Resolve the inconsistency of Illumina identifiers through nuID

Pan Du[‡,*], Warren A. Kibbe[‡,†], Gang Feng[‡,‡], Jared Flatow[‡,§], Simon Lin[‡,¶]

November 8, 2008

[‡]Robert H. Lurie Comprehensive Cancer Center
Northwestern University, Chicago, IL, 60611, USA

Contents

1 Illumina Identifiers and BeadStudio output files

Illumina uses two types of identifiers: Illumina gene identifiers and Illumina probe identifiers. As their names suggest, Illumina gene identifiers are designed for genes while Illumina probe identifiers are designed for probes. The problem of the gene identifier is that it can correspond to several different probes, which are supposed to match the same gene. In this case, it basically averages the measurements of these probes. This will cause big problem when these probes for the same gene have different measurement values. This happens often in real situations. Because of the binding affinity difference or alternative splicing, the probes corresponding the the sample gene identifier may have quite different expression levels and patterns. If we use the gene identifier to identify the measurements, then we cannot differentiate the difference between these probes.

[*]dupan@northwestern.edu
[†]wakibbe@northwestern.edu
[‡]g-feng@northwestern.edu
[§]jflatow@northwestern.edu
[¶]s-lin2@northwestern.edu

Another problem of using gene identifiers is that the mapping between gene identifiers and probes could be changed with our better understanding of the gene. Therefore, we recommend to use probe identifiers.

Before further discussion, let's describe more details of Illumina BeadStudio output files. BeadStudio usually will export a list of files, which include "Control Gene Profile.txt", "Group Probe Profile.txt", "Samples Table.txt", "Control Probe Profile.txt", "Sample Gene Profile.txt", "Group Gene Profile.txt", "Sample Probe Profile.txt". Among these files, the files with their name including "Probe" use Illumina probe identifiers, which are supposed to be unique for each probe. The files with their names including "Gene" use Illumina gene identifiers. As the probe identifiers were designed for each probe, we recommend to use "Sample Probe Profile.txt" or "Group Probe Profile.txt" for the data analysis.

One problem of Illumina identifiers (both Illumina gene identifiers and Illumina probe identifiers) is that they are not stable and consistent between chip versions and releases. For example, the early version of BeadStudio output files use a numeric number as probe identifier, later on it uses the new version of probe identifiers named as `"ILMN_0000"` ("0000" represents a numeric number). Also, the early version of BeadStudio output files use TargetID as gene identifier, later on gene symbols are directly used as the gene identifiers. The Illumina probe identifiers also change over time. Moreover, the identifiers are not unique. For instance, the same 50mer sequence has two different TargetIDs (early version of gene identifiers) used by Illumina: `"GI_21070949-S"` in the `Mouse_Ref-8_V1` chip and `"scl022190.1_154-S"` in the `Mouse-6_V1` chip. This causes difficulties when combining clinical microarray data collected over time using different versions of the chips. To solve these problems, we designed a nucleotide universal identifier (nuID), which encodes the 50mer oligonucleotide sequence and contains error checking and self-identification code. By using nuID, all the problems mentioned above can be easily solved. For details, please read [1].

```
> library(lumi)

This is mgcv 1.4-1

> library(lumiHumanAll.db)
```

2 nuID (nucleotide universal IDentifier)

Oligonucleotide probes that are sequence identical may have different identifiers between manufacturers and even between different versions of the same company's microarray; and sometimes the same identifier is reused and represents a completely different oligonucleotide, resulting in ambiguity and potentially mis-identification of the genes hybridizing to that probe. This also makes data interpretation and integration of different batches of data difficult.

We have devised a unique, non-degenerate encoding scheme that can be used as a universal representation to identify an oligonucleotide across manufacturers. We have named the encoded representation 'nuID' , for nucleotide universal identifier. Inspired by the fact that the raw sequence of the oligonucleotide is the true definition of identity for a probe, the encoding algorithm uniquely and non-degenerately transforms the sequence itself into a compact identifier (a lossless compression). In addition, we added a redundancy check (checksum) to validate

the integrity of the identifier. These two steps, encoding plus checksum, result in an nuID, which is a unique, non-degenerate, permanent, robust and efficient representation of the probe sequence. For commercial applications that require the sequence identity to be confidential, we have an encryption schema for nuID. We demonstrate the utility of nuIDs for the annotation of Illumina microarrays, and we believe it has universal applicability as a source-independent naming convention for oligomers.

The nuID schema has three significant advantages over using the oligo sequence directly as an identifier: first it is more compact due to the base-64 encoding; second, it has a built-in error detection and self-identification; and third, it can be encrypted in cases where the sequences are preferred not to be disclosed.

2.1 Examples of nuID

```
> ## provide an arbitrary nucleotide sequence as an example
> seq <- 'ACGTAAATTTCAGTTTAAAACCCCCCG'
> ## create a nuID for it
> id <- seq2id(seq)
> print(id)

[1] "YGwPOvwBVW"
```

The original nucleotide sequence can be easily recovered by `id2seq`

```
> id2seq(id)

[1] "ACGTAAATTTCAGTTTAAAACCCCCCG"
```

The nuID is self-identifiable. `is.nuID` can check the sequence is nuID or not. A real nuID

```
> is.nuID(id)

[1] TRUE
```

An random sequence

```
> is.nuID('adfqeqe')

[1] FALSE
```

3 Illumina ID mapping packages

Figure 1 shows the overview of Illumina ID mapping and annotation packages. We will explain each step and provide examples in next sub-sections.

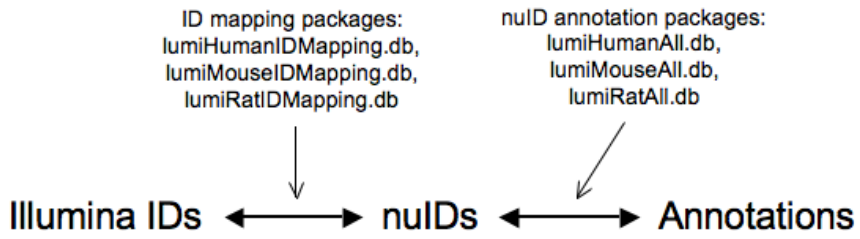

Figure 1: Overview of Illumina ID mapping and annotation packages

3.1 Mapping between Illumina IDs and nuIDs

Because of the unique advantages of nuIDs, converting Illumina IDs as nuIDs will make both probe annotation and maintenance easier. In the old versions of Illumina annotation packages, like *lumiHumanV1*, *lumiHumanV2*, *lumiMouseV1* and *lumiRatV1*, we included separate tables for TargetIDs and ProbeIDs mapping to nuIDs. This becomes difficult when we pool different version of nuIDs together because Illumina IDs may not be consistent across different versions and releases. To partially solve this problem, the lumiR function can automatically produce nuID based on the probe sequence included in the BeadStudio output file. If users have the Illumina manifest file of the chip, they can also use it for nuID conversion. The manifest file (.bgx) basically is a zipped file by gzip. The first step is to unzip the manifest file. The unzipped manifest file is a Tab-separated file, you can open it in Excel. Suppose `'MouseWG-6_V1_1_R2_11234304_A'` is the manifest file and x.lumi is a LumiBatch object indexed by Illumina Probe IDs, the user can use the following code for nuID conversion:

```
x.lumi = addNuID2lumi(x.lumi, annotationFile='MouseWG-6_V1_1_R2_11234304_A')
```

However, in many cases, the BeadStudio output file does not include the probe sequence information and the users do not have the manifest file, either. This makes it necessary to create a separately Illumina ID mapping package, which includes all ID mapping information all versions of Illumina chips. We created three ID mapping packages: *lumiHumanIDMapping*, *lumiMouseIDMapping.db* and *lumiRatIDMapping.db* for human, mouse and rat, respectively.

The purpose of these packages is to provide ID mappings between different types of Illumina identifiers of expression chips and nuIDs, and also mappings from nuIDs to the the most recent RefSeq release. Each library includes the data tables corresponding to all released Illumian expression chips of a particular species before the package releasing date. In each manifest file table, it includes nuIDs and different types of Illumina IDs: "Search_key" ("Search_Key"), "Target" ("ILMN_Gene"), "Accession", "ProbeId" ("Probe_Id"). It also includes a nuID_MappingInfo table, which keeps the mapping information of nuID to RefSeq ID and its related mapping quality information. We will describe this part in more details in next sub-section. By using these ID mapping packages, users can also check the original chip information by providing a list of IDs. There is a function `getChipInfo` in *lumi* package designed for this purpose.

Get information of the ID mapping library:

```
>   if (require(lumiHumanIDMapping))
+           lumiHumanIDMapping()

Database includes ID mapping information of following manifest files:
        HumanHT12_V3_0_R1_11283641_A
        HumanRef8_V1
        HumanRef8_V2_0_R1_11223162_A
        HumanRef8_V2_0_R2_11223162_A
        HumanRef8_V3_0_R0_11282963_A
        HumanWG6_V1
        HumanWG6_V2_0_R1_11223189_A
        HumanWG6_V2_0_R2_11223189_A
        HumanWG6_V2_11223189_B
        HumanWG6_V3_0_R0_11282955_A

It also includes their nuID mapping information.
 For more details, please type lumiHumanIDMapping_nuID().
```

Get chip information of the example data in the *lumi* package.

```
>   ## Load Illumina example data in lumi package
>   data(example.lumi)
>   ## Match the chip information of this example data
>   if (require(lumiHumanIDMapping))
+         getChipInfo(example.lumi, species='Human')

$chipVersion
[1] "HumanRef8_V1"

$species
[1] "Human"

$IDType
[1] "nuID"

$chipProbeNumber
[1] 24357

$inputProbeNumber
[1] 8000

$matchedProbeNumber
[1] 8000
```

3.2 Mapping from nuIDs to RefSeq database

As nuIDs can be directly converted to probe sequences, the latest probe annotations can be easily retrieved. Briefly, we BLASTed each probe sequence (converted from nuID) against the corresponding RefSeq genome. Then we processed the resulting BLAST run files and identified all hits to a probe sequence that have at least a contiguous hit of 17 nucleotides (17 is generally accepted as

a minimum number of contiguous bases required to get a hybridization signal with oligo arrays). We have found that many of the RefSeq models map to the same Entrez gene, so we treat those as single hits and take the best hit defined by expectation value to that model using that probe. We then summarize the total number of Entrez genes hit by a probe, and list the best RefSeq model accession number (if any) and 2nd best RefSeq model accession number (if any). We then score the best hit by giving it a strength, which is the length of the matched sequence plus the length of the longest contiguous sequence in the hit, divided by the length of the probe sequence and then multiply by 50 to convert the strength score to a range from 0 to 100. This procedure is done for the second best gene model hit as well. A uniqueness score is then calculated, and it is simply the strength of the best hit against the first gene model minus the strength of the best hit against the second gene model (in most cases there is not a second model, so this is zero), and this number divided by the strength of the first hit and then multiplied by 100 to convert to a score in the range of 0-100. We anticipate that most groups will be interested in only using probes for which the strength of the best model and the uniqueness score are both 95 or above. For more details, please visit website at: https://prod.bioinformatics.northwestern.edu/nuID/ .

The nuID mapping information is kept in the nuID_MappingInfo table in the ID Mapping library.

The nuID mapping table includes following fields (columns):

1. nuID: nuID for the probe sequence

2. Strength1: Strength of the best hit. This is measured as the longest contig between the probe and the hit sequence plus the number of bases of identity between the two sequences, divided by the total probe length, normalized to 100 for a perfect identical match.

3. Strength2: Strength of the second best gene hit. We are mapping to Entrez gene ids as multiple RefSeq accessions may have the same Entrez gene accession, reflecting differing splice sites, conflicting gene model evidence, or unresolved curation.

4. Uniqueness: (Strength1-Strength2) / Strength1*100.

5. Total hits: Total number of gene models (Entrez gene records) hit by the probe with at least 17 nucleotides

6. Accession: RefSeq gene model Accession number

7. EntrezID: The Entrez Gene ID corresponding to RefSeq Accession number shown in field "Accession"

8. Accession2: RefSeq gene model Accession number for the best hit for the second best gene model (Entrez gene model)

The nuID_MappingInfo table also includes mapping from nuID to EntrezID, which was produced by first mapping nuID to RefSeq and then from RefSeq to EntrezID. Function nuID2RefSeqID and nuID2EntrezID were designed for mapping nuIDs to RefSeq IDs and EntrezIDs, respectively. The functions also include mapping quality filtering. If users want to get all the mapping information, function `getNuIDMappingInfo` was designed for this purpose.

Get nuID mapping information in the ID mapping package:

```
>   if (require(lumiHumanIDMapping))
+         lumiHumanIDMapping_nuID()
```

```
nuID_MappingInfo table includes 97886 unique Homo sapiens Illumina probes mapping informat

The table includes following fields (see help(lumiHumanIDMapping_nuID) for detailed defini
nuID
        Strength1
        Strength2
        Uniqueness
        Total_hits
        Accession
        EntrezID
        Accession2
```

Map nuID to RefSeq ID:

```
>      nuIDs <- featureNames(example.lumi)
>       ## return all mapping information
>     if (require(lumiHumanIDMapping))
+           nuID2RefSeqID(nuIDs[1:10], lib.mapping='lumiHumanIDMapping', filterTh = c(Stre

The provided names of filterTh does not match the field names of nuID_MappingInfo table!
 No filtering will be performed!
oZsQEQXp9ccVIlwoQo 9qedFRd_5Cul.ueZeQ N5YrunuK0q.yIkukis H2UJFJzK0SUTR0SeNE
    "NM_001014811"        "NM_021020"        "NM_000683"        "NM_018146"
oJ1Bzu0pH3qq_Ks_40 B45RF4v0iPUfRHgIjk 6ewSAlc_5fF7uf_6kU ceOREl0EuSVLhF3gJU
                ""        "NM_033208"        "NM_014319"     "NM_001048171"
NJQtGpKWS69_bpev88 Z.VhoQSTotddJdne6k
       "NM_032326"                 ""
```

Map nuID to Entrez Gene ID:

```
>    if (require(lumiHumanIDMapping))
+         nuID2EntrezID(nuIDs[1:10], lib.mapping='lumiHumanIDMapping', filterTh = c(Streng

The provided names of filterTh does not match the field names of nuID_MappingInfo table!
 No filtering will be performed!
oZsQEQXp9ccVIlwoQo 9qedFRd_5Cul.ueZeQ N5YrunuK0q.yIkukis H2UJFJzK0SUTR0SeNE
           "10873"            "11178"             "152"            "55178"
oJ1Bzu0pH3qq_Ks_40 B45RF4v0iPUfRHgIjk 6ewSAlc_5fF7uf_6kU ceOREl0EuSVLhF3gJU
                ""            "91151"            "23592"             "4595"
NJQtGpKWS69_bpev88 Z.VhoQSTotddJdne6k
           "84286"                 ""
```

Return all mapping information related with nuID

```
>    if (require(lumiHumanIDMapping)) {
+         mappingInfo <- nuID2RefSeqID(nuIDs[1:10], lib.mapping='lumiHumanIDMapping', retu
+         head(mappingInfo)
+    }

The provided names of filterTh does not match the field names of nuID_MappingInfo table!
 No filtering will be performed!
                  Strength1 Strength2 Uniqueness Total_hits Accession
```

```
oZsQEQXp9ccVIlwoQo "100"     "0"       "100"      "1"        "NM_001014811"
9qedFRd_5Cul.ueZeQ "100"     "0"       "100"      "1"        "NM_021020"
N5YrunuK0q.yIkukis "100"     "0"       "100"      "1"        "NM_000683"
H2UJFJzK0SUTR0SeNE "100"     "0"       "100"      "1"        "NM_018146"
oJ1Bzu0pH3qq_Ks_40 "0"       "0"       "0"        "0"        ""
B45RF4v0iPUfRHgIjk "100"     "0"       "100"      "1"        "NM_033208"
                   EntrezID Accession2
oZsQEQXp9ccVIlwoQo "10873"  ""
9qedFRd_5Cul.ueZeQ "11178"  ""
N5YrunuK0q.yIkukis "152"    ""
H2UJFJzK0SUTR0SeNE "55178"  ""
oJ1Bzu0pH3qq_Ks_40 ""       ""
B45RF4v0iPUfRHgIjk "91151"  ""
```

4 Illumina microarray annotation packages

As the identifier inconsistency between different versions or even different releases of Illumina chips, it makes create annotation packages in the traditional way difficult. In traditional way, we have to create individual annotation packages for different identifiers and different versions and releases of chips. That will result in lots of annotation packages, and make the maintenance difficult. Users will be hard to decide which package to use. By using the nuID universal identifier, we are able to build one annotation database for different versions and releases of the human (or other species) chips. Moreover, the nuID can be directly converted to the probe sequence, and used to get the most updated refSeq matches and annotations. The recent transition of Bioconductor annotation packages to use SQLite databases made the package size is no longer a concern.

The latest version of Illumina annotation packages indexed by nuID are based on SQLite databases. They were build by using functions in *AnnotaionDbi* package. There are three Bioconductor annotation packages: *lumiHumanAll.db*, *lumiMouseAll.db* and *lumiRatAll.db* for three species Human, Mouse and Rat respectively. These packages include all the previously released Illumina expression chips. The previous versions of packages: *lumiHumanV1*, *lumiHumanV2*, *lumiMouseV1* and *lumiRatV1* will be discontinued. Basically, we converted the probe sequence as nuIDs and pooled them together. Then we mapped nuIDs to different mRNA transcript libraries, which include RefSeq and Unigene. The current mappings (version 1.2.0) were based on on the Illumina Manifest files of the chips. If there are duplicated nuIDs, then the latest version of mapping were used. In our next release, the mapping between nuID to RefSeq IDs will be based on BLASTing results (included in the ID Mapping library). Actually, users can easily build their own annotation package using functions, like `makeHUMANCHIP_DB`, in *AnnotaionDbi* package by providing the mappings from nuID to RefSeq IDs, which can be retrieved in the Illumina ID Mapping libraries described in the previous section. The usage of these annotation libraries is exactly the same as other Bioconductor annotation packages, like Affymetrix.

Here are some examples using the functions implemented in the *annotate* package:

Get gene symbols:

```
>      data(example.lumi)
>      nuIDs <- featureNames(example.lumi)
>      getSYMBOL(nuIDs[1:3], 'lumiHumanAll.db')

oZsQEQXp9ccVIlwoQo 9qedFRd_5Cul.ueZeQ N5YrunuK0q.yIkukis
             "ME3"            "LZTS1"           "ADRA2C"
```

Get Entrez Gene ID:

```
>      getEG(nuIDs[1:3], 'lumiHumanAll.db')

oZsQEQXp9ccVIlwoQo 9qedFRd_5Cul.ueZeQ N5YrunuK0q.yIkukis
           "10873"            "11178"              "152"
```

Get related GO categories:

```
>      goInfo <- getGO(nuIDs[1], 'lumiHumanAll.db')
>      goInfo[[1]][[1]]

$GOID
[1] "GO:0006090"

$Evidence
[1] "IDA"

$Ontology
[1] "BP"
```

A general look up function

```
>      lookUp(nuIDs[1:3], "lumiHumanAll.db", what="SYMBOL")

$oZsQEQXp9ccVIlwoQo
[1] "ME3"

$`9qedFRd_5Cul.ueZeQ`
[1] "LZTS1"

$N5YrunuK0q.yIkukis
[1] "ADRA2C"
```

Check what annotation elements available in the library

```
>      ls('package:lumiHumanAll.db')

 [1] "lumiHumanAll"              "lumiHumanAllACCNUM"
 [3] "lumiHumanAllALIAS2PROBE"   "lumiHumanAllCHR"
 [5] "lumiHumanAllCHRLENGTHS"    "lumiHumanAllCHRLOC"
 [7] "lumiHumanAllCHRLOCEND"     "lumiHumanAll_dbconn"
 [9] "lumiHumanAll_dbfile"       "lumiHumanAll_dbInfo"
[11] "lumiHumanAll_dbschema"     "lumiHumanAllENSEMBL"
[13] "lumiHumanAllENSEMBL2PROBE" "lumiHumanAllENTREZID"
[15] "lumiHumanAllENZYME"        "lumiHumanAllENZYME2PROBE"
```

```
[17] "lumiHumanAllGENENAME"       "lumiHumanAllGO"
[19] "lumiHumanAllGO2ALLPROBES"   "lumiHumanAllGO2PROBE"
[21] "lumiHumanAllMAP"            "lumiHumanAllMAPCOUNTS"
[23] "lumiHumanAllOMIM"           "lumiHumanAllORGANISM"
[25] "lumiHumanAllPATH"           "lumiHumanAllPATH2PROBE"
[27] "lumiHumanAllPFAM"           "lumiHumanAllPMID"
[29] "lumiHumanAllPMID2PROBE"     "lumiHumanAllPROSITE"
[31] "lumiHumanAllREFSEQ"         "lumiHumanAllSYMBOL"
[33] "lumiHumanAllUNIGENE"        "lumiHumanAllUNIPROT"
```

Need to mention, currently there are two sets of Illumina annotation packages in Bioconductor. The Illumina annotation packages mentioned here are named as "lumixxxx", e.g. "lumiHumanAll.db" and are maintained by us. There are another set of packages, named as "illuminaxxxx". These packages are indexed by Illumina IDs. They can also be used together with *lumi* package when the microarray data are indexed by Illumina IDs.

The majority of Bioconductor annotation packages are probe-based annotation package. As a result, these packages are manufacturers and array dependent. These packages include all kinds of mapping from probe to a specific gene annotation. For different packages of the same species, these gene annotations basically are the same. As a result, the majority information of different packages are redundant. Bioconductor also provides Entrez gene based annotation packages. Each species has one Entrez gene based annotation package. For example, *org.Hs.eg.db*, *org.Mm.eg.db* and *org.Rn.eg.db* are designed for Human, Mouse and Rat respectively. Combining these packages with the Illumina ID mapping package, we can also analyze all kinds of chips of the same species.

5 References

Du, P., Kibbe, W.A. and Lin, S.M., "nuID: A universal naming schema of oligonucleotides for Illumina, Affymetrix, and other microarrays", Biology Direct 2007, 2:16 (31May2007).

https://prod.bioinformatics.northwestern.edu/nuID/