SomaticCancerAlterations

Julian Gehring, EMBL Heidelberg

October 18, 2014

Contents

1 Motivation 1

2 Data Sets 1

3 Exploring Mutational Data 2

4 Exploring Multiple Studies 4

5 Data Provenance 6
- 5.1 TCGA Data . . . 6
  - 5.1.1 Processing . . . 7
  - 5.1.2 Selection Criteria of Data Sets . . . 7
  - 5.1.3 Consistency Check . . . 7

6 Alternatives 7

7 Session Info 7

1 Motivation

Over the last years, large efforts have been taken to characterize the somatic landscape of cancers. Many of the conducted studies make their results publicly available, providing a valuable resource for investigating beyond the level of individual cohorts. The *SomaticCancerAlterations* package collects mutational data of several tumor types, currently focusing on the TCGA calls sets, and aims for a tight integration with *R* and *Bioconductor* workflows. In the following, we will illustrate how to access this data and give examples for use cases.

2 Data Sets

The Cancer Genome Atlas (TCGA)[1] is a consortium effort to analyze a variety of tumor types, including gene expression, methylation, copy number changes, and somatic mutations[2]. With the *SomaticCancerAlterations* package, we provide the callsets of somatic mutations for all publically available TCGA studies. Over time, more studies will be added, as they become available and unrestriced in their usage.

To get started, we get a list of all available data sets and access the metadata associated with each study.

[1] http://cancergenome.nih.gov
[2] https://wiki.nci.nih.gov/display/TCGA/TCGA+Home

```
all_datasets = scaListDatasets()
print(all_datasets)
```

```
## [1] "gbm_tcga"  "hnsc_tcga" "kirc_tcga" "luad_tcga" "lusc_tcga" "ov_tcga"   "skcm_tcga"
## [8] "thca_tcga"
```

```
meta_data = scaMetadata()
print(meta_data)
```

```
##           Cancer_Type        Center NCBI_Build Sequence_Source Sequencing_Phase
## gbm_tcga          GBM broad.mit.edu         37             WXS          Phase_I
## hnsc_tcga        HNSC broad.mit.edu         37         Capture          Phase_I
## kirc_tcga        KIRC broad.mit.edu         37         Capture          Phase_I
## luad_tcga        LUAD broad.mit.edu         37             WXS          Phase_I
## lusc_tcga        LUSC broad.mit.edu         37             WXS          Phase_I
## ov_tcga            OV broad.mit.edu         37             WXS          Phase_I
## skcm_tcga        SKCM broad.mit.edu         37         Capture          Phase_I
## thca_tcga        THCA broad.mit.edu         37             WXS          Phase_I
##                Sequencer Number_Samples Number_Patients
## gbm_tcga  Illumina GAIIx            291             291
## hnsc_tcga Illumina GAIIx            319             319
## kirc_tcga Illumina GAIIx            297             293
## luad_tcga Illumina GAIIx            538             519
## lusc_tcga Illumina GAIIx            178             178
## ov_tcga   Illumina GAIIx            142             142
## skcm_tcga Illumina GAIIx            266             264
## thca_tcga Illumina GAIIx            406             403
##                                    Cancer_Name
## gbm_tcga                Glioblastoma multiforme
## hnsc_tcga Head and Neck squamous cell carcinoma
## kirc_tcga                    Kidney Chromophobe
## luad_tcga                   Lung adenocarcinoma
## lusc_tcga          Lung squamous cell carcinoma
## ov_tcga       Ovarian serous cystadenocarcinoma
## skcm_tcga                Skin Cutaneous Melanoma
## thca_tcga                    Thyroid carcinoma
```

Next, we load a single dataset with the scaLoadDataset function.

```
ov = scaLoadDatasets("ov_tcga", merge = TRUE)
```

3 Exploring Mutational Data

The somatic variants of each study are represented as a object, ordered by genomic positions. Additional columns describe properties of the variant and relate it the the affected gene, sample, and patient.

```
head(ov, 3)
```

```
## GRanges object with 3 ranges and 14 metadata columns:
##           seqnames             ranges strand | Hugo_Symbol Entrez_Gene_Id
##              <Rle>          <IRanges>  <Rle> |    <factor>      <integer>
##   ov_tcga        1 [1334552, 1334552]      * |       CCNL2          81669
##   ov_tcga        1 [1961652, 1961652]      * |       GABRD           2563
##   ov_tcga        1 [2420688, 2420688]      * |       PLCH2           9651
##           Variant_Classification Variant_Type Reference_Allele Tumor_Seq_Allele1
```

```
##                         <factor>     <factor>         <factor>          <factor>
##   ov_tcga                 Silent          SNP                C                 C
##   ov_tcga                 Silent          SNP                C                 C
##   ov_tcga      Missense_Mutation          SNP                C                 C
##           Tumor_Seq_Allele2 Verification_Status Validation_Status Mutation_Status
##                     <factor>            <factor>          <factor>        <factor>
##   ov_tcga                 T             Unknown             Valid         Somatic
##   ov_tcga                 T             Unknown             Valid         Somatic
##   ov_tcga                 G             Unknown             Valid         Somatic
##             Patient_ID                   Sample_ID     index  Dataset
##               <factor>                    <factor> <integer> <factor>
##   ov_tcga TCGA-24-2262 TCGA-24-2262-01A-01W-0799-08      3901  ov_tcga
##   ov_tcga TCGA-24-1552 TCGA-24-1552-01A-01W-0551-08      3414  ov_tcga
##   ov_tcga TCGA-13-1484 TCGA-13-1484-01A-01W-0545-08      1567  ov_tcga
##   -------
##   seqinfo: 86 sequences from an unspecified genome
```

```
with(mcols(ov), table(Variant_Classification, Variant_Type))
```

```
##                         Variant_Type
## Variant_Classification    DEL  INS  SNP
##   3'UTR                     0    0    3
##   5'Flank                   0    0    1
##   5'UTR                     0    0    1
##   Frame_Shift_Del          79    0    0
##   Frame_Shift_Ins           0   16    0
##   IGR                       0    0    5
##   In_Frame_Del             26    0    0
##   In_Frame_Ins              0    1    0
##   Intron                    0    0   34
##   Missense_Mutation         0    0 4299
##   Nonsense_Mutation         0    0  285
##   Nonstop_Mutation          0    0    6
##   RNA                       0    0    1
##   Silent                    0    0 1417
##   Splice_Site               9    2  121
##   Translation_Start_Site    1    0    1
```

With such data at hand, we can identify the samples and genes haboring the most mutations.

```
head(sort(table(ov$Sample_ID), decreasing = TRUE))
```

```
##
## TCGA-09-2049-01D-01W-0799-08 TCGA-13-0923-01A-01W-0420-08 TCGA-09-2050-01A-01W-0799-08
##                          119                          118                          111
## TCGA-25-1326-01A-01W-0492-08 TCGA-25-1313-01A-01W-0492-08 TCGA-23-1110-01A-01D-0428-08
##                          110                          104                          102
```

```
head(sort(table(ov$Hugo_Symbol), decreasing = TRUE), 10)
```

```
##
##    TP53     TTN PCDHAC2   MUC16   MUC17 PCDHGC5   USH2A   CSMD3 CD163L1 DYNC1H1
##     118      30      14      12       9       9       9       8       7       7
```

4 Exploring Multiple Studies

Instead of focusing on an individual study, we can also import several at once. The results are stored as a *GRangesList* in which each element corresponds to a single study. This can be merged into a single *GRanges* object with `merge = TRUE`.

```
three_studies = scaLoadDatasets(all_datasets[1:3])

print(elementLengths(three_studies))
```

```
##  gbm_tcga hnsc_tcga kirc_tcga
##     22166     73766     26265
```

```
class(three_studies)
```

```
## [1] "SimpleGenomicRangesList"
## attr(,"package")
## [1] "GenomicRanges"
```

```
merged_studies = scaLoadDatasets(all_datasets[1:3], merge = TRUE)

class(merged_studies)
```

```
## [1] "GRanges"
## attr(,"package")
## [1] "GenomicRanges"
```

We then compute the number of mutations per gene and study:

```
gene_study_count = with(mcols(merged_studies), table(Hugo_Symbol, Dataset))

gene_study_count = gene_study_count[order(apply(gene_study_count, 1, sum), decreasing = TRUE), ]

gene_study_count = addmargins(gene_study_count)

head(gene_study_count)
```

```
##            Dataset
## Hugo_Symbol gbm_tcga hnsc_tcga kirc_tcga  Sum
##     Unknown       29       899       630 1558
##     TTN          121       401       125  647
##     TP53         101       323         8  432
##     MUC16         68       155        46  269
##     ADAM6          0       173        63  236
##     MUC4          17        32       130  179
```

Further, we can subset the data by regions of interests, and compute descriptive statistics only on the subset.

```
tp53_region = GRanges("17", IRanges(7571720, 7590863))

tp53_studies = subsetByOverlaps(merged_studies, tp53_region)
```

For example, we can investigate which type of somatic variants can be found in TP53 throughout the studies.

```
addmargins(table(tp53_studies$Variant_Classification, tp53_studies$Dataset))
```

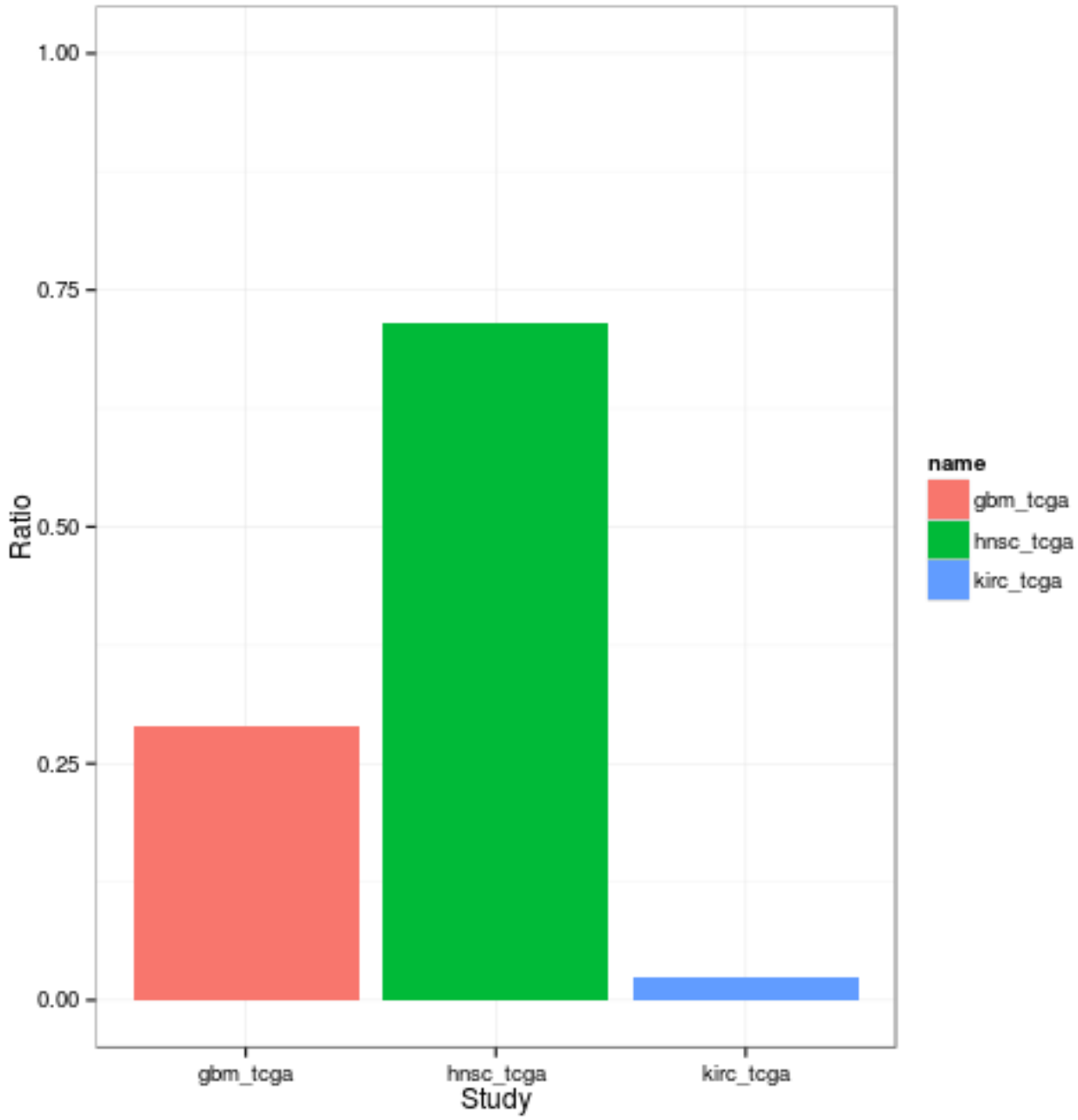
```
##
##                         gbm_tcga hnsc_tcga kirc_tcga Sum
##   Frame_Shift_Del              6        41         0  47
##   Frame_Shift_Ins              1        11         0  12
```

```
##   In_Frame_Del                  2         7         0   9
##   In_Frame_Ins                  0         2         0   2
##   Missense_Mutation            81       183         6 270
##   Nonsense_Mutation             4        54         0  58
##   Nonstop_Mutation              0         0         0   0
##   Silent                        1         6         1   8
##   Splice_Site                   6        19         1  26
##   Translation_Start_Site        0         0         0   0
##   RNA                           0         0         0   0
##   Sum                         101       323         8 432
```

To go further, how many patients have mutations in TP53 for each cancer type?

```
fraction_mutated_region = function(y, region) {
    s = subsetByOverlaps(y, region)
    m = length(unique(s$Patient_ID)) / metadata(s)$Number_Patients
    return(m)
}

mutated_fraction = sapply(three_studies, fraction_mutated_region, tp53_region)

mutated_fraction = data.frame(name = names(three_studies), fraction =
mutated_fraction)
```

```
library(ggplot2)

p = ggplot(mutated_fraction) + ggplot2::geom_bar(aes(x = name, y = fraction,
fill = name), stat = "identity") + ylim(0, 1) + xlab("Study") + ylab("Ratio") +
theme_bw()

print(p)
```



5 Data Provenance

5.1 TCGA Data

When importing the mutation data from the TCGA servers, we checked the data for consistency and fix common ambiguities in the annotation.

5.1.1 Processing

1. Selection of the most recent somatic variant calls for each study. These were stored as `*.maf` files in the TCGA data directory[3]. If both manually curated and automatically generated variant calls were available, the curated version was chosen.
2. Importing of the `*.maf` files into *R* and checking for consistency with the TCGA MAF specifications[4]. Please note that these guidelines are currently only suggestions and most TCGA files violate some of these.
3. Transformation of the imported variants into a `GRanges` object, with one row for each reported variant. Only columns related to the genomic origin of the somatic variant were stored, additional columns describing higher-level effects, such as mutational consequences and alterations at the protein level, were dropped. The `seqlevels` information defining the chromosomal ranges were taken from the 1000genomes phase 2 reference assembly[5].
4. The patient barcode was extracted from the sample barcode.
5. Metadata describing the design and analysis of the study was extracted.
6. The processed variants were written to disk, with one file for each study. The metadata for all studies were stored as a single, separate object.

5.1.2 Selection Criteria of Data Sets

We included data sets in the package that were

- conducted by the Broad Institute.
- cleared for unrestricted access and usage[6].
- sequenced with Illumina platforms.

5.1.3 Consistency Check

According to the TCGA specifications for the `MAF` files, we screened and corrected for common artifacts in the data regarding annotation. This included:

- Transfering of all genomic coordinates to the NCBI 37 reference notation (with the chromosome always depicted as 'MT')
- Checking of the entries against all allowed values for this field (currently for the columns `Hugo_Symbol`, `Chromosome`, `Strand`, `Variant_Classification`, `Variant_Type`, `Reference_Allele`, `Tumor_Seq_Allele1`, `Tumor_Seq_Allele2`, `Verification_Status`, `Validation_Status`, `Sequencer`).

6 Alternatives

The TCGA data sets can be accessed in different ways. First, the TCGA itself offers access to certain types of its collected data[7]. Another approach has been taken by the cBioPortal for Cancer Genomics[8] which has performed high-level analyses of several TCGA data sources, such as gene expression and copy number changes. This summarized data can be queried through an *R* interface[9].

7 Session Info

[3] https://tcga-data.nci.nih.gov/tcgafiles/ftp_auth/distro_ftpusers/anonymous/tumor/
[4] https://wiki.nci.nih.gov/display/TCGA/Mutation+Annotation+Format+(MAF)+Specification
[5] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/phase2_reference_assembly_sequence/
[6] http://cancergenome.nih.gov/abouttcga/policies/publicationguidelines
[7] https://tcga-data.nci.nih.gov/tcga/tcgaDownload.jsp
[8] http://www.cbioportal.org/public-portal
[9] http://www.cbioportal.org/public-portal/cgds_r.jsp

```
## R version 3.1.1 Patched (2014-09-25 r66681)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods
## [9] base
##
## other attached packages:
## [1] ggbio_1.14.0                  ggplot2_1.0.0
## [3] GenomicRanges_1.18.1          GenomeInfoDb_1.2.0
## [5] IRanges_2.0.0                 S4Vectors_0.4.0
## [7] BiocGenerics_0.12.0           SomaticCancerAlterations_1.1.1
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.28.0     BBmisc_1.7               BSgenome_1.34.0
##  [4] BatchJobs_1.4            Biobase_2.26.0           BiocParallel_1.0.0
##  [7] BiocStyle_1.4.1          Biostrings_2.34.0        DBI_0.3.1
## [10] Formula_1.1-2            GGally_0.4.8             GenomicAlignments_1.2.0
## [13] GenomicFeatures_1.18.0   Hmisc_3.14-5             MASS_7.3-35
## [16] OrganismDbi_1.8.0        RBGL_1.42.0              RColorBrewer_1.0-5
## [19] RCurl_1.95-4.3           RSQLite_0.11.4           Rcpp_0.11.3
## [22] Rsamtools_1.18.0         VariantAnnotation_1.12.0 XML_3.98-1.1
## [25] XVector_0.6.0            acepack_1.3-3.3          base64enc_0.1-2
## [28] biomaRt_2.22.0           biovizBase_1.14.0        bitops_1.0-6
## [31] brew_1.0-6               checkmate_1.4            cluster_1.15.3
## [34] codetools_0.2-9          colorspace_1.2-4         dichromat_2.0-0
## [37] digest_0.6.4             evaluate_0.5.5           exomeCopy_1.12.0
## [40] fail_1.2                 foreach_1.4.2            foreign_0.8-61
## [43] formatR_1.0              graph_1.44.0             grid_3.1.1
## [46] gridExtra_0.9.1          gtable_0.1.2             highr_0.3
## [49] iterators_1.0.7          knitr_1.7                labeling_0.3
## [52] lattice_0.20-29          latticeExtra_0.6-26      munsell_0.4.2
## [55] nnet_7.3-8               plyr_1.8.1               proto_0.3-10
## [58] reshape_0.8.5            reshape2_1.4             rpart_4.1-8
## [61] rtracklayer_1.26.1       scales_0.2.4             sendmailR_1.2-1
## [64] splines_3.1.1            stringr_0.6.2            survival_2.37-7
## [67] tools_3.1.1              zlibbioc_1.12.0
```