Practical uses for CpG probe annotations

Sean Maden

2022-12-15

This vignette provides several practical demonstrations of how to use HM450K and EPIC platform CpG probe annotations, which are provided in the platform-specific manifests. These examples are shown for an example `RGChannelSet` using functions from the `minfi` package.

Probe manifest background

The CpG probe manifest files were provided by Illumina and contain a number of useful fields pretaining to the genome location of a CpG probe. When we read DNAm IDAT files into a `SummarizedExperiment` object using the `minfi` package (e.g. using `read.metharray.exp()` or similar), they are automatically combined with an array platform manifest containing rich probe-level details about the probe chemistry, the genome location, and various functional features and annotations to be aware of.

Getting the HM450K manifest annotations from an `RGChannelSet`

We can easily access the manifest of the older HM450K platform. Using an example `RGChannelSet` object from the `minfiData` package, we can extract and inspect the full HM450K platform manifest with `getAnnotation()` as follows:

```
rg <- get(data("RGsetEx"))
man <- as.data.frame(getAnnotation(rg))
colnames(man)
```

```
##  [1] "chr"                      "pos"
##  [3] "strand"                   "Name"
##  [5] "AddressA"                 "AddressB"
##  [7] "ProbeSeqA"                "ProbeSeqB"
##  [9] "Type"                     "NextBase"
## [11] "Color"                    "Probe_rs"
## [13] "Probe_maf"                "CpG_rs"
## [15] "CpG_maf"                  "SBE_rs"
## [17] "SBE_maf"                  "Islands_Name"
## [19] "Relation_to_Island"       "Forward_Sequence"
## [21] "SourceSeq"                "Random_Loci"
## [23] "Methyl27_Loci"            "UCSC_RefGene_Name"
## [25] "UCSC_RefGene_Accession"   "UCSC_RefGene_Group"
## [27] "Phantom"                  "DMR"
## [29] "Enhancer"                 "HMM_Island"
## [31] "Regulatory_Feature_Name"  "Regulatory_Feature_Group"
## [33] "DHS"
```

Array platform manifests are made available in the Bioconductor packages `IlluminaHumanMethylation450kmanifest` and `IlluminaHumanMethylationEPICmanifest`, so be sure to have these installed before working with DNAm arrays as `SummarizedExperiment` objects. These manifests are unchanged from the original manifest files, which can be accessed from Illumina's website. The HM450K manifest can be found here, and the EPIC manifest can be found here. The official documents also include detailed descriptions of the manifest columns.

Getting the EPIC manifest annotations from an `RGChannelSet`

The newer EPIC platform contains a manifest with updated probe information, including several new annotation columns useful for probe filters.

```
rg.epic <- get(data("RGsetEPIC"))
man.epic <- as.data.frame(getAnnotation(rg.epic))
colnames(man.epic)
```

```
##  [1] "chr"
##  [2] "pos"
##  [3] "strand"
##  [4] "Name"
##  [5] "AddressA"
##  [6] "AddressB"
##  [7] "ProbeSeqA"
##  [8] "ProbeSeqB"
##  [9] "Type"
## [10] "NextBase"
## [11] "Color"
## [12] "Probe_rs"
## [13] "Probe_maf"
## [14] "CpG_rs"
## [15] "CpG_maf"
## [16] "SBE_rs"
## [17] "SBE_maf"
## [18] "Islands_Name"
## [19] "Relation_to_Island"
## [20] "Forward_Sequence"
## [21] "SourceSeq"
## [22] "UCSC_RefGene_Name"
## [23] "UCSC_RefGene_Accession"
## [24] "UCSC_RefGene_Group"
## [25] "Phantom4_Enhancers"
## [26] "Phantom5_Enhancers"
## [27] "DMR"
## [28] "X450k_Enhancer"
## [29] "HMM_Island"
## [30] "Regulatory_Feature_Name"
## [31] "Regulatory_Feature_Group"
## [32] "GencodeBasicV12_NAME"
## [33] "GencodeBasicV12_Accession"
## [34] "GencodeBasicV12_Group"
## [35] "GencodeCompV12_NAME"
## [36] "GencodeCompV12_Accession"
## [37] "GencodeCompV12_Group"
```

```
## [38] "DNase_Hypersensitivity_NAME"
## [39] "DNase_Hypersensitivity_Evidence_Count"
## [40] "OpenChromatin_NAME"
## [41] "OpenChromatin_Evidence_Count"
## [42] "TFBS_NAME"
## [43] "TFBS_Evidence_Count"
## [44] "Methyl27_Loci"
## [45] "Methyl450_Loci"
## [46] "Random_Loci"
```

Annotation-based probe filters

There are many reasons we may want to filter or remove a CpG probe prior to performing downstream analyses such as differential methylation tests or epigenome-wide association tests. For instance, it is common to filter CpG probes which are likely to be impacted by underlying population genetic variation, or probes which have consistent poor quality. Two common reasons to filter probes are , or based on studies showing extensive cross-reactivity that limits a probe's reliability. We can see how to perform these filters below.

Filtering on common SNPs

We can filter a probe based on high likelihood of a confounding single nucleotide polymorphism (SNP), or a variant that occurs near a probe (e.g. within the 50bp annealing sequence), or if it overlaps the exact CpG location. These probe-proximal SNPs can be identified either from the manifest (see above), or accessed independently using `getSnpInfo()`. For instance:

```
getSnpInfo(rg)
```

```
## DataFrame with 485512 rows and 6 columns
##                    Probe_rs Probe_maf      CpG_rs   CpG_maf      SBE_rs
##                 <character> <numeric> <character> <numeric> <character>
## cg00050873               NA        NA          NA        NA          NA
## cg00212031               NA        NA          NA        NA          NA
## cg00213748               NA        NA          NA        NA          NA
## cg00214611               NA        NA          NA        NA          NA
## cg00455876               NA        NA          NA        NA          NA
## ...                     ...       ...         ...       ...         ...
## ch.22.909671F            NA        NA          NA        NA          NA
## ch.22.46830341F          NA        NA          NA        NA          NA
## ch.22.1008279F           NA        NA          NA        NA          NA
## ch.22.47579720R  rs79009754  0.012587          NA        NA          NA
## ch.22.48274842R  rs79812973  0.022589          NA        NA          NA
##                   SBE_maf
##                 <numeric>
## cg00050873             NA
## cg00212031             NA
## cg00213748             NA
## cg00214611             NA
## cg00455876             NA
## ...                   ...
## ch.22.909671F          NA
## ch.22.46830341F        NA
```

```
## ch.22.1008279F         NA
## ch.22.47579720R        NA
## ch.22.48274842R        NA
```

It is recommended that CpG probes which contain a common SNP be filtered prior to analysis. Common SNPs are identified based on their minor allele frequency (MAF, e.g. from columns `Probe_maf`, `CpG_maf`, or `SBE_maf`).

The function `dropLociWithSnps()` allows one to filter a mapped `MethylSet` or `RatioSet` based on the SNP MAF frequency. For example:

```
gm <- preprocessRaw(rg) # make MethylSet
gms <- mapToGenome(gm) # make GenomicMethylSet
gmsf <- dropLociWithSnps(gms) # filter probes with SNPs
```

The minimum MAF frequency filter can be changed in `dropLociWithSnps()` by setting the `maf` argument. See `?dropLociWithSnps` for details. Also note the SNP information is a product of its reference, and details about the specific genetic variant databases mined for this SNP info can be found in the manifest documentation (see above).

Filtering on cross-reactive CpG probes

A number of probes on both the EPIC and HM450K platforms were previously found to possess high cross-reactivity, an undesirable quality that limits probe measurement reliability. We can identify and remove probes with evidence of cross-reactivity from either platform by simply filtering on the identifiers of the probes to be removed. Several commonly utilized sources of CpG IDs for cross-reactive probes include:

- Chen et al 2013 cross-reactive HM450K probes (source). Note, these probes have been uploaded here.
- Pidsley et al 2016 cross-reactive EPIC probes (source). Note, these probes have also been uploaded here.

We can filter out lists of probe CpG IDs (e.g. `cg00050873`) by subsetting on the `Name` column in the manifest.

Querying and quantifying genome regions and functional groups

Many DNAm studies note whether methylation occurs in a gene's promoter or body region. One reason is that we expect the functional consequences of DNAm to vary given its location in a gene, or indeed if it occurs in a gene at all. Another common way to annotate CpG probes is according to their position relative to protein-coding gene regions as well as relative to cytosine- and guanine-dense regions called CpG Islands. We can quickly group probes according to either their gene or CpG Island region using information in the manifest, as shown below.

Find probes mapping to a gene

The gene IDs can be found under the columns `UCSC_RefGene_Name` (for the common gene name) and `UCSC_RefGene_Accession` (for the official RefGene accession). Note that entries in these columns correspond to transcript labels, so gene names and IDs may be repeated multiple times separated by a semi-colon character. This formatting means we need to use regular expression to isolate all the probes by a given gene name.

We can quantify the number of intergenic (non-gene-mapping) and genic (gene-mapping) probes with:

```
as.data.frame(table(man$UCSC_RefGene_Name==""))
```

```
##    Var1   Freq
## 1 FALSE 365860
## 2  TRUE 119652
```

We can further grab all probes mapping to the gene `ARID1A` using:

```
gene.str <- "ARID1A" # common gene name
gene.patt <- paste0("(^|;)", gene.str, "($|;)") # regex pattern
which.gene <- which(grepl(gene.patt, man$UCSC_RefGene_Name)) # gene filter
manf <- man[which.gene,]
dim(manf)
```

```
## [1] 29 33
```

Viewing gene functional region frequencies

We can find the gene relation annotations under the column `UCSC_RefGene_Group` in the manifest. Values under this column are either blank, if a probe does not map to a gene (a.k.a. "intergenic probes"), or contains one or more location annotations corresponding to various overlapping transcript IDs.

To get all mapped gene regions as a vector, use:

```
gene.regionv <- unlist(strsplit(man$UCSC_RefGene_Group, ";"))
```

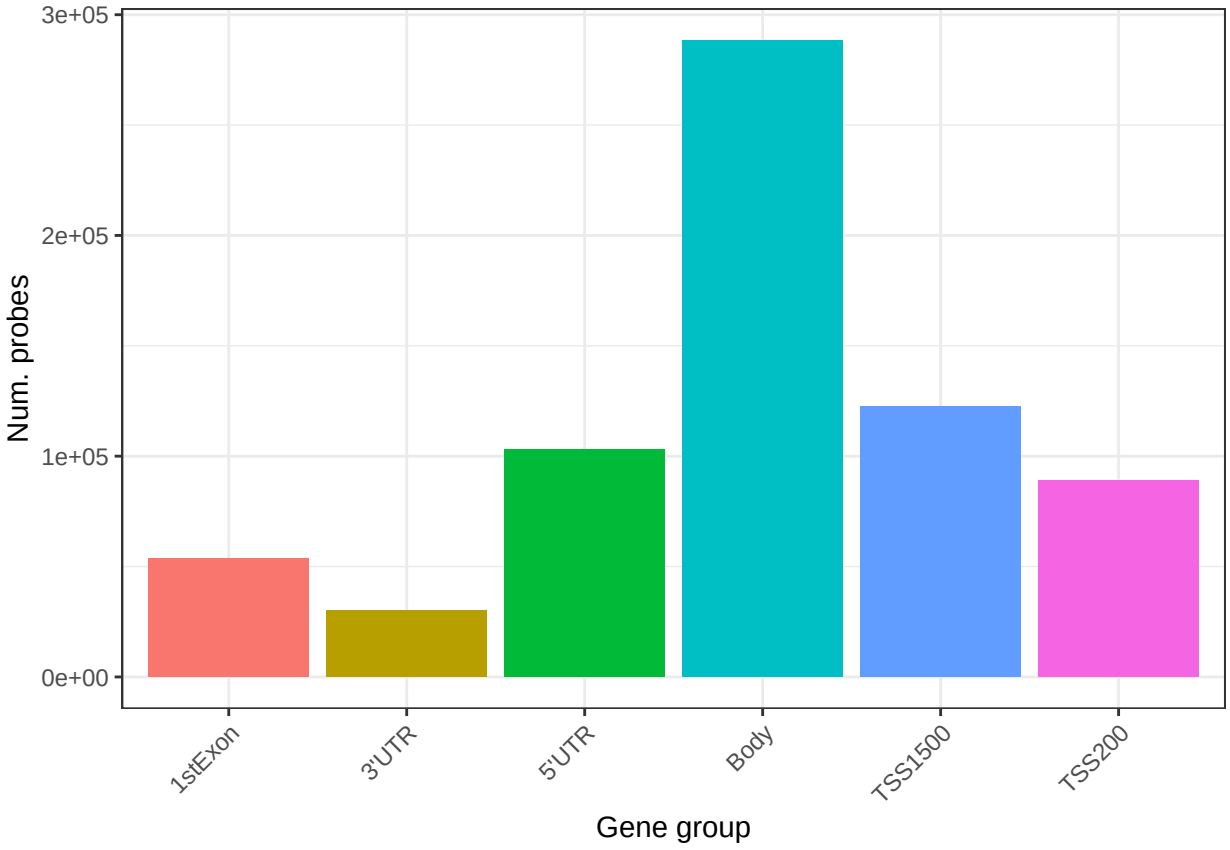
To view all possible group values, we can use:

```
unique(gene.regionv)
```

```
## [1] "Body"    "TSS1500" "TSS200"  "1stExon" "5'UTR"   "3'UTR"
```

To view the frequency with which each region is mapped by an array probe, use:

```
df <- as.data.frame(table(gene.regionv)) # get group frequencies
ggplot(df, aes(x = gene.regionv, y = Freq, fill = gene.regionv)) + theme_bw() +
  geom_bar( stat = "identity") + xlab("Gene group") + ylab("Num. probes") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1))
```



Identifying promoter-mapping and gene body-mapping CpG probes

We can use this logic of the `UCSC_RefGene_Group` variable to build conditions for new functional regions of interest, namely promoter and gene body regions.

For promoters, we can use a common definition of a probe that maps to either the `5'UTR` or either of the downstream (1500bp, `TSS1500`) or upstream (500bp, `TSS500`) transcription start sites. We can make the new promoter variable using:

```
# make pattern: "(^|;)5'UTR($|;)|(^|;)TSS1500($|;)|(^|;)TSS500($|;)"
prom.catv <- c("5'UTR", "TSS1500", "TSS500")
prom.patt <- paste0("(^|;)", prom.catv, "($|;)", collapse = "|")
# use regex to detect promoter-mapping probes
man$promoter <- ifelse(grepl(prom.patt, man$UCSC_RefGene_Group),
                       TRUE, FALSE)
as.data.frame(table(man$promoter))
```

```
##    Var1   Freq
## 1 FALSE 345715
## 2  TRUE 139797
```

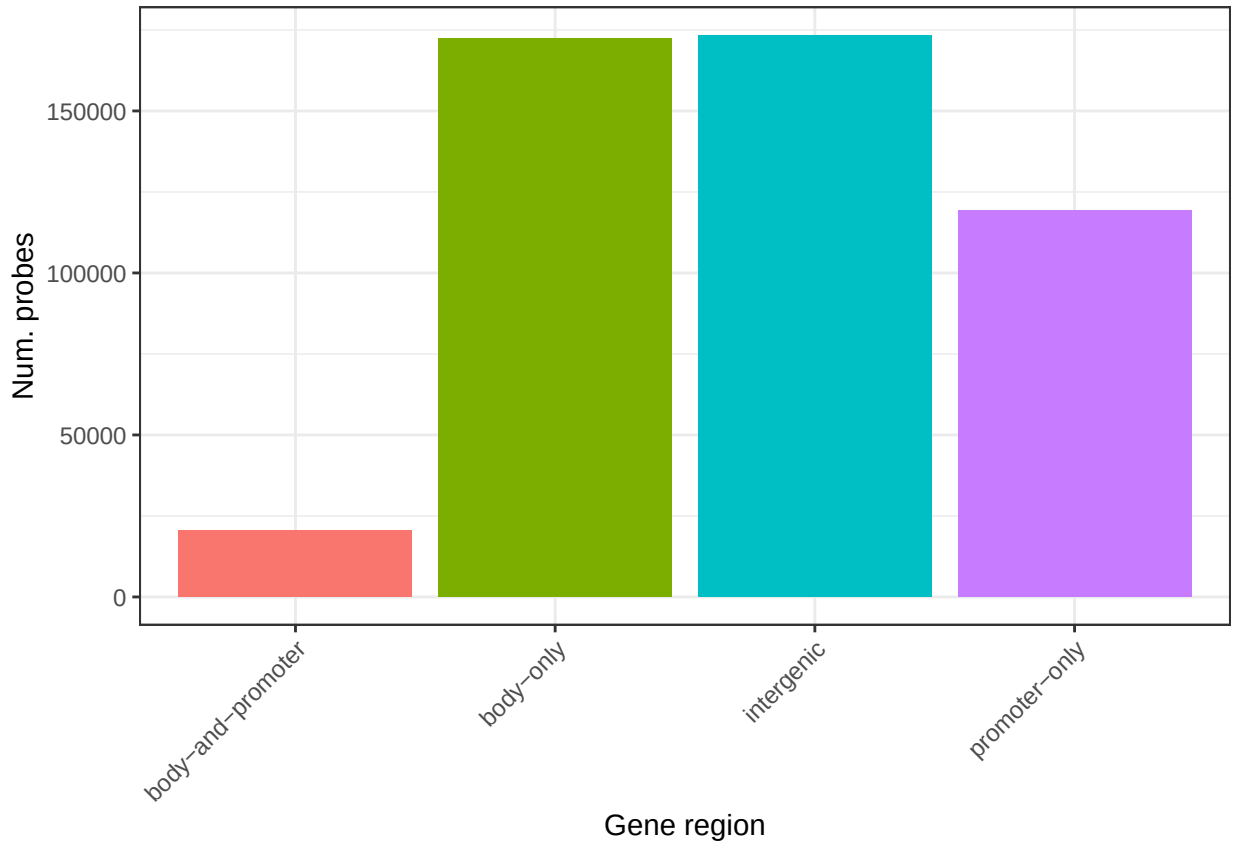
One way we can define the gene body is as all gene-mapping probes which don't map to the gene promoter. Using a similar strategy as for the new `promoter` variable, we can define a new `gene_body` variable with:

```
# make pattern: "(^|;)Body($|;)|(^|;)3'UTR($|;)"
body.catv <- c("Body", "3'UTR")
body.patt <- paste0("(^|;)", body.catv, "($|;)", collapse = "|")
# use regex to detect body-mapping probes
man$gene_body <- ifelse(grepl(body.patt, man$UCSC_RefGene_Group),
                        TRUE, FALSE)
table(man$gene_body)
```

```
##
##  FALSE   TRUE
## 292622 192890
```

Since the manifest is transcript-centric, you will find that some probes map to the promoter of one transcript and the body of another transcript. You can view these as follows:

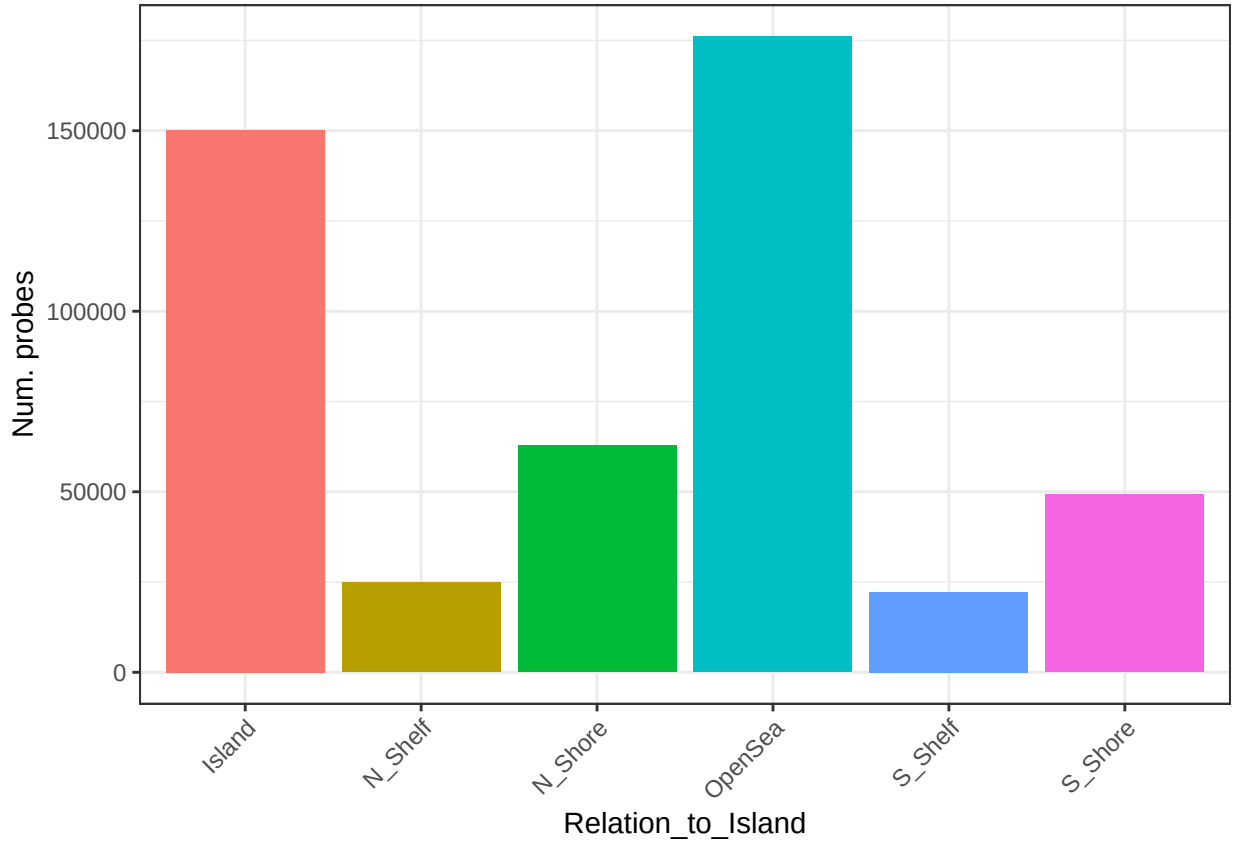
```
dfp <- as.data.frame(table(man$gene_body, man$promoter))
dfp$region <- c("intergenic", "body-only", "promoter-only", "body-and-promoter")
ggplot(dfp, aes(x = region, y = Freq, fill = region)) + theme_bw() +
  geom_bar( stat = "identity") + xlab("Gene region") + ylab("Num. probes") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1))
```



CpG Island annotations

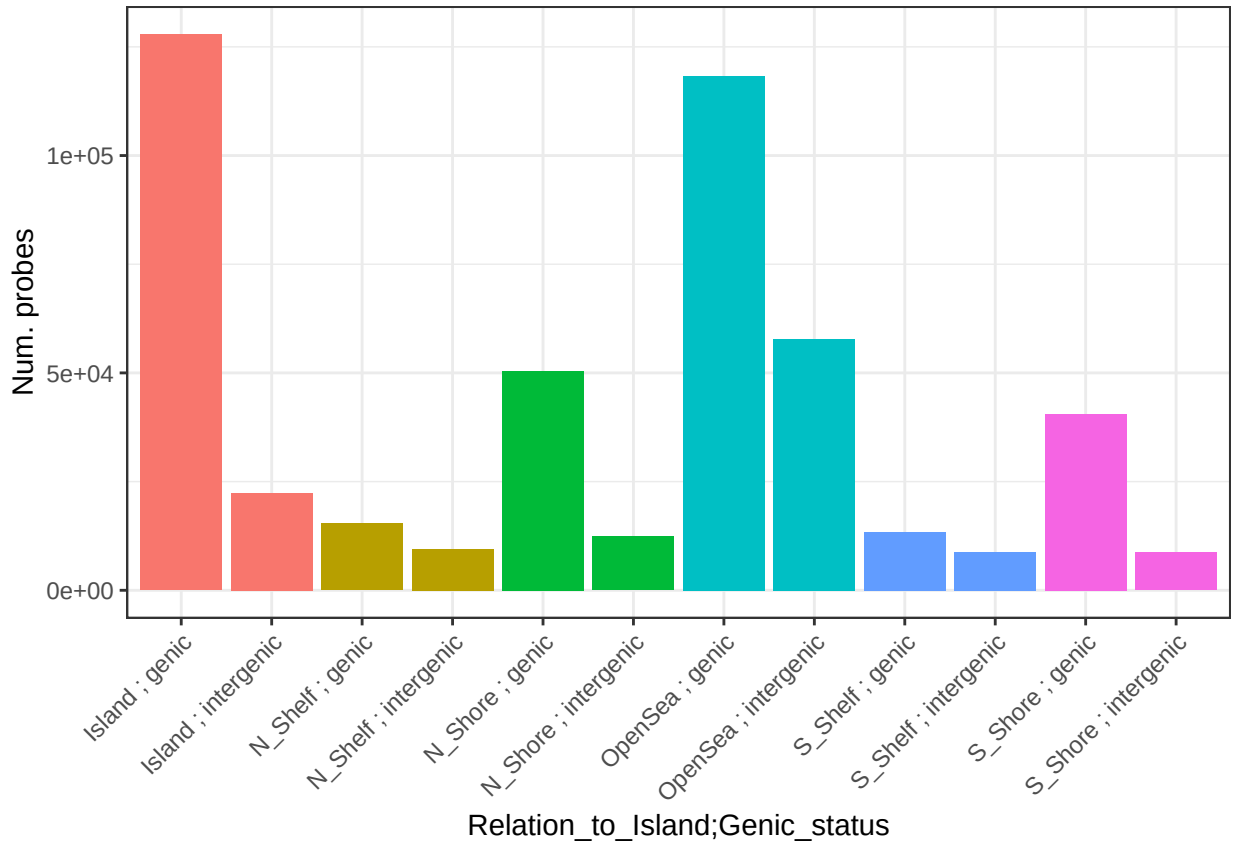
It is comparatively straightforward to view a probe's relation to a CpG Island from the manifest using the variable `Relation_to_Island`.

```
dfp <- as.data.frame(table(man$Relation_to_Island))
ggplot(dfp, aes(x = Var1, y = Freq, fill = Var1)) + theme_bw() +
  geom_bar(stat = "identity") + xlab("Relation_to_Island") + ylab("Num. probes") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1))
```



It is also easy to add the "genic" information, or whether a probe maps to a gene, alongside the CpG Island annotations, as follows:

```
dfp <- as.data.frame(table(man$Relation_to_Island, man$UCSC_RefGene_Name==""))
dfp$region <- paste0(dfp$Var1," ; ", c(rep("genic", 6), rep("intergenic", 6)))
ggplot(dfp, aes(x = region, y = Freq, fill = Var1)) + theme_bw() +
  geom_bar(stat = "identity") + xlab("Relation_to_Island;Genic_status") +
  ylab("Num. probes") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1))
```



Conclusions

This vignette described practical uses for the platform-specific, manifest-based annotations for CpG probes on HM450K and EPIC platforms. Additional resources, information, and guidance for analyzing compilations of DNAm array datasets can be found in the following vignettes:

- [`recountmethylation`] R/Bioconductor package documentation
- `minfi` R/Bioconductor package and documentation