refine.bio Documentation

Release 0.1.0

The Childhood Cancer Data Lab

Nov 14, 2018

Contents

refine.bio is a multi-organism collection of genome-wide transcriptome or gene expression data that has been obtained from publicly available repositories and uniformly processed and normalized. refine.bio allows biologists, clinicians, and machine learning researchers to search for experiments from different source repositories all in one place and build custom data sets for their questions of interest.

refine.bio is well-suited for quickly assessing if signals are present in particular datasets, for identifying and obtaining data sets for accelerated validation of findings, and for building large compendia for training machine learning models that can adequately handle the technical noise associated with integrating multiple experiments and platforms. For examples of how to use refine.bio data, please see our use cases for downstream analysis. refine.bio is *not* a substitute for experiments and processing pipelines tailored to answer *specific* biological questions of interest or for input from relevant experts (e.g., those with statistics expertise), but rather a repository of samples processed with standardized pipelines that have been selected based on their wide-ranging utility.

CHAPTER 1

Frequently asked questions

CHAPTER 2

Contents

2.1 Source Data

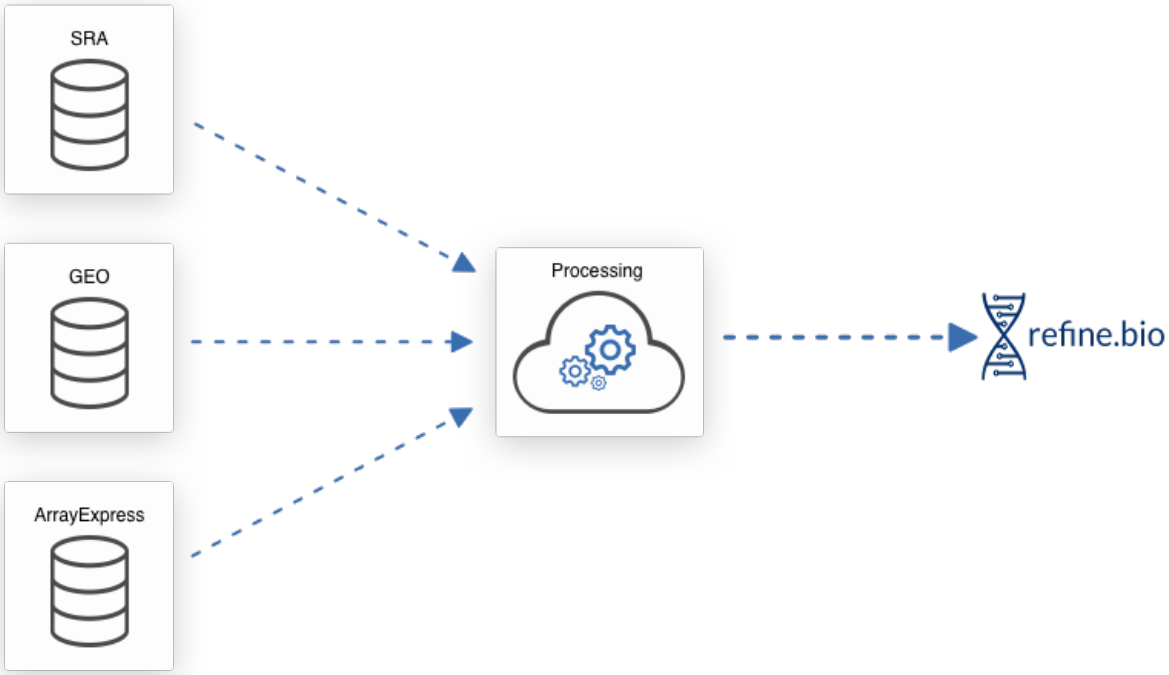
2.1.1 Types of Data

The current version of refine.bio is designed to process gene expression data. This includes both microarray data and RNA-seq data. We normalize data to Ensembl gene identifiers and provide abundance estimates.

More precisely, we support microarray platforms based on their GEO or ArrayExpress accessions. We currently support Affymetrix microarrays and Illumina BeadArrays, and we are continuing to evaluate and add support for more platforms. This table contains the microarray platforms that we support. We process a subset of platforms using the BrainArray Custom CDFs, which are denoted by a `y` in the `is_brainarray` column. We also support RNA-seq experiments performed on these short-read platforms. For more information on how data are processed, see the *Processing Information* section of this document. If there is a platform that you would like to see processed, please file an issue on GitHub. If you would prefer to report issues via e-mail, you can also email ccdl@alexslemonade.org.

2.1.2 Sources

We download gene expression information and metadata from EBI's ArrayExpress, NCBI's Gene Expression Omnibus (GEO), and NCBI's Sequence Read Archive (SRA). NCBI's SRA also contains experiments from EBI's ENA (example) and the DDBJ (example).

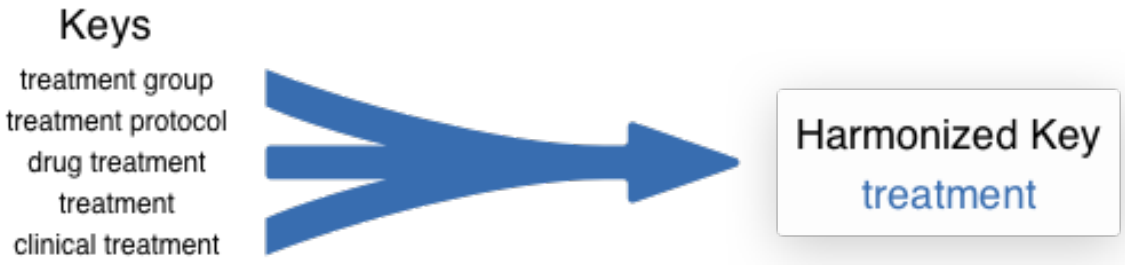


2.1.3 Metadata

We provide metadata that we obtain from the source repositories. We also attempt to, where possible, perform some harmonization of metadata to improve the discoverability of useful data. Note that we do not yet obtain sample metadata from the BioSample database, so the metadata available for RNA-seq samples is limited.

refine.bio-harmonized Metadata

Scientists who upload results don't always use the same names for related values. This makes it challenging to search across datasets. We have put some processes in place to smooth out some of these issues.



To produce lightly harmonized metadata, we combine certain fields based on similar keys. We do this for convenience and to aid in searches. For example, `treatment`, `treatment group`, `treatment protocol`, `drug treatment`, and `clinical treatment` fields get collapsed down to `treatment`. The fields that we currently collapse to includes `specimen part`, `genetic information`, `disease`, `disease stage`, `treatment`, `race`, `subject`, `development stage`, `compound`, `cell_line`, and `time`.

See the table below for a complete set of mappings between the keys from source data and the harmonized keys. Values are stripped of white space and forced to lowercase.

We type-cast age values to doubles. Sex is a special case; we map to `female` and `male` values if the values are one of the following:

Only harmonized values are displayed in the sample table on the web interface. When downloading refine.bio data, these harmonized metadata are denoted with the `refinebio_` prefix.

We recommend that users confirm metadata fields that are particularly important via the submitter-supplied metadata. If you find that the harmonized metadata does not accurately reflect the metadata supplied by the submitter, please file an issue on GitHub so that we can resolve it. If you would prefer to report issues via e-mail, you can also email ccdl@alexslemonade.org.

Submitter Supplied Metadata

We also capture the metadata as submitted to the source repositories. This includes experiment titles and descriptions or abstracts. Protocol information, which generally contains the type of sample preparation, is handled differently by different databases. We push this information down to the sample level and provide it that way. In the case of data imported from ArrayExpress, the protocol *may* be the information from the full experiment and not just the sample in question. Sample metadata in their original, unharmonized form are available as part of *refine.bio downloads*.

2.2 Processing Information

2.2.1 refine.bio processed

Because refine.bio is designed to be consistently updated, we use processing and normalization methods that operate on a single sample wherever possible. Processing and normalization methods that require multiple samples (e.g., Robust Multi-array Average or RMA) generally have outputs that are influenced by whatever samples are included and rerunning these methods whenever a new sample is added to the system would be impractical.

Microarray pipelines

Affymetrix

SCAN (Single Channel Array Normalization) is a normalization method for develop for single channel Affymetrix microarrays that allows us to process individual samples. SCAN models and corrects for the effect of technical bias, such as GC content, using a mixture-modeling approach. For more information about this approach, see the primary publication (Piccolo, et al. *Genomics.* 2012.) and the SCAN.UPC Bioconductor package documentation. We specifically use the `SCANfast` implementation of SCAN and the Brainarray packages as probe-summary packages when available.

Platform detection

We have encountered instances where the platform label from the source repository and the metadata included in the sample's raw data file (`.CEL` file) itself do not match. In these cases, we take the platform information included in the raw data (`.CEL`) file header to be the true platform label.

Illumina BeadArrays

Dr. Stephen Piccolo, the developer of SCAN, has adapted the algorithm for use with Illumina BeadArrays for refine.bio. Because this Illumina SCAN methodology is not yet incorporated into the SCAN.UPC package, we briefly summarize the methods below.

We require that non-normalized or raw expression values and detection p-values to be present in Illumina non-normalized data. If we infer that background correction has not occurred in the non-normalized data (e.g., there are no negative expression values), the data are background corrected using the `limma::nec` function (Shi, Oshlack, and Smyth. *Nucleic Acids Research.* 2010.). Following background correction – either upstream presumably in the Illumina BeadStudio software or in our processor, arrays are normalized with SCAN. SCAN requires probe sequence information obtained from the Illumina BeadArray Bioconductor annotation packages (e.g., `illuminaHumanv1.db`). We only retain probes that have a "Good" or "Perfect" rating in these packages; this quality rating is in reference to how well a probe is likely to measure its target transcript.

Platform detection

We infer the Illumina BeadArray platform that a sample is likely to be run on by comparing the probe identifiers in the unprocessed file to probes for each of the Illumina expression arrays for a given organism. We again use the Illumina Bioconductor annotation packages for this step. For instance, the overlap between the probe identifiers in a human sample and the probe identifiers in each human platform (`v1`, `v2`, `v3`, and `v4`) is calculated. The platform with the highest overlap (provided it is >75%) is inferred to be the true platform. Some analyses around this platform detection procedure can be found in this repository.

RNA-seq pipelines

We use Salmon and tximport to process all RNA-seq data in refine.bio. We obtain sra files run on our supported short-read platforms from NCBI Sequence Read Archive and use `fasterq-dump` (with default behavior) to obtain fastq files for use with Salmon. Note that any unmated reads from paired experiments are discarded.

We use the library strategy and library source metadata fields to identify RNA-seq experiments. It's possible that experiments that are inappropriate for use with Salmon will still appear in refine.bio (e.g., long-read platforms that are labeled incorrectly in the source repository). If you find an experiment that you believe is inappropriate for use with Salmon, please file an issue on GitHub so that we can resolve it. If you would prefer to report issues via e-mail, you can also email ccdl@alexslemonade.org.

Salmon

Salmon is an alignment-free method for estimating transcript abundances from RNA-seq data (Patro, et al. *Nature Methods*. 2017.). We use it in quasi-mapping mode, which is significantly faster than alignment-based approaches and requires us to build a Salmon transcriptome index.

Transcriptome index

We build a custom reference transcriptome (using RSEM `rsem-prepare-reference`) by filtering the Ensembl genomic DNA assembly to remove *pseudogenes*, which we expect could negatively impact the quantification of protein-coding genes. This means we're obtaining abundance estimates for coding as well as non-coding transcripts.

Building a transcriptome index with `salmon index` requires us to specify a value for the parameter `-k` that determines the size of the k-mers used for the index. The length of a read determines what k-mer size is appropriate. Consistent with the recommendations of the authors of Salmon, we use an index build with $k = 31$ when quantifying samples with reads with length > 75bp. We use $k = 23$ for shorter read lengths.

The refine.bio processed Salmon indices are available for download. You can make use of our API like so:

```
https://api.refine.bio/transcriptome_indices/?organism=<ORGANISM>&length=<LENGTH>
```

Where `<ORGANISM>` is the scientific name of the species in all caps separated by underscores and `<LENGTH>` is either `SHORT` or `LONG`.

To obtain the zebrafish (*Danio rerio*) index used for >75bp reads, use:

```
https://api.refine.bio/transcriptome_indices/?organism=DANIO_RERIO&length=LONG
```

The `s3_url` field will allow you to download the index.

Quantification with Salmon

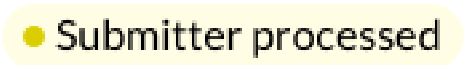
When quantifying transcripts with `salmon quant`, we take advantage of options that allow Salmon to learn and attempt to correct for certain biases in sequencing data. We include the flags `--seqBias` to correct for random hexamer priming and, if this is a **paired-end** experiment, `--gcBias` to correct for GC content when running salmon quant. We set the library type parameter such that Salmon will infer the sequencing library type automatically for the reads it is quantifying (`-l A`).

tximport

Salmon quantification is at the *transcript-level*. To better integrate with the microarray data contained in refine.bio, we summarize the transcript-level information to the *gene-level* with `tximport` (Soneson, Love, and Robinson. *F1000 Research.* 2015.).

Our tximport implementation generates "lengthScaledTPM", which are gene-level count-scale values that are generated by scaling TPM using the average transcript length across samples and to the library size. Note that tximport is applied at the *experiment-level* rather than to single samples. For additional information, see the tximport Bioconductor page , the tximport tutorial *Importing transcript abundance datasets with tximport*, and Soneson, Love, and Robinson. *F1000Research.* 2015..

2.2.2 Submitter processed



Sometimes raw data for a sample is either unavailable at the source repository or exists in a form that we can not process. For microarray platforms that we support, we obtain the submitter processed expression data and use these values in refine.bio with some modification (e.g., log2-transformation where we detect it has not been performed).

As noted above, we use Ensembl gene identifiers throughout refine.bio. Submitter processed data may use other gene (or probe) identifiers that we must convert to Ensembl gene identifiers. We describe the processes for Affymetrix and Illumina data below. Note in the case of one-to-many mappings when going from the ID used by the submitter to the

Ensembl gene ID, expression values are duplicated: if a probe maps to two Ensembl gene ids, those two Ensembl gene ids will both have the probe's expression value following conversion.

Affymetrix

We have created custom gene mapping files for most of the Affymetrix platforms we support. Briefly, for Brainarray supported platforms, we use the Brainarray (e.g., `hgu133plus2hsensgprobe`) and the platform-specific annotation package from Bioconductor (e.g., `hgu133plus2.db`) to generate a platform-specific mapping file that includes probe IDs, Ensembl gene IDs, gene symbols, Entrez IDs, RefSeq and Unigene identifiers. The rationale for only using probes or IDs that are accounted for in the Brainarray package is two-fold: 1) Brainarray packages are updated as we learn more about the genome and 2) it allows for these submitter processed data to be more consistent with refine.bio processed data. We support identifier conversion for a limited number of platforms that either do not have a Brainarray or Bioconductory annotation packages.

The code for deriving these mappings and more details are available at https://github.com/AlexsLemonade/identifier-refinery. If you find an issue with these mappings, please file an issue on GitHub so that we can resolve it. If you would prefer to report issues via e-mail, you can also email ccdl@alexslemonade.org.

Illumina

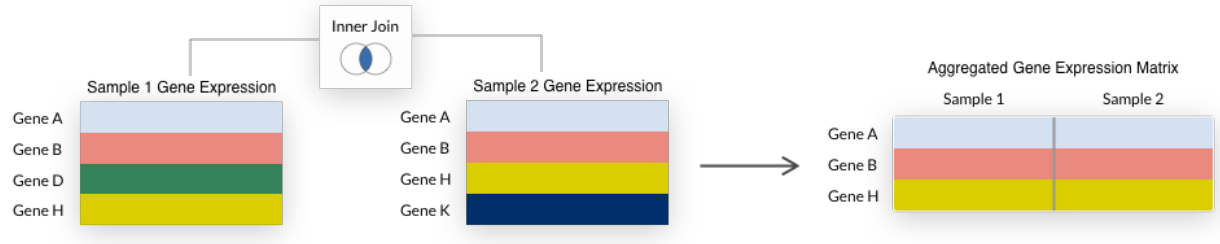
We support conversion from Illumina BeadArray probe IDs to Ensembl gene IDs using Bioconductor Illumina BeadArray expression packages, allowing for one-to-many mappings.

2.2.3 Aggregations

refine.bio allows users to aggregate their selected samples in two ways: by experiment or by species. We use the term aggregate or aggregation to refer to the process of combining *individual samples* to form a *multi-sample* gene expression matrix (see also: *Downloadable Files*).

- **By experiment:** Samples that belong to the same experiment will become a single gene expression matrix. If you have selected all samples from two experiments with 10 and 15 samples, respectively, and have chosen the `by experiment` option, you will receive two gene expression matrices with 10 and 15 samples, respectively.
- **By species:** All samples assaying the same species will be aggregated into a single gene expression matrix. If you have selected three experiments each from human and mouse and the `by species` option, you receive two gene expression matrices that contain all human and all mouse samples, respectively.

For either aggregation method, we summarize duplicate Ensembl gene IDs to the mean expression value and only include genes (rows) that are represented in **all** samples being aggregated. This is also known as an inner join and is illustrated below.



Note that some early generation microarrays measure fewer genes than their more recent counterparts, so their inclusion when aggregating `by species` may result in a small number of genes being returned.

Limitations of gene identifiers when combining across platforms

We use Ensembl gene identifiers across refine.bio and, within platform, we use the same annotation to maintain consistency (e.g., all samples from the same Affymetrix platform use the same Brainarray package or identifier mapping derived from said package). However, Brainarray packages or Bioconductor annotation packages may be assembled from different genome builds compared each other or compared to the genome build used to generate transcriptome indices. If there tend to be considerable differences between (relatively) recent genome builds for your organism of interest or you are performing downstream analysis that would be sensitive to these differences, we do not recommend aggregating by species.

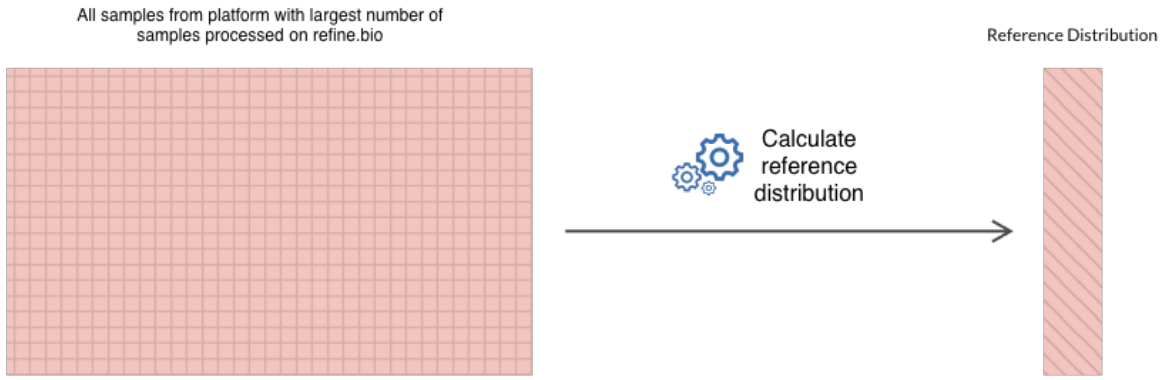
2.2.4 Transformations

Quantile normalization

refine.bio is designed to allow for the aggregation of multiple platforms and even multiple technologies. With that in mind, we would like the distributions of samples from different platforms/technologies to be as similar as possible. We use quantile normalization to accomplish this. Specifically, we generate a reference distribution for each organism from a large body of data with the `normalize.quantiles.determine.target` function from the preprocessCore R package and quantile normalize samples that a user selects for download with this target (using the `normalize.quantiles.use.target` function of `preprocessCore`). We go into more detail below.

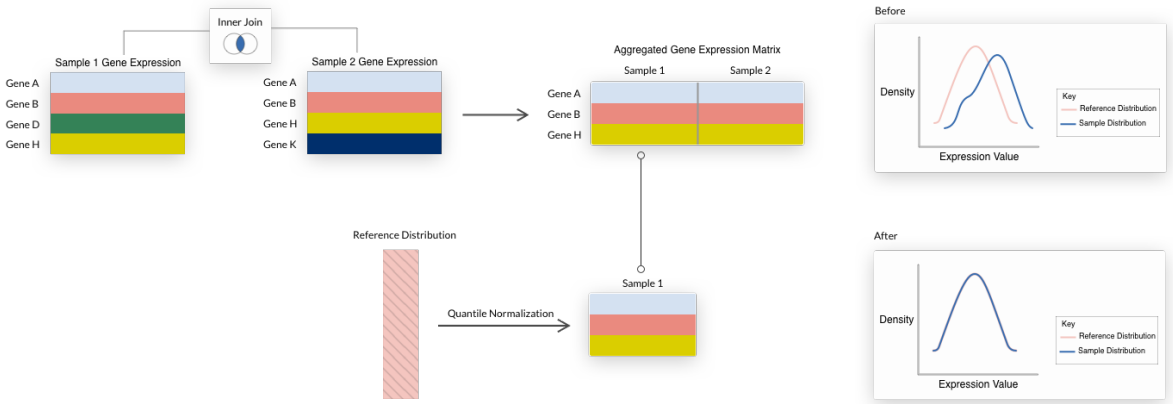
Reference distribution

By performing quantile normalization, we assume that the differences in expression values between samples arise solely from technical differences. This is not always the case; for instance, samples included in refine.bio are from multiple tissues. We'll use as many samples as possible to generate the reference or target distribution. By including as diverse biological conditions as we have available to us to inform the reference distribution, we attempt to generate a tissue-agnostic consensus. To that end, we use the Affymetrix microarray platform with the largest number of samples for a given organism (e.g., `hgu133plus2` in humans) and only samples we have processed from raw as shown below.



Quantile normalizing samples for delivery

Once we have a reference/target distribution for a given organism, we use it to quantile normalize any samples that a user has selected for download. This quantile normalization step takes place *after* the summarization and inner join steps described above and illustrated below.



As a result of the quantile normalization shown above, Sample 1 now has the same underlying distribution as the reference for that organism.

Note that only gene expression matrices that we are able to successfully quantile normalize will be available for download.

Limitations of quantile normalization across platforms with many zeroes

Quantile normalization is a strategy that can address many technical effects, generally at the cost of retaining certain sources of biological variability. However, in cases where there are many ties that are different between samples, the transformation can produce outputs with different distributions. This arises in data processed by refine.bio when RNA-seq and microarray data are combined. To confirm that we have quantile normalized data correctly before returning results to the user, we evaluate the top half of expression values and confirm that a KS test produces a non-significant p-value. Users who seek to analyze RNA-seq and microarray data together should be aware that the low-expressing genes may not be comparable across the sets.

Gene transformations

In some cases, it may be useful to row-normalize or transform the gene expression values in a matrix (e.g., following aggregation and quantile normalization). We offer the following options for transformations:

- **None:** No row-wise transformation is performed.
- **Z-score:** Row values are z-scored using the `StandardScaler` from `scikit-learn`. This transformation is useful for examining samples' gene expression values relative to the rest of the samples in the expression matrix (either all selected samples from that *species* when aggregating by species or all selected samples in an *experiment* when aggregating by experiment). If a sample has a positive value for a gene, that gene is more highly expressed in that sample compared to the mean of all samples; if that value is negative, that gene is less expressed compared to the population. It assumes that the data are normally distributed.
- **Zero to one:** Rows are scaled to values `[0,1]` using the `MinMaxScaler` from `scikit-learn`. We expect this transformation to be most useful for certain machine learning applications (e.g., those using cross-entropy as a loss function).

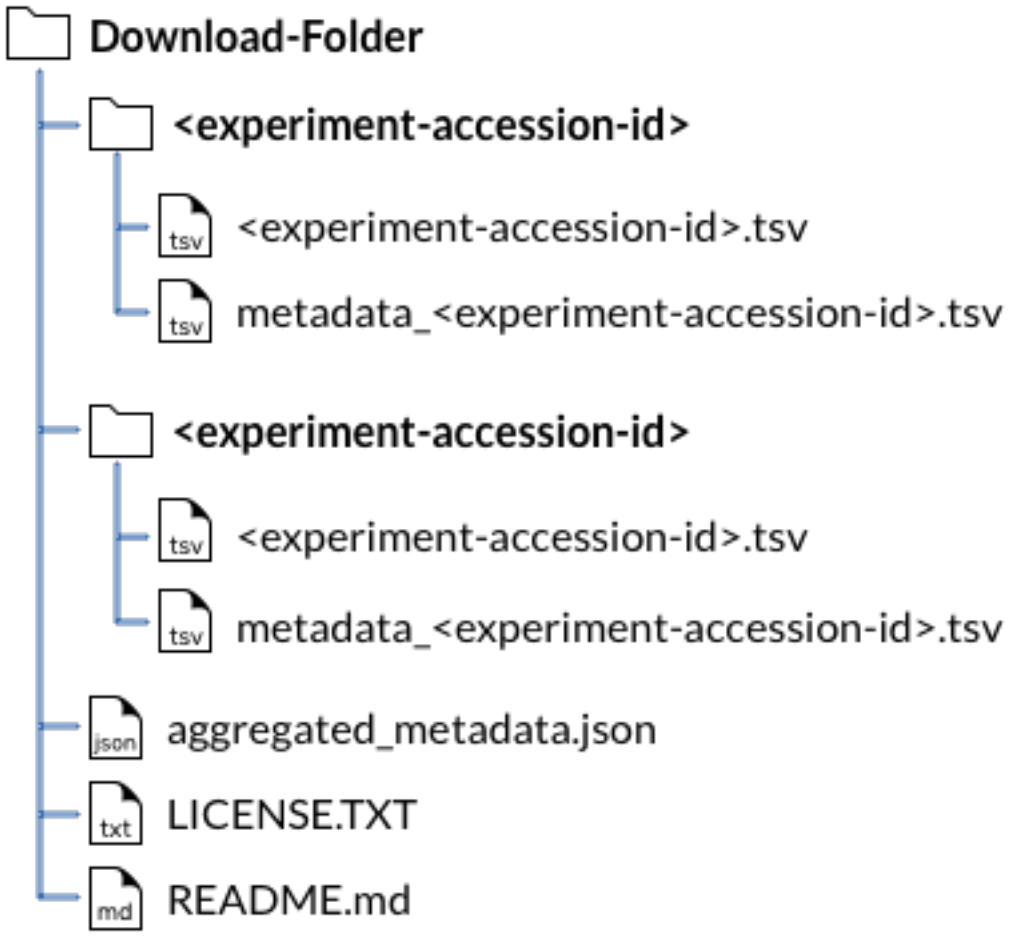
In the plot below, we demonstrate the effect of different scaling options on gene expression values (using a randomly selected human dataset, microarray platform, and gene):

Note that the distributions retain the same general *shape*, but the range of values and the density are altered by the transformations.

2.3 Downloadable Files

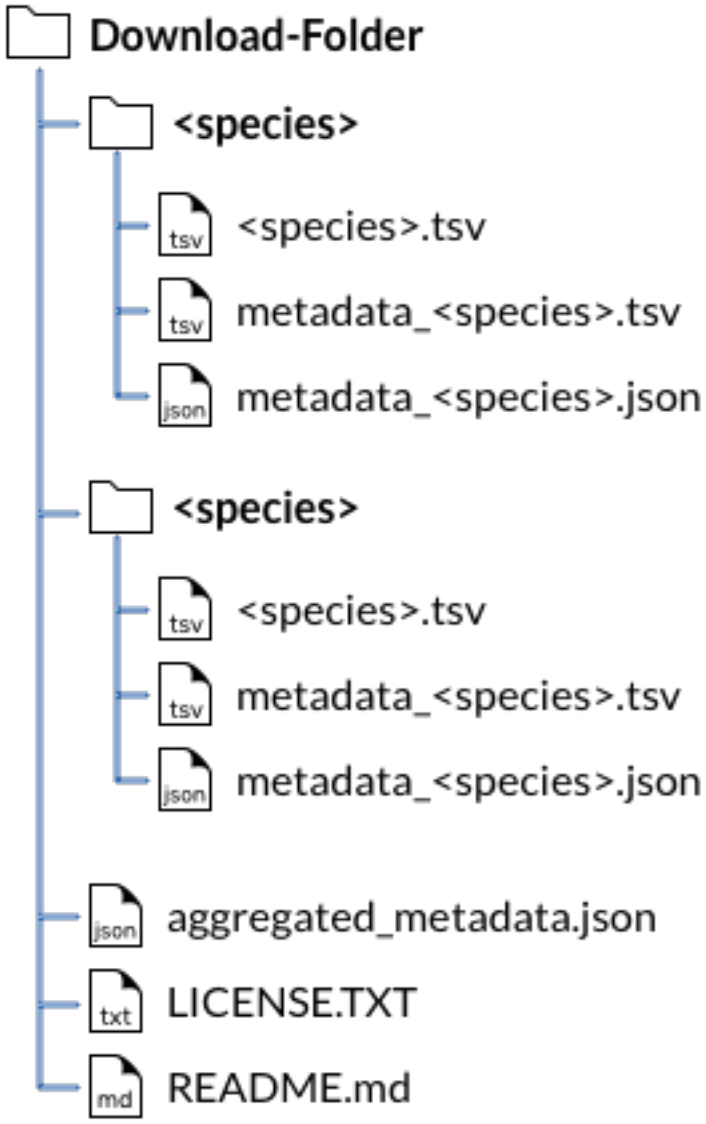
Users can download gene expression data and associated sample and experiment metadata from refine.bio. These files are delivered as a zip file. The folder structure within the zip file is determined by whether a user selected to aggregate by **experiment** or by **species**.

2.3.1 The download folder structure for data aggregated by experiment:



In this example, two experiments were selected. There will be as many folders as there are selected experiments.

2.3.2 The download folder structure for data aggregated by species:



In this example, samples from two species were selected. There will be as many folders as there are selected experiments and this will be the case regardless of how many individual experiments were included.

In both cases, `aggregated_metadata.json` contains metadata, including both *experiment* metadata (e.g., experiment description and title) and *sample* metadata for everything included in the download. Below we describe the files included in the delivered zip file.

2.3.3 Gene Expression Matrix

Gene expression matrices are delived in tab-separated value (TSV) format. In these matrices, rows are *genes* or *features* and columns are *samples*. Note that this format is consistent with the input expected by many programs specifically designed for working with gene expression matrices, but some machine learning libraries will expect

this to be transposed. The column names or header will contain values corresponding to sample accessions (denoted `refinebio_accession_code` in metadata files). You can use these values in the header to map between a sample's gene expression data and its metadata (e.g., disease label or age). See also *Use Cases for Downstream Analysis*.

2.3.4 Sample Metadata

Sample metadata is delivered in the `metadata_<experiment-accession-id>.tsv`, `metadata_<species>.json`, `metadata_<species>.tsv`, and `aggregated_metadata.json` files.

The primary way we identify samples is by using the sample accession, denoted by `refinebio_accession_code`. Harmonized metadata fields (see the *section on harmonized metadata*) are noted with a `refinebio_` prefix. The `refinebio_source_archive_url` and `refinebio_source_database` fields indicate where the sample was obtained from. If there are no keys from the source data associated with a harmonized key, the harmonized metadata field will be empty. We also deliver submitter-supplied data; see below for more details. **We recommend that users confirm metadata fields that are particularly important via the submitter-supplied metadata.** If you find that refine.bio metadata does not accurately reflect the metadata supplied by the submitter, please file an issue on GitHub so that we can resolve it. If you would prefer to report issues via e-mail, you can also email ccdl@alexslemonade.org.

TSV files

In metadata TSV files, samples are represented as rows. The first column contains the `refinebio_accession_code` field, which match the header/column names in the gene expression matrix, followed by refine.bio-harmonized fields (e.g., `refinebio_`), and finally submitter-supplied values. Some information from source repositories comes in the form of nested values, which we attempt to "flatten" where possible. Note that some information from source repositories is redundant–ArrayExpress samples often have the same information in `characteristic` and `variable` fields–and we *assume* that if a field appears in both, the values are identical. For samples run on Illumina BeadArray platforms, information about what platform we detected and the metrics used to make that determination will also be included.

Columns in these files will often have missing values, as not all fields will be available for every sample included in a file. This will be particularly evident when aggregating by experiments that have different submitter-supplied information associated with them (e.g., one experiment contains a `imatinib` key and all others do not).

JSON files

Submitter supplied metadata and source urls are delivered in `refinebio_annotations`. As described above, harmonized values are noted with a `refinebio_` prefix.

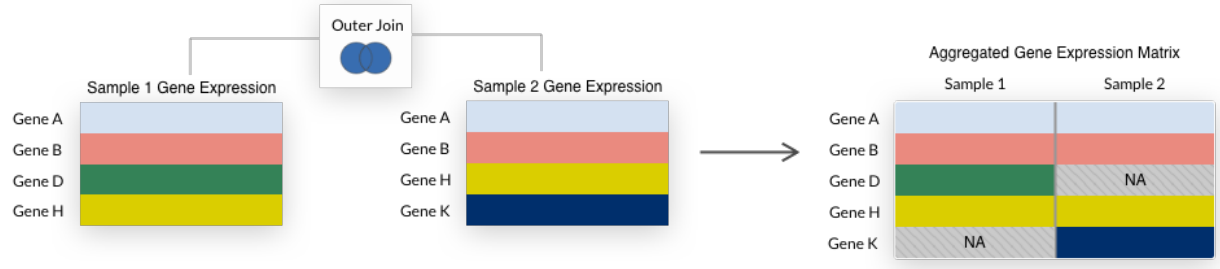
2.3.5 Experiment Metadata

Experiment metadata (e.g., experiment description and title) is delivered in the `metadata_<species>.json` and `aggregated_metadata.json` files.

2.4 Species compendia

refine.bio is currently in beta. Once refine.bio reaches production, we will periodically release compendia comprised of all the samples from a species that we were able to process. We refer to these as **species compendia**. We'll process

these compendia in a manner that is different from the options that are available via the web user interface. We describe our intended processing pipeline below. Instead of selecting only genes available in all samples, we take the union of all genes, filling in any missing values with `NA` (e.g., perform a full outer join as illustrated below).



We drop any genes that have missing values in greater than 30% of samples. We impute the remaining missing values with KNN impute. We then quantile normalize all samples as described above.

2.5 Use Cases for Downstream Analysis

Our `refinebio-examples` repo includes a number of different analyses you can perform with data from refine.bio. We include examples in the R programming language, and where applicable, GenePattern Notebooks and scripts to prepare refine.bio data for use with GenePattern. The following examples are included:

- Differential expression analysis [README, notebook]
- Converting between different gene identifiers [README, notebook]
- Ortholog mapping [README, notebook]
- Clustering/heatmap generation [README, notebook]

2.6 License

2.7 Frequently Asked Questions

2.7.1 What is the difference between refine.bio-processed and submitter-processed datasets?

Samples and the datasets they comprise are designated as refine.bio-processed if we were able to obtain raw data in a suitable format for one of our processing pipelines. If no suitable raw data is available for a sample on a supported platform, we obtain the *processed* data available from the source repository and modify it to be more consistent with refine.bio-processed data; these samples are termed submitter-processed. See the **Source Data** section for more information.

2.7.2 How do you process the data?

We process samples run on supported microarray platforms with Single-Channel Array Normalization (SCAN) and transcriptomic samples run on supported sequencing platforms with Salmon and tximport. Please see the **Processing Information** section for more details.

2.7.3 What type of data does refine.bio support?

refine.bio currently supports gene expression data, specifically genome-scale microarray and RNA-seq data. See our supported microarray and RNA-seq platforms.

2.7.4 What does "corrected" metadata mean?

Scientists will often use different terminology to refer to a similar sample metadata field or *key*. For example, `treatment` and `treatment protocol` may make reference to the same kinds of information. We attempt to perform some mapping between keys to aid in searches. See the **refine.bio-harmonized Metadata** section for more information, including a full list of the mappings we perform.

2.7.5 Why do the values differ a little bit if I download different datasets?

We've prioritized keeping expression values consistent *within* a dataset based on the samples it contains. Specifically, we remove any genes that are not measured in every sample in a dataset and we do that prior to performing quantile normalization. When using quantile normalization, the expression value a gene is assigned in a particular sample depends on the *rank* of that gene. If a user download different datasets, which may have different numbers of genes, it's possible then that the same gene in the same sample would have a different expression value between them.

2.7.6 Why do I get a limited number of genes back when I aggregate samples from different platforms?

Different platforms will often measure different sets of genes. These differences can be particularly pronounced when comparing older microarray platforms to more recent platforms. When aggregating samples, we retain *only* the genes present in *every sample*. If the dataset delivered to you has fewer genes than you were expecting for that organism, it could be the result of combining multiple platforms or it may be from an older microarray platform.

2.7.7 Why can't I add certain samples to my dataset?

refine.bio will sometimes obtain the metadata (e.g., sample title or experimental protocol) associated with a sample but the raw or submitter processed expression data files are in a format that we can not process. We do not allow you to add these samples to your dataset because we can not deliver expression values.

2.7.8 How can I find out what versions of software/packages were used to process the data?

Version information for the packages we think are *most important* for data processing is available on the pop-up displayed when you click a sample's processing information link.

The same package information is in the processor list available via our API:

```
https://api.refine.bio/processors/
```

In addition, you may wish to obtain our Docker images (prefixed with `dr_`) which will allow you to access version information for every dependency.

CHAPTER 3

Questions/Feedback?

If you have a question or comment, please file an issue on GitHub or send us an email at ccdl@alexslemonade.org.