go2gdc: a Go package for flexible filtering, downloading and integrating multi-omics data from Genomic Data Commons

Lin Wei

2020-04-17

Contents

List of Tables

List of Figures

Preface

This short book introduces an open-source Go package, go2gdc, as well as its executable binary files on various operating systems, for flexible filtering, downloading and integrating open-access data from NCI's Genomic Data Commons (GDC).

Abstract

The NCI's Genomic Data Commons (GDC), a data-driven platform hosting cancer genome programs (including TCGA etc.), provides most data files at sample/aliquot level (1 file per sample). Although some public databases and R packages can offer some integrated GDC data, it is still difficult for most researcher without programming skill to automatically find cases which have both of tumor and normal samples in all of different omics types. Here the go2gdc, an open-source Go package, is developed for flexible filtering, downloading and integrating multi-omics open-access data from GDC. Its executable binary files on various operating systems make the wide usage possible. And its stepwise module design increases its flexibility significantly.

Chapter 1

Introduction

1.1 motivation

The NCI's Genomic Data Commons (GDC)[1] provides the cancer research community a data-driven platform with a unified data repository of big quantity and high quality data from several cancer genome programs at the NCI Center for Cancer Genomics (CCG), including The Cancer Genome Atlas (TCGA) and Therapeutically Applicable Research to Generate Effective Treatments (TARGET), by which the GDC enables data sharing across cancer genomic studies in support of precision medicine.

The GDC Data Portal is a interface for cancer researchers and bioinformaticians to search and download cancer data for analysis. It archives high-quality datasets of Mutations, Copy Number Variants, Expression Quantification and Post-transcriptional Modifications. Worldwide cancer researchers have involved GDC data in their studies.

Being a data-driven platform, GDC provides many characteristics filters for users to find data files of interested cases and/or samples. Most of these data files are at sample/aliquot level, which means the data in a single file comes from same sample. These make the flexible filter possible, but bring an extra step of data integration to users if they need a batch of samples for analysis.

If "integrated" is necessary, users can try the Broad GDAC Firehose, which provides integrated *original* TCGA data of all samples from same project (cancer type). Here the *original* means their TCGA data have not been harmonized by GDC, like the data from GDC Legacy Archive, where the original TCGA data archived. If both "integrated" and "harmonized" are preferable, the GDC Xena Hub of UCSC[2] is an option, from where users can find integrated tables of all samples from specified cancer type

and/or sample type.

If not all samples in the integrated table is interested, users would have to process the integrated table manually, or use third-party R[[3]] packages: TCGA-Assembler 2[4][5] or TCGAbiolinks[6][7]. With either, users with R skills can download and process GDC data in R environment.

When processing big-size data, the reliable internet connection is necessary for long-time downloading, and big RAM is essential for big-size integrating. For data security, the servers with big RAM are always protected by firewall, which restricts their internet connection. For example, some bioinformaticians work on a personal computer (PC) with reliable internet connection but small RAM, and submit their batch work to a server with big RAM but restricted network connection (by firewall). That means a new tool which can downloading on PC and integrating on server respectively will bring much convenience. And in many cases, the operating system (OS) of PC is Windows or Mac, while the OS of server is Linux. That is to say the tool should be an OS-independent tool.

On the other hand, with the growing demand of deep learning in cancer research, more researchers without R skills but with Python and TensorFlow skills will try to utilize the GDC data. In these cases, an R-free tool for flexible data preparing will significantly facilitate the studies of cancer research community.

Here I introduce an open-source Go package, go2gdc, as well as its executable binary files on various OS, for flexible filtering, downloading and integrating open-access data from GDC Data Portal API.

1.2 why go2gdc

Comparing to the published GDC API communication software:

1. TCGA-Assembler 2[4][5] and
2. TCGAbiolinks[6][7],

the go2gdc is:

1. **easy**:
    1. The executable binary files work in Windows, MacOS, FreeBSD and Linux.
    2. No programming skill (R, Python or Go etc) is necessary.
2. **flexible**:
    1. All users can start from

    1. a filter-file (with keywords describing target data),
    2. a json format answer-file of filter query, or
    3. a pair of answer-file with downloaded-file.
  2. Advanced users can integrate
    1. the go2gdc binary file into their shell script, or
    2. the go2gdc source code into their Go code.
  3. The downloading and integrating can be performed on same computer or on different computers respectively
    1. downloading on a personal laptop with high speed internet connection, then
    2. integrating on a server with big RAM but restricted network connection (by firewall).

Comparing to Firehose and GDC Legacy Archive, the TCGA data got by go2gdc is:

1. **up-to-date** (by GDC):
  1. re-harmonized TCGA data, and
  2. latest version of cases information.
2. **uniform** (by go2gdc):
  1. a table with only one header row, in which
  2. one gene (probe) per row with leading gene_id (probe_id), and
  3. one sample per column with leading "data_source_id" (Id).
3. **traceable** (by go2gdc):
  1. the "data_source_id" (Id) consisting of
    1. "project ID",
    2. "sample ID" and
    3. "UUID of source file".

Another, comparing to GDC Xena Hub of UCSC, the go2gdc can easily find special cases which:

1. have data from different sample types, such as
  1. all cases with Primary Solid Tumor (01) samples data
  2. all cases with Solid Tissue Normal (11) samples data
  3. all cases with data from both "01" and "11" samples
2. have data from different oimcs types, such as
  1. all cases with both microRNA and gene expression data from "01" samples
  2. all cases with both microRNA and gene expression data from "11" samples
  3. all cases with both microRNA and gene expression data from "01" and "11" samples

Additionally, the go2gdc provides the involved cases information **simultaneously** and

exactly.

1.3 get started

Before using go2gdc, no installation is needed. It is provided in two form:

1. binary file and
2. source code

The easiest way to get started with go2gdc is using the executable go2gdc binary file.

1. find the appropriate binary file for the operating system
2. copy the binary file into the working directory
3. run it in command-line interface:
    1. ".\go2gdc" in Command Prompt (Windows)
    2. "./go2gdc" in Terminal (MacOS, FreeBSD or Linux)
4. to run "go2gdc" anywhere, add the directory of go2gdc to the environment variable PATH

For Shell users, the go2gdc binary file can be integrated into their shell script.

For Go programmers, the go2gdc source code can be customized and integrated into their own Go code.

The usage will be detailed in Chapter 2. Please note, the commands used in examples are in MacOS or Linux format.

Chapter 2

Usage

```
go2gdc from=${fromStatus}:${fromPath} to=${toStatus}:${toPath}
```

- **`${fromStatus}`** or **`${toStatus}`**: one of four *ordered* statuses
  1. filter
  2. answer
  3. downloaded
  4. integrated
- **`${fromPath}`** or **`${toPath}`**: can be
  - relative path:
    - ".\x", "..\y\x" (Windows) or
    - "x", "./x", "../y/x" (others)
  - absolute path:
    - "C:\z\y\x" (Windows) or
    - "/z/y/x" (others)
- **`from=`**: the **start** point
  - `${fromStatus}`: the status to start from, can be one of
    - filter
    - answer
    - downloaded
  - `${fromPath}`: the path of file to **read from**
- **`to=`**: the **end** point:
  - `${toStatus}`: the status to end at, can be one of
    - answer
    - downloaded
    - integrated
  - `${toPath}`: the path **prefix** of file to **write to**

To user, the simplest procedure is starting from a few keywords to make a filter-file (e.g. filter.txt) like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = false
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

Using filter-file template, user can customize a filter of data files to download and integrate. Then, given filter-file, the go2gdc will :

1. query GDC with parsed **filter**(s);
2. generate the group(s) of **answer** with cases information files;
3. **download** the filtered data files;
4. **integrate** the data in downloaded-file.

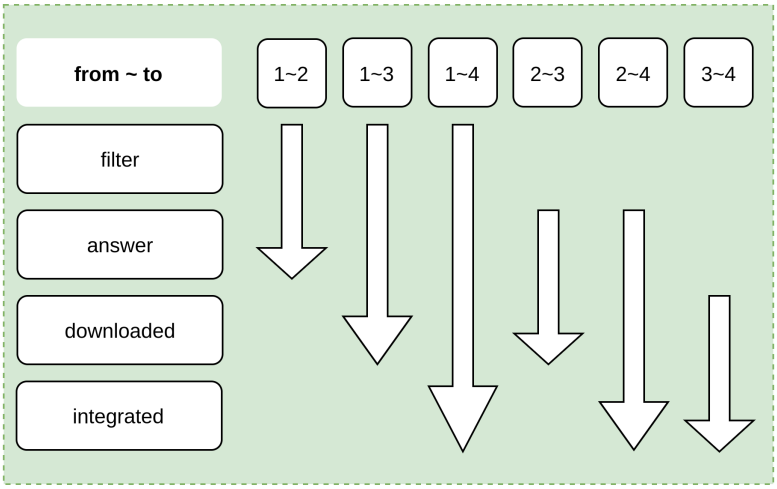

Figure 2.1: Flexible flows of go2gdc

The Figure 2.1 shows different flows with alternative pairs of from~to statuses.

2.1 1~2: filter ~ answer

```
# For Windows:
go2gdc from=filter:.\path\from\filter.txt to=answer:.\path\to\answer--

# For MacOS, FreeBSD or Linux:
go2gdc from=filter:./path/from/filter.txt to=answer:./path/to/answer--
```

1. read **filter**-file
    1. make one filter for each omic type
    2. query GDC with every parsed filter
2. save the json format **answer** of each filter
    - with path prefix "**`./path/to/answer--`**"
    - with file extension ".json"
3. save the information of involved cases
    - with path prefix "**`./path/to/answer--`**"
    - with filename suffix "–Case.json" for detailed information
    - with filename suffix "–Case.tsv" for integrated table

2.2 1~3: filter ~ downloaded

```
# For Windows:
go2gdc from=filter:.\path\from\filter.txt to=downloaded:.\path\to\downloaded--

# For MacOS, FreeBSD or Linux:
go2gdc from=filter:./path/from/filter.txt to=downloaded:./path/to/downloaded--
```

1. read **filter**-file
    1. make one filter for each omic type
    2. query GDC with every parsed filter
2. save the json format **answer** of each filter
    - with path prefix "**`./path/to/downloaded--`**"
    - with file extension ".json"
3. save the information of involved cases
    - with path prefix "**`./path/to/downloaded--`**"
    - with filename suffix "–Case.json" for detailed information
    - with filename suffix "–Case.tsv" for integrated table
4. **download** the filtered data files
    - save the downloaded-file with path prefix "**`./path/to/downloaded--`**"

2.3 1~4: filter ~ integrated

```
# For Windows:
go2gdc from=filter:.\path\from\filter.txt to=integrated:.\path\to\integrated--

# For MacOS, FreeBSD or Linux:
go2gdc from=filter:./path/from/filter.txt to=integrated:./path/to/integrated--
```

1. read **filter**-file
    1. make one filter for each omic type
    2. query GDC with every parsed filter
2. save the json format **answer** of each filter
    - with path prefix "**`./path/to/integrated--`**"
    - with file extension ".json"
3. save the information of involved cases
    - with path prefix "**`./path/to/integrated--`**"
    - with filename suffix "–Case.json" for detailed information
    - with filename suffix "–Case.tsv" for integrated table
4. **download** the filtered data files
    - save the downloaded-file with path prefix "**`./path/to/integrated--`**"
5. **integrate** the data in downloaded-file
    - save the integrated-file(s) with path prefix "**`./path/to/integrated--`**"

2.4 2~3: answer ~ downloaded

```
# For Windows:
go2gdc from=answer:.\path\from\answer.json to=downloaded:.\path\to\downloaded--

# For MacOS, FreeBSD or Linux:
go2gdc from=answer:./path/from/answer.json to=downloaded:./path/to/downloaded--
```

1. read the json format **answer**-file
2. save the information of involved cases
    - with path prefix "**`./path/from/answer`**"
    - with filename suffix "–Case.json" for detailed information
    - with filename suffix "–Case.tsv" for integrated table
3. **download** the filtered data files
    - save the downloaded-file with path prefix "**`./path/to/downloaded--`**"

2.5 2~4: answer ~ integrated

```
# For Windows:
go2gdc from=answer:.\path\from\answer.json to=integrated:.\path\to\integrated--

# For MacOS, FreeBSD or Linux:
go2gdc from=answer:./path/from/answer.json to=integrated:./path/to/integrated--
```

1. read the json format **answer**-file
2. save the information of involved cases
    - with path prefix "**`./path/from/answer`**"
    - with filename suffix "–Case.json" for detailed information
    - with filename suffix "–Case.tsv" for integrated table
3. **download** the filtered data files
    - save the downloaded-file with path prefix "**`./path/to/integrated--`**"
4. **integrate** the data in downloaded-file
    - save the integrated-file(s) with path prefix "**`./path/to/integrated--`**"

2.6 3~4: downloaded ~ integrated

```
# For Windows:
go2gdc from=downloaded:.\path\from\downloaded.tar.gz to=integrated:.\path\to\integrated--

# For MacOS, FreeBSD or Linux:
go2gdc from=downloaded:./path/from/downloaded.tar.gz to=integrated:./path/to/integrated--
```

1. read the pair of **answer**-file and **downloaded**-file
2. **integrate** the data in downloaded-file
    - save the integrated-file(s) with path prefix "**`./path/to/integrated--`**"

Note:

Since the step "downloaded ~ integrated" may be performed on a server without internet connection, there is no cases information files generated here.

Chapter 3

1~2: filter ~ answer

The filter can be defined via a text file (e.g. filter.txt) by **fields** with *comma-separated values* :

1. **`omics_type`** (necessary), can be one or some of
    - somatic_mutation,
    - cnv_gene,
    - cnv_segment_somatic_only,
    - cnv_segment_somatic_and_germline,
    - gene_htseq_fpkm_uq,
    - gene_htseq_fpkm,
    - gene_htseq_counts,
    - gene_star_counts,
    - methy_27,
    - methy_450,
    - mir,
    - mir_isoform.
2. **`project_id`** (necessary), can be one or some IDs of GDC project.
3. **`sample_type_id`** (optional), can be
    - blank (all available sample_type_id) or
    - list of sample_type_id.
4. **`sample_types_for_separated_integration`** (optional), can be one of
    - true (t),
    - false (f),
    - blank (= false).
5. **`sample_types_for_cases_intersection`** (optional), can be one of
    - true (t),
    - false (f),

    - blank (= false).
6. **`omics_types_for_cases_intersection`** (optional), can be one of
    - true (t),
    - false (f),
    - blank (= false).
7. **`keep_samples_from_same_case`** (optional), can be one of
    - none
    - one
    - all
    - blank (= all).
8. **`case_id`** (optional), can be
    - blank (for all available cases) or
    - list of case IDs.

Given filter-file, the go2gdc will:

first (Figure 3.1),

1. parse the filter-file into the (*1st*-)filter
2. read field `omics_type`,
    - for **each** `omics_type` value, make a `omics_type` specific (*2nd*-)filter
3. for **each** `omics_type` value,
    a. assemble the (*3rd*-)filter with value(s) of fields:
        - `omics_type` (only one value)
        - `project_id` (one or multiple values)
        - `sample_type_id` (if any)
        - `case_id` (if any)
    b. send the (*3rd*-)filter to GDC, parse the response, then
    c. for **each** `sample_type_id` value, make the (*4th*-)filter with `keep_samples_from_same_case` value:
        - if `all` (or blank), do nothing
        - if `none`, eliminate all samples from same case
        - if `one`, randomly choose one sample to represent the case
    d. make the (*5th*-)filter with `sample_types_for_cases_intersection` value:
        - if `true` (or `t`), select the cases which have **all** types of samples requested in field `sample_type_id`
        - if `false` (or `f`, blank), do nothing
    e. make the (*6th*-)filter with `sample_types_for_separated_integration` value:
        - if `true` (or `t`), generate a group of filters for answer-files, 1 filter per `sample_type_id` value
        - if `false` (or `f`, blank), generate only one filter, including all types of

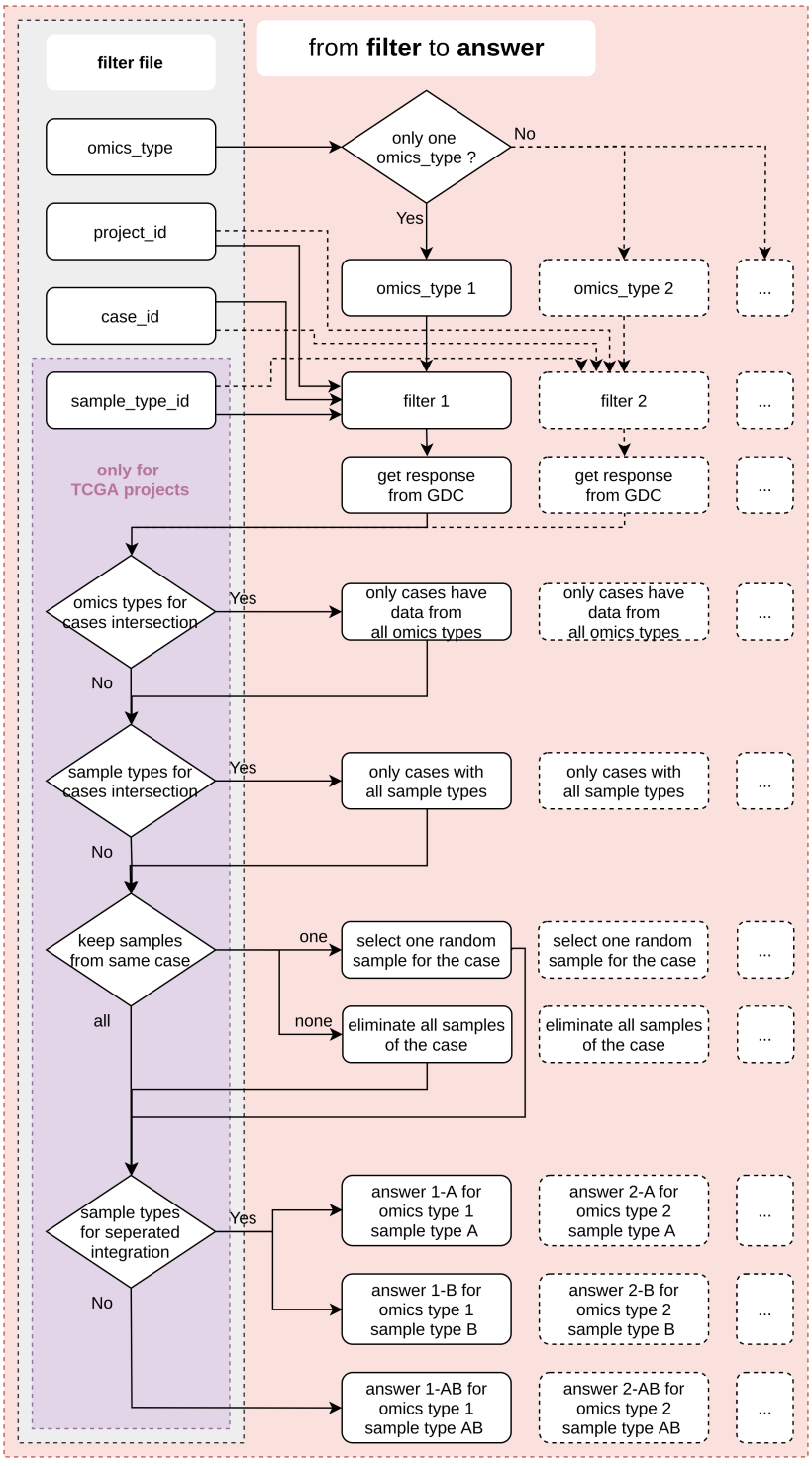

Figure 3.1: From filter to answer

samples requested in field `sample_type_id`

4. make the (*7th-*)filter with `omics_types_for_cases_intersection` value:
   - if `true` (or `t`), for each `sample_type_id` value, select the cases which appear in all `omics_type` values
   - if `false` (or `f`, blank), do nothing
5. get **one** answer-file for **each** (*7th-*)filter

second (Figure 3.2), with **each** answer-file,

1. get the `case_id`(s) from answer-file;
2. make and save **a pair** of involved cases information files with `case_id`(s).
   - `case.json`: cases information in json format
   - `case.tsv`: extracted cases information table

Note:

1. For **every** filter-file, the go2gdc will generate **one or more** answer-files.
2. For **every** answer-file, the go2gdc will generate **a pair of** cases information files.

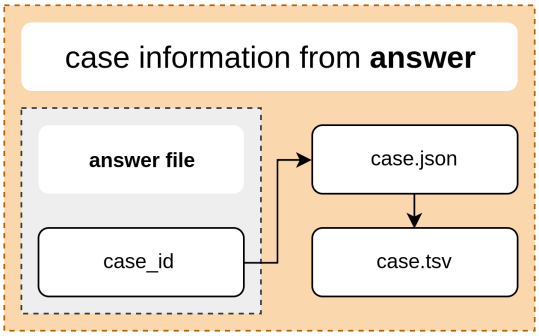

Figure 3.2: Case information from answer

To help readers' understanding, here are some examples of **filter-file**, which can be generated with example.sh:

Filters for transcriptome data

1. mir
2. gene_htseq_fpkm_uq

from lung cancer projects

1. TCGA-LUAD
2. TCGA-LUSC

in different scenarios:

3.1 all available cases

To get data:

1. of all available cases

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id =
sample_types_for_separated_integration =
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id =
```

in which:

- `omics_type` = **mir, gene_htseq_fpkm_uq**
- `project_id` = **TCGA-LUAD, TCGA-LUSC**

will generate **2 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `sample_type_id` = all available sample types
3. `case_id` = all available cases

and *differences*:

1. `omics_type` = gene_htseq_fpkm_uq: `answer.json` + `case.josn` + `case.tsv`
2. `omics_type` = `mir`: `answer.json` + `case.josn` + `case.tsv`

3.2 separated integration in different sample types respectively

To get data:

1. of all available cases
2. with separated integration in different sample types respectively

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id =
sample_types_for_separated_integration = true
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id =
```

in which:

- `sample_types_for_separated_integration` = **true**

will generate **6 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = all available cases

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 02: `answer.json` + `case.json` + `case.tsv`
    3. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 02: `answer.json` + `case.json` + `case.tsv`
    3. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`

Note:

1. `sample_type_id` = 01: Primary Solid Tumor (in TCGA-LUAD and TCGA-LUSC)
2. `sample_type_id` = 02: Recurrent Solid Tumor (only in TCGA-LUAD)
3. `sample_type_id` = 11: Solid Tissue Normal (in TCGA-LUAD and TCGA-LUSC)

3.3 only cases which have any specified sample types

To get data:

1. of only cases which have sample type ***either*** "01" ***or*** "11"
    1. **01**: Primary Solid Tumor
    2. **11**: Solid Tissue Normal
2. with separated integration in different sample types respectively

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id =
```

in which:

- `sample_type_id` = **01, 11**

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = all available cases

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`

3.4 only cases which have all specified sample types

To get data:

1. of only cases which have sample types ***both*** "01" ***and*** "11"
    1. **01**: Primary Solid Tumor
    2. **11**: Solid Tissue Normal
2. with separated integration in different sample types respectively

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id =
```

in which:

- `sample_types_for_cases_intersection` = **true**

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = have both "01" and "11" sample types

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`

3.5 only cases have all specified sample types in all omics types

To get data:

1. of only cases which:
    1. have sample types ***both*** "01" ***and*** "11"
        1. **01**: Primary Solid Tumor
        2. **11**: Solid Tissue Normal
    2. in **all** omics types
        1. mir
        2. gene_htseq_fpkm_uq
2. with separated integration in different sample types respectively

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case =
case_id =
```

in which:

- `omics_types_for_cases_intersection` = **true**

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = have both "01" and "11" sample types in all omics types

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv`
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv`

3.6 using case IDs

Extracting the case ids from answer-files of the 3.5.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id = TCGA-22-4609, TCGA-22-5472, TCGA-22-5478, TCGA-22-5483,
          TCGA-33-4587, TCGA-33-6737, TCGA-34-7107, TCGA-34-8454,
          TCGA-39-5040, TCGA-43-5670, TCGA-43-6143, TCGA-43-6771,
          TCGA-43-6773, TCGA-43-7657, TCGA-43-7658, TCGA-44-2655,
          TCGA-44-2657, TCGA-44-2661, TCGA-44-2665, TCGA-44-2668,
          TCGA-44-3396, TCGA-44-3398, TCGA-44-6776, TCGA-44-6777,
          TCGA-44-6778, TCGA-49-6742, TCGA-49-6743, TCGA-49-6744,
          TCGA-49-6745, TCGA-50-5930, TCGA-50-5932, TCGA-50-5933,
          TCGA-56-7222, TCGA-56-7579, TCGA-56-7580, TCGA-56-7582,
          TCGA-56-7730, TCGA-56-7731, TCGA-56-7823, TCGA-56-8082,
          TCGA-56-8083, TCGA-56-8201, TCGA-56-8309, TCGA-56-8623,
          TCGA-58-8386, TCGA-77-7138, TCGA-77-7142, TCGA-77-7337,
          TCGA-77-7338, TCGA-77-8007, TCGA-77-8008, TCGA-85-7710,
          TCGA-90-6837, TCGA-90-7767, TCGA-91-6835, TCGA-91-6836,
          TCGA-92-7340
```

in which:

- `case_id` = TCGA-22-4609, ..., TCGA-44-2668, ..., TCGA-92-7340

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`) (same as the 3.5), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = in the scope of given case ids

and *differences*:

1. `omics_type` = gene_htseq_fpkm_uq
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (59 files from 57 cases)

        1. **three** files from case **TCGA-44-2668**
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case

Note:

1. Everyone of these cases has both "01" and "11" sample types in all omics types.
2. No other (TCGA-LUAD, TCGA-LUSC) case has both "01" and "11" sample types in all omics types.
3. So these 4 answer-files are same as the 3.5.

And,

1. Usually, there is only **one file per case** in each `omics_type` and each `sample_type_id`.
2. The case **TCGA-44-2668** is special.

3.7 only case TCGA-44-2668

To get data:

1. of case **TCGA-44-2668**
2. with separated integration in different sample types respectively

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id =
sample_types_for_separated_integration = true
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id = TCGA-44-2668
```

in which:

- `case_id` = TCGA-44-2668

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = TCGA-44-2668

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (3 files from 1 case)
        1. `barcode`: TCGA-44-2668-01A-01R-A278-07
        2. `barcode`: TCGA-44-2668-01A-01R-0946-07
        3. `barcode`: TCGA-44-2668-01B-02R-A277-07
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (1 file from 1 case)
        1. `barcode`: TCGA-44-2668-11A-01R-1758-07
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (1 file from 1 case)
        1. `barcode`: TCGA-44-2668-01A-01T-0947-13
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (1 file from 1 case)
        1. `barcode`: TCGA-44-2668-11A-01R-1757-13

Note:

1. Usually, there is only **one file per case** in each `omics_type` and each `sample_type_id`.
2. The case TCGA-44-2668 is a special case in project TCGA-LUAD.
3. If "**one file per case**" is necessary, user should
    1. choose one file from three, or
    2. eliminate the case TCGA-44-2668, or
    3. other procedure (such as `max`, `min`, `mean`, etc).
4. In GDC Xena Hub of UCSC (LUAD, HTSeq - FPKM-UQ), there are 2 columns for case TCGA-44-22668:
    1. column "TCGA-44-2668-01B" from TCGA-44-2668-01B-02R-A277-07
    2. column "TCGA-44-2668-01A" from average of:
        1. TCGA-44-2668-01A-01R-A278-07 and
        2. TCGA-44-2668-01A-01R-0946-07.

3.8 eliminating TCGA-44-2668 manually

Similar to the 3.6, only **except** case TCGA-44-2668.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id = TCGA-22-4609, TCGA-22-5472, TCGA-22-5478, TCGA-22-5483,
          TCGA-33-4587, TCGA-33-6737, TCGA-34-7107, TCGA-34-8454,
          TCGA-39-5040, TCGA-43-5670, TCGA-43-6143, TCGA-43-6771,
          TCGA-43-6773, TCGA-43-7657, TCGA-43-7658, TCGA-44-2655,
          TCGA-44-2657, TCGA-44-2661, TCGA-44-2665,
          TCGA-44-3396, TCGA-44-3398, TCGA-44-6776, TCGA-44-6777,
          TCGA-44-6778, TCGA-49-6742, TCGA-49-6743, TCGA-49-6744,
          TCGA-49-6745, TCGA-50-5930, TCGA-50-5932, TCGA-50-5933,
          TCGA-56-7222, TCGA-56-7579, TCGA-56-7580, TCGA-56-7582,
          TCGA-56-7730, TCGA-56-7731, TCGA-56-7823, TCGA-56-8082,
          TCGA-56-8083, TCGA-56-8201, TCGA-56-8309, TCGA-56-8623,
          TCGA-58-8386, TCGA-77-7138, TCGA-77-7142, TCGA-77-7337,
          TCGA-77-7338, TCGA-77-8007, TCGA-77-8008, TCGA-85-7710,
          TCGA-90-6837, TCGA-90-7767, TCGA-91-6835, TCGA-91-6836,
          TCGA-92-7340
```

in which:

- `case_id` = TCGA-22-4609, ... (except TCGA-44-2668), ..., TCGA-92-7340

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = in the scope of given case ids

and *differences*:

1. `omics_type` = gene_htseq_fpkm_uq
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)

    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)

Note:

1. In these answer-files, there is only **one file per case** in each `omics_type` and each `sample_type_id`.
2. From these 4 answer-files, the 4 integrated-files will have same **case order** in their column headers.

3.9 eliminating TCGA-44-2668 automatically

Similar to the 3.8, but eliminating case TCGA-44-2668 automatically.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

in which:

- `keep_samples_from_same_case` = **none**

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`) (same as the 3.8), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = in the scope of given case ids

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (56 files from 56 cases)

Note:

1. All these 4 answer-files are same as the 3.8.
2. In these answer-files, there is only **one file per case** in each `omics_type` and each `sample_type_id`.
3. From these 4 answer-files, the 4 integrated-files will have same **case order** in their column headers.

3.10 alternative way to keep all samples of TCGA-44-2668

Similar to the 3.5, keeping **all** samples from TCGA-44-2668.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = all
case_id =
```

in which:

- `keep_samples_from_same_case` = **all** (blank means "all")

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`) (same as the 3.5), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = have both "01" and "11" sample types in all omics types

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (59 files from 57 cases)
        1. **three** files from case **TCGA-44-2668**
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668

2. one file from each other case

Note:

1. All these 4 answer-files are same as the 3.5.

3.11 keep TCGA-44-2668 with one random sample

Similar to the 3.10, but keep TCGA-44-2668 case with **one random sample**.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = one
case_id =
```

in which:

- `keep_samples_from_same_case` = **one**

will generate **4 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = have both "01" and "11" sample types in all omics types

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. **one random file** from case **TCGA-44-2668**
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
2. `omics_type` = `mir`
    1. `sample_type_id` = 01: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case
    2. `sample_type_id` = 11: `answer.json` + `case.json` + `case.tsv` (57 files from 57 cases)
        1. one file from case TCGA-44-2668
        2. one file from each other case

3.12 like 3.9, but not separated integration in different sample types

Similar to the 3.9, but no separated integration in different sample types.

Using filter-file like this:

```
omics_type = mir, gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = false
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

in which:

- `sample_types_for_separated_integration` = **false**

will generate **2 groups** of 3 files (`answer.json` + `case.json` + `case.tsv`), with *similarities*:

1. `project_id` = TCGA-LUAD and TCGA-LUSC
2. `case_id` = have both "01" and "11" sample types in all omics types

and *differences*:

1. `omics_type` = `gene_htseq_fpkm_uq`: `answer.json` + `case.json` + `case.tsv` (112 files from 56 cases)
    1. `sample_type_id` = 01: 56 files
    2. `sample_type_id` = 11: 56 files
2. `omics_type` = `mir`: `answer.json` + `case.json` + `case.tsv` (112 files from 56 cases)
    1. `sample_type_id` = 01: 56 files
    2. `sample_type_id` = 11: 56 files

Chapter 4

2~3: answer ~ downloaded

Every answer-file is a json format response from GDC server, which has useful information:

1. `case_id`: for cases information files: `case.json` and `case.tsv`
2. `file_id`: for target files to download
3. `omics_type`: for integration strategy
4. `omics_type + sample_type_id`: for the file name of downloaded-file and integrated-file(s)
5. `project_id` + `barcode` + `file_id`: for traceable data_source_id

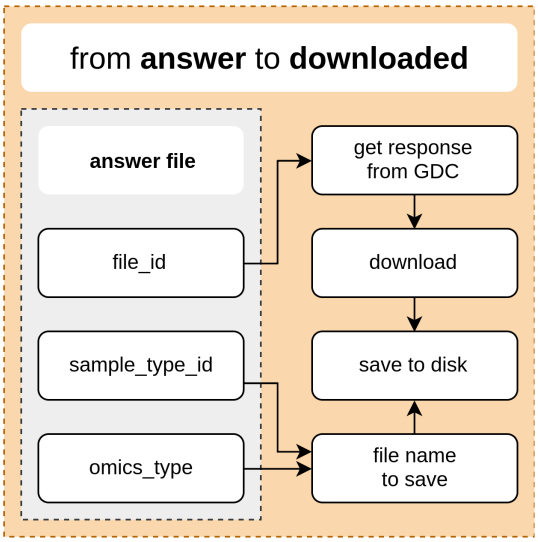

Figure 4.1: From answer to downloaded

In this step (Figure 4.1), with **one** answer-file, the go2gdc will

1. get the `case_id`(s) from answer-file
2. make and save **a pair** of involved cases information files (Figure 3.2)
    - `case.json`: cases information in json format
    - `case.tsv`: extracted cases information table
3. get the `file_id`(s) from answer-file
4. download target data files with `file_id`(s) from GDC
5. save the downloaded-file with given path prefix in 2 alternative format:
    - if **only one** target file to download
        - an uncompressed file
    - if **more than one** target files to download
        - a `*.tar.gz` file including a "MANIFEST.txt" file of all data files

4.1 TCGA-44-2668 cancer samples

Like 3.7, but

1. `omics_type` = `gene_htseq_fpkm_uq`
2. `sample_type_id` = `01`.

In this answer-file (`answer.json`) includes 3 target files:

1. TCGA-44-2668-01A-01R-A278-07
2. TCGA-44-2668-01A-01R-0946-07
3. TCGA-44-2668-01B-02R-A277-07

Use command like this (in Linux):

```
# change directory to "go2gdc/doc"
# set environment
s0Project=Project_TCGA-LUAD
s0Omicstype=Omitstype_gene_htseq_fpkm_uq
s0Sampletype=Sampletype_01
s0Filename=${s0Project}--${s0Omicstype}--${s0Sampletype}.json
# make a copy of answer-file
cp ../example/status1to2-eg7--${s0Filename} \
   ../example/status2to3-eg1--${s0Filename}
# run go2gdc
../bin/windows_amd64/go2gdc \
  from=answer:../example/status2to3-eg1--${s0Filename} \
  to=downloaded:../example/status2to3-eg1--
```

will first make a copy of answer-file

- status2to3-eg1–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_01.json

then starts from this answer-file,

1. get the `case_id`(s), make and save **a pair** of involved cases information files (`case.json` + `case.tsv`)
   - status2to3-eg1–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_01–Case.json
   - status2to3-eg1–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_01–Case.tsv
2. download target data files with `file_id`(s) and save the zip file as `downloaded.tar.gz`:
   - status2to3-eg1–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_01.tar.gz

4.2 TCGA-44-2668 normal sample

Like 3.7, but

1. `omics_type` = `gene_htseq_fpkm_uq`
2. `sample_type_id` = 11.

In this answer-file `answer.json` includes only 1 target file:

1. TCGA-44-2668-11A-01R-1758-07

Use command like this (in Linux):

```
# change directory to "go2gdc/doc"
# set environment
s0Project=Project_TCGA-LUAD
s0Omicstype=Omitstype_gene_htseq_fpkm_uq
s0Sampletype=Sampletype_11
s0Filename=${s0Project}--${s0Omicstype}--${s0Sampletype}.json
# make a copy of answer-file
cp ../example/status1to2-eg7--${s0Filename} \
   ../example/status2to3-eg2--${s0Filename}
# run go2gdc
../bin/windows_amd64/go2gdc \
  from=answer:../example/status2to3-eg2--${s0Filename} \
  to=downloaded:../example/status2to3-eg2--
```

will first make a copy of answer-file

- status2to3-eg2–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_11.json

then starts from this answer-file,

1. get the `case_id`(s), make and save **a pair** of involved cases information files (`case.json` + `case.tsv`)
   - status2to3-eg2–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_11–Case.json
   - status2to3-eg2–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_11–Case.tsv
2. download target data file with `file_id`(s) and save it in `downloaded-directory`:
   - status2to3-eg2–Project_TCGA-LUAD–Omicstype_gene_htseq_fpkm_uq–Sampletype_11/

Chapter 5

3~4: downloaded ~ integrated

With **one pair** of answer-file and downloaded-file (Figure 5.1), the go2gdc will

1. recognize the `omics_type` from answer-file;
2. determine the integration strategy with `omics_type`;
3. untar (if necessary) and read the data in downloaded-file;
4. integrate the data file(s) with `omics_type` specified integration strategy;
5. save the integrated-file(s) with given path prefix.

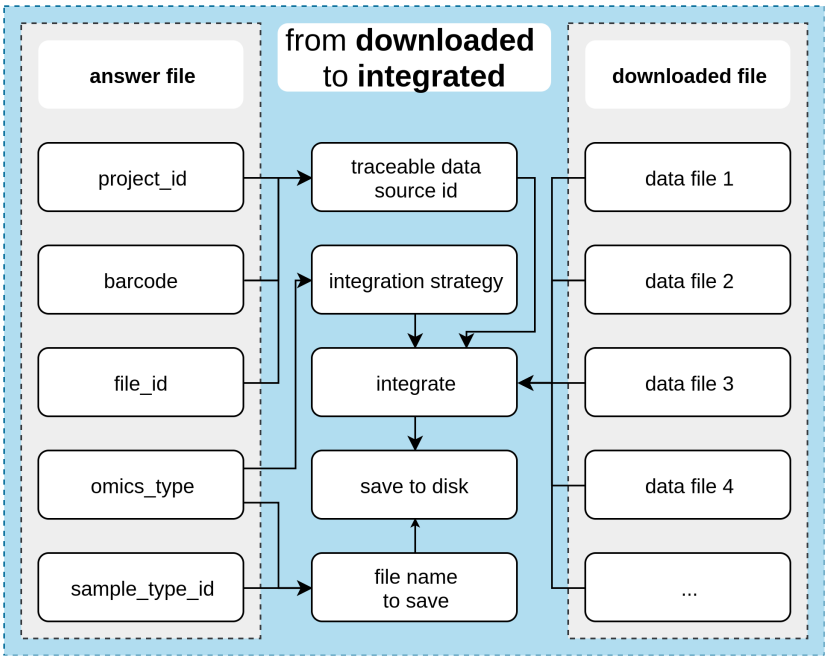

Figure 5.1: From downloaded to integrated

Currently, the go2gdc supports 12 omics types (Table 5.1). Since every omics type has it own data format, go2gdc integrates omics type specified data files with different integration strategy, i.e., the integration strategy is determined by `omics_type`.

From GDC, the data files of most omics types are at aliquot level which means the data in a single file comes from same sample. Except for the "somatic_mutation" and "cnv_gene", data files from which are at project level, i.e., the data in a single file comes from same project but multiple cases. These can be found in "file count" column of Table 5.1.

The case filtering function of go2gdc is designed to be performed before downloading. So the data selection with "sample type" and "case id" in filter-file will not work for these two omics types. The "case filter" column in Table 5.1 shows the differences.

Till now, the TCGA projects is the majority part of the GDC data. As marked in "TCGA only" column of Table 5.1, some omics types only have data from TCGA projects.

Table 5.1: Information of omics types

No.	omics_type	file count	cases filter	TCGA only
1	somatic_mutation	4 files/project	No	Yes
2	cnv_gene	1 file/project	No	Yes
3	cnv_segment_somatic_only	1 file/aliquot	Yes	Yes
4	cnv_segment_somatic_and_germline	1 file/aliquot	Yes	Yes
5	gene_htseq_fpkm_uq	1 file/aliquot	Yes	No
6	gene_htseq_fpkm	1 file/aliquot	Yes	No
7	gene_htseq_counts	1 file/aliquot	Yes	No
8	gene_star_counts	1 file/aliquot	Yes	No
9	methy_27	1 file/aliquot	Yes	Yes
10	methy_450	1 file/aliquot	Yes	Yes
11	mir	1 file/aliquot	Yes	No
12	mir_isoform	1 file/aliquot	Yes	No

Note:

1. This step is designed to avoid internet connection, so
   - it is possible to run on a server protected by firewall;
   - no cases information generation here, which needs internet connection.
2. In the following selections, the header "Id-?" is "data_source_id":
   - "project ID" + "sample ID" + "UUID of source file"
3. The filter similar to 3.9 will be used for demo in different omics types.

5.1 somatic_mutation

From GDC, the data files of "somatic_mutation" are at project level, i.e., the data in a single file comes from same project but multiple cases.

For every TCGA project, GDC provide 4 "somatic_mutation" files from 4 pipelines respectively:

1. muse
2. mutect
3. somaticsniper
4. varscan

When `omics_type = somatic_mutation`, the sample types and cases selection fields in filter-file will not work:

1. `sample_type_id =`
2. `sample_types_for_separated_integration =`
3. `sample_types_for_cases_intersection =`
4. `omics_types_for_cases_intersection =`
5. `keep_samples_from_same_case =`
6. `case_id =`

which means these 6 fields will be omitted. With this filter-file:

```
omics_type = somatic_mutation
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from 4 pipelines of every project:

Table 5.2: somatic mutation: GDC file (any project, any pipeline)

Hugo_Symbol	Entrez_Gene_Id	...
symol1	entrez1	...
symol2	entrez2	...
...	...	...

will be integrated by go2gdc into 1 project-level table (*.tsv) after:

1. adding 3 heading columns
    1. “file_id” (UUID of source file)
    2. “project_id”
    3. “calling_pipeline”

Table 5.3: somatic mutation: integrated table (2 projects, 4 pipelines)

file_id	project_id	calling_pipeline	Hugo_Symbol	Entrez_Gene_Id	...
UUID-1	TCGA-LUAD	muse	symol1	entrez1	...
UUID-1	TCGA-LUAD	muse	symol2	entrez2	...
UUID-2	TCGA-LUAD	mutect	symol1	entrez1	...
UUID-2	TCGA-LUAD	mutect	symol2	entrez2	...
UUID-3	TCGA-LUAD	somaticsniper	symol1	entrez1	...
UUID-3	TCGA-LUAD	somaticsniper	symol1	entrez1	...
UUID-4	TCGA-LUAD	varscan	symol2	entrez2	...
UUID-4	TCGA-LUAD	varscan	symol2	entrez2	...
UUID-5	TCGA-LUSC	muse	symol1	entrez1	...
UUID-5	TCGA-LUSC	muse	symol2	entrez2	...
UUID-6	TCGA-LUSC	mutect	symol1	entrez1	...
UUID-6	TCGA-LUSC	mutect	symol2	entrez2	...
UUID-7	TCGA-LUSC	somaticsniper	symol1	entrez1	...
UUID-7	TCGA-LUSC	somaticsniper	symol1	entrez1	...
UUID-8	TCGA-LUSC	varscan	symol2	entrez2	...
UUID-8	TCGA-LUSC	varscan	symol2	entrez2	...
...	...	...	...	...	...

Note:

1. only 4 “somatic_mutation” files per TCGA project, which includes somatic mutation data from multiple cases (aliquots).

5.2 cnv_gene

From GDC, the data files of "cnv_gene" are at project level, i.e., the data in a single file comes from same project but multiple cases.

For every TCGA project, GDC provide 1 "cnv_gene" file.

When `omics_type = somatic_mutation`, the sample types and cases selection fields in filter-file will not work:

1. `sample_type_id =`
2. `sample_types_for_separated_integration =`
3. `sample_types_for_cases_intersection =`
4. `omics_types_for_cases_intersection =`
5. `keep_samples_from_same_case =`
6. `case_id =`

which means these 6 fields will be omitted. With this filter-file:

```
omics_type = cnv_gene
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different projects:

Table 5.4: cnv gene: GDC file (TCGA-LUAD with aliquot A, B)

Gene Symbol	Gene ID	Cytoband	AliquotId-A	AliquotId-B	...
ENSG00000008128.21	0	1p36.33	a1	b1	...
ENSG00000008130.14	0	1p36.33	a2	b2	...
ENSG00000067606.14	0	1p36.33	a3	b3	...
...	...	...	...	...	...

Table 5.5: cnv gene: GDC file (TCGA-LUSC with aliquot C, D)

Gene Symbol	Gene ID	Cytoband	AliquotId-C	AliquotId-D	...
ENSG00000008128.21	0	1p36.33	c1	d1	...
ENSG00000008130.14	0	1p36.33	c2	d2	...
ENSG00000067606.14	0	1p36.33	c3	d3	...
...	...	...	...	...	...

will be integrated by go2gdc into 1 project-level table (*.tsv) after:

1. skipping 2 columns
    1. "Gene ID"
    2. "Cytoband"
2. using new header row:
    - gene "Id"
    - aliquot "data_source_id" (Id-?)
        - "project ID" + "sample ID" + "UUID of source file"

Table 5.6: cnv gene: integrated table

Id	Id-A	Id-B	Id-C	Id-D	...
ENSG00000008128.21	a1	b1	c1	d1	...
ENSG00000008130.14	a2	b2	c2	d2	...
ENSG00000067606.14	a3	b3	c3	d3	...
...	...	...	...	...	...

Note:

1. only 1 "cnv_gene" file per TCGA project, which includes gene level "Copy Number Variation" data from multiple cases (aliquots).

5.3 cnv_segment_somatic_only

From GDC, the data files of "cnv_segment_somatic_only" are at aliquot level, i.e., the data in a single file comes from same sample.

With this filter-file:

```
omics_type = cnv_segment_somatic_only
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.7: cnv segment somatic only: GDC file (aliquot A)

GDC_Aliquot	Chromosome	Start	End	Num_Probes	Segment_Mean
AliquotId-A	Chr(region1)	Start(1)	End(1)	NumProbes(1)	a1
AliquotId-A	Chr(region2)	Start(2)	End(2)	NumProbes(2)	a2
AliquotId-A	Chr(region3)	Start(3)	End(3)	NumProbes(3)	a3
...	...	...	...	...	...

Table 5.8: cnv segment somatic only: GDC file (aliquot B)

GDC_Aliquot	Chromosome	Start	End	Num_Probes	Segment_Mean
AliquotId-B	Chr(region1)	Start(1)	End(1)	NumProbes(1)	b1
AliquotId-B	Chr(region3)	Start(3)	End(3)	NumProbes(3)	b3
AliquotId-B	Chr(region4)	Start(4)	End(4)	NumProbes(4)	b4
...	...	...	...	...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. skipping 2 column:
    1. "GDC_Aliquot"
    2. "Num_Probes"
2. uniting 3 columns:

    1. “Chromosome”
    2. “Start”
    3. “End”
3. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.9: cnv segment somatic only: integrated table

Id	Id-A	Id-B	...
Chr___Start___End(region1)	a1	b1	...
Chr___Start___End(region2)	a2		...
Chr___Start___End(region3)	a3	b3	...
Chr___Start___End(region4)		b4	...
...	...	...	...

5.4 cnv_segment_somatic_and_germline

From GDC, the data files of "cnv_segment_somatic_and_germline" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = cnv_segment_somatic_and_germline
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.10: cnv segment somatic and germline: GDC file (aliquot A)

GDC_Aliquot	Chromosome	Start	End	Num_Probes	Segment_Mean
AliquotId-A	Chr(region1)	Start(1)	End(1)	NumProbes(1)	a1
AliquotId-A	Chr(region2)	Start(2)	End(2)	NumProbes(2)	a2
AliquotId-A	Chr(region3)	Start(3)	End(3)	NumProbes(3)	a3
...	...	...	...	...	...

Table 5.11: cnv segment somatic and germline: GDC file (aliquot B)

GDC_Aliquot	Chromosome	Start	End	Num_Probes	Segment_Mean
AliquotId-B	Chr(region1)	Start(1)	End(1)	NumProbes(1)	b1
AliquotId-B	Chr(region3)	Start(3)	End(3)	NumProbes(3)	b3
AliquotId-B	Chr(region4)	Start(4)	End(4)	NumProbes(4)	b4
...	...	...	...	...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. skipping 2 column:
    1. "GDC_Aliquot"

    2. “Num_Probes”
2. uniting 3 columns:
    1. “Chromosome”
    2. “Start”
    3. “End”
3. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.12: cnv segment somatic and germline: integrated table

Id	Id-A	Id-B	...
Chr___Start___End(region1)	a1	b1	...
Chr___Start___End(region2)	a2		...
Chr___Start___End(region3)	a3	b3	...
Chr___Start___End(region4)		b4	...
...	...	...	...

5.5 gene_htseq_fpkm_uq

From GDC, the data files of "gene_htseq_fpkm_uq" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = gene_htseq_fpkm_uq
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.13: gene htseq fpkm uq: GDC file (aliquot A)

ENSG00000000003.13	a1
ENSG00000000005.5	a2
ENSG00000000419.11	a3
...	...

Table 5.14: gene htseq fpkm uq: GDC file (aliquot B)

ENSG00000000003.13	b1
ENSG00000000005.5	b2
ENSG00000000419.11	b3
...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. adding 1 heading row:
   - gene "Id"
   - aliquot "data_source_id" (Id-?)
     - "project ID" + "sample ID" + "UUID of source file"

Table 5.15: gene htseq fpkm uq: integrated table

Id	Id-A	Id-B	...
ENSG00000000003.13	a1	b1	...
ENSG00000000005.5	a2	b2	...
ENSG00000000419.11	a3	b3	...
...	...	...	...

5.6 gene_htseq_fpkm

From GDC, the data files of "gene_htseq_fpkm" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = gene_htseq_fpkm
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.16: gene htseq fpkm: GDC file (aliquot A)

ENSG00000000003.13	a1
ENSG00000000005.5	a2
ENSG00000000419.11	a3
...	...

Table 5.17: gene htseq fpkm: GDC file (aliquot B)

ENSG00000000003.13	b1
ENSG00000000005.5	b2
ENSG00000000419.11	b3
...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. adding 1 heading row:
    - gene "Id"
    - aliquot "data_source_id" (Id-?)
        - "project ID" + "sample ID" + "UUID of source file"

Table 5.18: gene htseq fpkm: integrated table

Id	Id-A	Id-B	...
ENSG00000000003.13	a1	b1	...
ENSG00000000005.5	a2	b2	...
ENSG00000000419.11	a3	b3	...
...	...	...	...

5.7 gene_htseq_counts

From GDC, the data files of "gene_htseq_counts" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = gene_htseq_counts
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.19: gene htseq counts: GDC file (aliquot A)

ENSG00000000003.13	a1
ENSG00000000005.5	a2
ENSG00000000419.11	a3
...	...

Table 5.20: gene htseq counts: GDC file (aliquot B)

ENSG00000000003.13	b1
ENSG00000000005.5	b2
ENSG00000000419.11	b3
...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. adding 1 heading row:
   - gene "Id"
   - aliquot "data_source_id" (Id-?)
     - "project ID" + "sample ID" + "UUID of source file"

2. skipping 5 rows:
    1. "__no_feature"
    2. "__ambiguous"
    3. "__too_low_aQual"
    4. "__not_aligned"
    5. "__alignment_not_unique"

Table 5.21: gene htseq counts: integrated table

Id	Id-A	Id-B	...
ENSG00000000003.13	a1	b1	...
ENSG00000000005.5	a2	b2	...
ENSG00000000419.11	a3	b3	...
...	...	...	...

5.8 gene_star_counts

From GDC, the data files of "gene_star_counts" are at aliquot level, i.e., the data in a single file comes from same sample.

The "gene_star_counts" is a special omics type, which has no TCGA data. So another project "NCICCR-DLBCL" is used for example.

When `omics_type = gene_star_counts`, the sample types related fields in filter-file will not work:

1. `sample_type_id =`
2. `sample_types_for_separated_integration =`
3. `sample_types_for_cases_intersection =`
4. `keep_samples_from_same_case =`

which means these 4 fields will be omitted.

Using filter-file like this:

```
omics_type = gene_star_counts
project_id = NCICCR-DLBCL
sample_type_id =
sample_types_for_separated_integration =
sample_types_for_cases_intersection =
omics_types_for_cases_intersection =
keep_samples_from_same_case =
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.22: gene star counts: GDC file (aliquot A)

#gene	unstranded	stranded_first	stranded_second
ENSG00000000003.13	a11	a12	a13
ENSG00000000005.5	a21	a22	a23
ENSG00000000419.11	a31	a32	a33
...	...	...	...

Table 5.23: gene star counts: GDC file (aliquot B)

#gene	unstranded	stranded_first	stranded_second
ENSG00000000003.13	b11	b12	b13
ENSG00000000005.5	b21	b22	b23
ENSG00000000419.11	b31	b32	b33
...	...	...	...

will be integrated by go2gdc into 1 table (*.tsv) after:

1. skipping 2 columns:
    1. "stranded_first"
    2. "stranded_second"
2. skipping 4 rows:
    1. "N_unmapped"
    2. "N_multimapping"
    3. "N_noFeature"
    4. "N_ambiguous"
3. using new header row:
    - gene "Id"
    - aliquot "data_source_id" (Id-?)
        - "project ID" + "sample ID" + "UUID of source file"

Table 5.24: gene star counts: integrated table

Id	Id-A	Id-B	...
ENSG00000000003.13	a11	b11	...
ENSG00000000005.5	a21	b21	...
ENSG00000000419.11	a31	b31	...
...	...	...	...

Note:

1. No TCGA project has "gene_star_counts" omics type data.

5.9 methy_27

From GDC, the data files of "methy_27" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = methy_27
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.25: methy 27: GDC file (aliquot A)

Composite Element REF	Beta_value	Annotations...
cg00000292	a1	Annotations-1
cg00002426	a2	Annotations-2
cg00003994	a3	Annotations-3
...	...	...

Table 5.26: methy 27: GDC file (aliquot B)

Composite Element REF	Beta_value	Annotations...
cg00000292	b1	Annotations-1
cg00002426	b2	Annotations-2
cg00003994	b3	Annotations-3
...	...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. uniting 11 columns
   - 1 id column:
     1. "Composite Element REF"
   - 10 annotation columns:

    1. “Beta_value”
    2. “Chromosome”
    3. “Start”
    4. “End”
    5. “Gene_Symbol”
    6. “Gene_Type”
    7. “Transcript_ID”
    8. “Position_to_TSS”
    9. “CGI_Coordinate”
    10. “Feature_Type”
2. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.27: methy 27: integrated table

Id___Annotations	Id-A	Id-B	...
cg00000292___Annotations-1	a1	b1	...
cg00002426___Annotations-2	a2	b2	...
cg00003994___Annotations-3	a3	b3	...
...	...	...	...

5.10 methy_450

From GDC, the data files of "methy_450" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = methy_450
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.28: methy 450: GDC file (aliquot A)

Composite Element REF	Beta_value	Annotations...
cg00000029	a1	Annotations-1
cg00000108	a2	Annotations-2
cg00000109	a3	Annotations-3
...	...	...

Table 5.29: methy 450: GDC file (aliquot B)

Composite Element REF	Beta_value	Annotations...
cg00000029	b1	Annotations-1
cg00000108	b2	Annotations-2
cg00000109	b3	Annotations-3
...	...	...

will be integrated by go2gdc into 1 aliquot-level table (*.tsv) after:

1. uniting 11 columns
   - 1 id column:
     1. "Composite Element REF"
   - 10 annotation columns:

    1. “Beta_value”
    2. “Chromosome”
    3. “Start”
    4. “End”
    5. “Gene_Symbol”
    6. “Gene_Type”
    7. “Transcript_ID”
    8. “Position_to_TSS”
    9. “CGI_Coordinate”
    10. “Feature_Type”
2. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.30: methy 450: integrated table

Id___Annotations	Id-A	Id-B	...
cg00000029___Annotations-1	a1	b1	...
cg00000108___Annotations-2	a2	b2	...
cg00000109___Annotations-3	a3	b3	...
...	...	...	...

5.11 mir

From GDC, the data files of "mir" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = mir
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.31: mir: GDC file (aliquot A)

miRNA_ID	read_count	reads_per_million_miRNA_mapped	cross-mapped
hsa-let-7a-1	a11	a12	a13
hsa-let-7a-2	a21	a22	a23
hsa-let-7a-3	a31	a32	a33
...	...	...	...

Table 5.32: mir: GDC file (aliquot B)

miRNA_ID	read_count	reads_per_million_miRNA_mapped	cross-mapped
hsa-let-7a-1	b11	b12	b13
hsa-let-7a-2	b21	b22	b23
hsa-let-7a-3	b31	b32	b33
...	...	...	...

will be integrated by go2gdc into 3 aliquot-level tables (*.tsv):

1. of 3 data columns respectively:
    1. "read_count"
    2. "reads_per_million_miRNA_mapped"
    3. "cross-mapped"

2. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.33: mir: integrated table of ”read count”

Id	Id-A	Id-B	...
hsa-let-7a-1	a11	b11	...
hsa-let-7a-2	a21	b21	...
hsa-let-7a-3	a31	b31	...
...	...	...	...

Table 5.34: mir: integrated table of ”reads per million miRNA mapped”

Id	Id-A	Id-B	...
hsa-let-7a-1	a12	b12	...
hsa-let-7a-2	a22	b22	...
hsa-let-7a-3	a32	b32	...
...	...	...	...

Table 5.35: mir: integrated table of ”corss mapped”

Id	Id-A	Id-B	...
hsa-let-7a-1	a13	b13	...
hsa-let-7a-2	a23	b23	...
hsa-let-7a-3	a33	b33	...
...	...	...	...

5.12 mir_isoform

From GDC, the data files of "mir_isoform" are at aliquot level, i.e., the data in a single file comes from same sample.

Using filter-file like this:

```
omics_type = mir_isoform
project_id = TCGA-LUAD, TCGA-LUSC
sample_type_id = 01, 11
sample_types_for_separated_integration = true
sample_types_for_cases_intersection = true
omics_types_for_cases_intersection = true
keep_samples_from_same_case = none
case_id =
```

The downloaded GDC files from different aliquots:

Table 5.36: mir isoform: GDC file (aliquot A)

miRNA_ID	isoform_coords	read_count	reads_..._mapped	cross-mapped	miRNA_region
hsa-let-7a-1	coords-1	a11	a12	a13	region-1
hsa-let-7a-1	coords-2	a21	a22	a23	region-2
hsa-let-7a-1	coords-3	a31	a32	a33	region-3
...	...	...	...	...	...

Table 5.37: mir isoform: GDC file (aliquot B)

miRNA_ID	isoform_coords	read_count	reads_..._mapped	cross-mapped	miRNA_region
hsa-let-7a-1	coords-1	b11	b12	b13	region-1
hsa-let-7a-1	coords-2	b21	b22	b23	region-2
hsa-let-7a-1	coords-4	b41	b42	b43	region-4
...	...	...	...	...	...

will be integrated by go2gdc into 3 aliquot-level tables (*.tsv):

1. of 3 data columns respectively:
    1. "read_count"
    2. "reads_per_million_miRNA_mapped"
    3. "cross-mapped"

2. using new header row:
    - gene “Id”
    - aliquot “data_source_id” (Id-?)
        - “project ID” + “sample ID” + “UUID of source file”

Table 5.38: mir isoform: integrated table of ”read count”

Id	Id-A	Id-B	...
hsa-let-7a-1___coords-1___region-1	a11	b11	...
hsa-let-7a-1___coords-2___region-2	a21	b21	...
hsa-let-7a-1___coords-3___region-3	a31		...
hsa-let-7a-1___coords-4___region-4		b41	...
...	...	...	...

Table 5.39: mir isoform: integrated table of ”reads per million miRNA mapped”

Id	Id-A	Id-B	...
hsa-let-7a-1___coords-1___region-1	a12	b12	...
hsa-let-7a-1___coords-2___region-2	a22	b22	...
hsa-let-7a-1___coords-3___region-3	a32		...
hsa-let-7a-1___coords-4___region-4		b42	...
...	...	...	...

Table 5.40: mir isoform: integrated table of ”corss mapped”

Id	Id-A	Id-B	...
hsa-let-7a-1___coords-1___region-1	a13	b13	...
hsa-let-7a-1___coords-2___region-2	a23	b23	...
hsa-let-7a-1___coords-3___region-3	a33		...
hsa-let-7a-1___coords-4___region-4		b43	...
...	...	...	...

Authorship

Affiliations

State Key Laboratory of Oncogenes and Related Genes, Shanghai Cancer Institute, Renji Hospital, Shanghai Jiao Tong University School of Medicine, Shanghai, China.

Lin Wei

Contribution

L.W. designed, created the package and documents.

Correspondence

Correspondence to Lin Wei

Competing interests

The author declare no competing interests.

Acknowledgment

This work is a memorial to my mother, Weijuan Wang (1948-2014), who always encouraged me to be honest and brave.

References

1. Jensen, Mark A; Ferretti, Vincent; Grossman, Robert L; Staudt, Louis M; The nci genomic data commons as an engine for precision medicine, *Blood*, 2017, 130(4):453–459. https://doi.org/10.1182/blood-2017-03-735654

2. Goldman, Mary; Craft, Brian; Zhu, Jingchun; Haussler, David; Abstract 2584: The ucsc xena system for cancer genomics data visualization and interpretation, *Cancer Research*, 2017, 2017:Abstract nr 2584. https://doi.org/10.1158/1538-7445.am2017-2584

3. R Core Team; R: A language and environment for statistical computing, 2019, https://www.R-project.org/

4. Wei, Lin; Jin, Zhilin; Yang, Shengjie; Xu, Yanxun; Zhu, Yitan; Ji, Yuan; TCGA-assembler 2: Software pipeline for retrieval and processing of tcga/cptac data, *Bioinformatics*, 2018, 34(9):1615–1617. https://doi.org/10.1093/bioinformatics/btx812

5. Zhu, Yitan; Qiu, Peng; Ji, Yuan; TCGA-assembler: Open-source software for retrieving and processing tcga data, *Nature Methods*, 2014, 11:599–600. https://doi.org/10.1038/nmeth.2956

6. Mounir, Mohamed; Lucchetta, Marta; Silva, Tiago C.; Olsen, Catharina; Bontempi, Gianluca; Chen, Xi; Noushmehr, Houtan; Colaprico, Antonio; Papaleo, Elena; New functionalities in the tcgabiolinks package for the study and integration of cancer data from gdc and gtex, *PLoS Computational Biology*, 2019, 15:e1006701. https://doi.org/10.1371/journal.pcbi.1006701

7. Colaprico, Antonio; Silva, Tiago C.; Olsen, Catharina; Garofano, Luciano; Cava, Claudia; Garolini, Davide; Sabedot, Thais S.; Malta, Tathiane M.; Pagnotta, Stefano M.; Castiglioni, Isabella; Ceccarelli, Michele; Bontempi, Gianluca; Noushmehr, Houtan; TCGAbiolinks: An r/bioconductor package for integrative analysis of tcga data, *Nucleic Acids Research*, 2016, 44(8):e71. https://doi.org/10.1093/nar/gkv1507