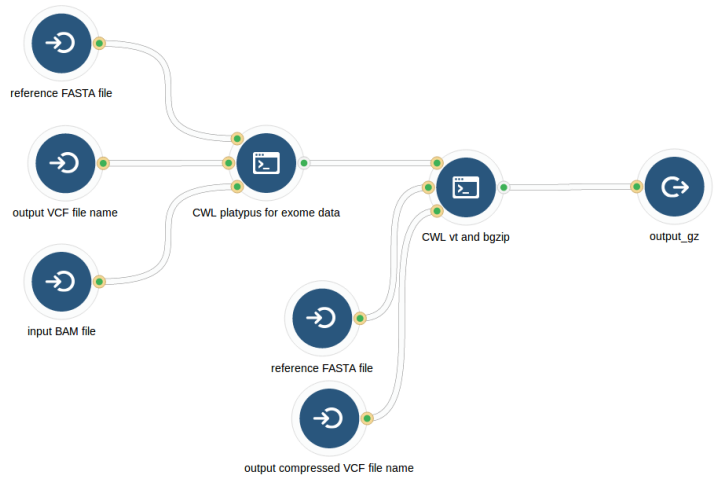
Proposed protocol:

- authentification with the token from CGC interface: **Auth()**
- get the data in the R environment
  - get the project we will work on: **Auth()$project()**
  - get the files we we work on: **Auth()$project$file()**
- create tool from JSON file (tool definition in CWL): **convert_app()**
- add the app to the project: **p$app_add()**
- create a task **p$task_add()** and run it **tsk$run()**

sevenbridges-r R package

How would we use it?

In case of a workflow, *i.e.* connected tools by inputs/outputs:



Connect tools with function **link()**:

```
my_workflow = link(tool1, tool2,
    "#output_id_tool1", "#input_id_tool2")
```

International Agency for Research on Cancer

World Health Organization

sevenbridges-r R package

example: extract header with samtools on one BAM

1. Define your project

```
library("sevenbridges")

project="iarc-course-tutorial"

a <- Auth(token = "***",
          url = "https://cgc-api.sbgenomics.com/v2/")

p <- a$project(id = paste("tdelhomme/", project, sep=""))
```

sevenbridges-r R package

example: extract header with samtools on one BAM

2. Define the file you want to run your tool on

```
my_bam = p$file("bam", complete = TRUE)[1]
```

complete parameter force to extract all the files (by default: 50 first)

sevenbridges-r R package

example: extract header with samtools on one BAM

3. Add the tool to your project from the JSON

```
f = "header.json"

samtools_header = convert_app(f)

p$app_add("header_Rapi", samtools_header)
```

sevenbridges-r R package

example: extract header with samtools on one BAM

4. Add and run the task

```
tsk = p$task_add(name = "test_header",
      description = "samtools header for IARC course",
      app = "tdelhomme/iarc-course-tutorial/
             header_Rapi",
      inputs = list(input_BAM = my_bam,
                    output_file_name=
                        "test_header_from_R.txt"
                    ))
tsk$run()
```

sevenbridges-r R package

live demo

DEMO