Overview
1. The NCI Genomic Data Commons
2. The GenomicDataCommons R package
3. Querying the GDC from R
4. Variant analysis with R
Summary

BIMM-143, Lecture 18

Investigating cancer genomics datasets

BIMM-143 Lecture 18:
Barry Grant < http://thegrantlab.org/bimm143/ (http://thegrantlab.org/bimm143/) >
2019-03-06 (21:27:43 PST on Wed, Mar 06)

Overview

Cancer is fundamentally a disease of the genome, caused by changes in the DNA, RNA, and proteins of a cell that push cell growth into overdrive. Identifying the genomic alterations that arise in a given cancer can help researchers decode how a particular cancer develops and improve upon the diagnosis and treatment of cancers based on their distinct molecular abnormalities.

With the ability to sequence whole genomes and exomes, attention has turned to trying to understand the full spectrum of genetic mutations that underlie cancer.

The genomes or exomes of tens of thousands of cancers have now been sequenced. Analyzing this data can yield important new insights into cancer biology. This is important because it is estimated that cancer will strike over 40% of people at some point in their lifetime with frequently devastating effects.

1. The NCI Genomic Data Commons

The National Cancer Institute (NCI) in the US has established the **Genomic Data Commons** (https://gdc.cancer.gov/about-gdc) (or **GDC** for short) for sharing cancer genomics data-sets.

This includes data from a range of large scale projects such as **The Cancer Genome Atlas** (TCGA) and other projects. The TGCA project aims to generate comprehensive, multi-dimensional maps of the key genomic changes in major types and sub-types of cancer. As of writing, TCGA has analyzed matched tumor and normal tissues from over 11,000 patients covering 33 cancer types and sub-types.

You can get a feel for the types of cancer data contained in the NCI-GDC by visiting their new web portal: https://portal.gdc.cancer.gov (https://portal.gdc.cancer.gov).

Exploring the GDC online

Visit the NCI-GDC web portal and enter p53 into the search box.

Q1. How many *Cases* (i.e. patient samples) have been found to have p53 mutations?

Q2. What are the top 6 misssense mutations found in this gene?
HINT: Scroll down to the 'TP53 - Protein' section and mouse over the displayed plot. For example **R175H** is found in 156 cases.

Q3. Which domain of the protein (as annotated by PFAM) do these mutations reside in?

Q4. What are the top 6 *primary sites* (i.e. cancer locations such as Lung, Brain, etc.) with p53 mutations and how many *primary sites* have p53 mutations been found in?
HINT: Clicking on the number links in the *Cancer Distribution* section will take you to a summary of available data accross *cases*, *genes*, and *mutations* for p53. Looking at the *cases* data will give you a ranked listing of *primary sites*.

Return to the NCI-GDC homepage and using a similar search and explore strategy answer the following questions:

Q5. What is the most frequentely mutated position associated with cancer in the **KRas** protein (i.e. the amino acid with the most mutations)?

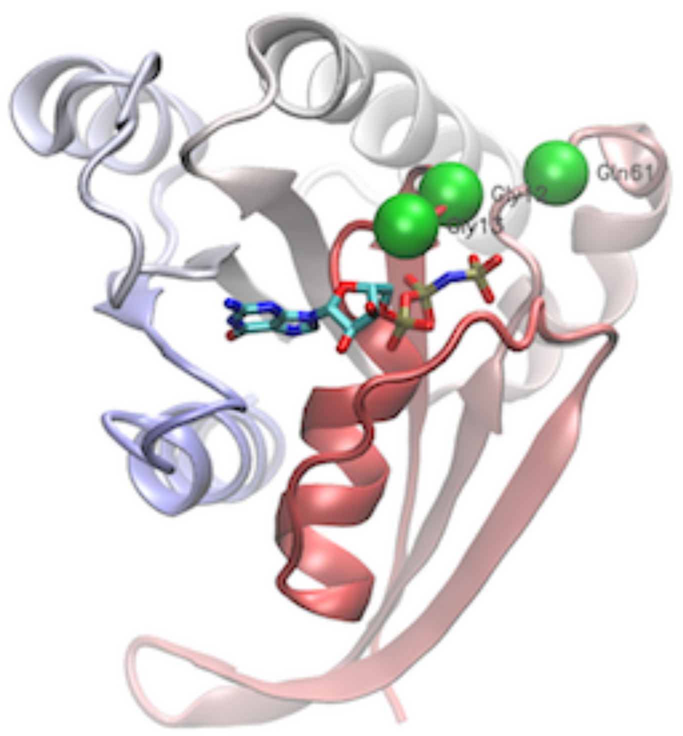
Q6. Are KRas mutations common in Pancreatic Adenocarcinoma (i.e. is the Pancreas a common '*primary site*' for KRas mutations?).

Q6. What is the '*TGCA project*' with the most KRas mutations?

Q7. What precent of cases for this '*TGCA project*' have KRas mutations and what precent of cases have p53 mutations?
HINT: Placing your mouse over the project bar in the **Cancer Distribution** panel will bring up a tooltip with useful summary data.

Q8. How many TGCA Pancreatic Adenocarcinoma *cases* (i.e. patients from the TCGA-PAAD project) have RNA-Seq data available?

Side-Note: If Barry forgets please remind him to show the top miss-sense mutation sites on the protein structure (and discuss mechanism) as well as demo how to find RNA-Seq FASTQ files and other data such as biopsy images etc.



By now it should be clear that the NCI-GDC is a rich source of both genomic and clinical data for a wide range of cancers. For example, at the time of writing there are 5,306 files associated with Pancreatic Adenocarcinoma and 14,278 for Colon Adenocarcinoma. These include RNA-Seq, WXS (whole exome sequencing), Methylation and Genotyping arrays as well as well as rich metadata associated with each case, file and biospecimen.

2. The GenomicDataCommons R package

The GenomicDataCommons (https://bioconductor.org/packages/release/bioc/html/GenomicDataCommons.html) Bioconductor package provides functions for querying, accessing, and mining the NCI-GDC in R. Using this package allows us to couple large cancer genomics data sets (for example the actual RNA-Seq, WXS or SNP data) directly to the plethora of state-of-the-art bioinformatics methods available in R. This is important because it greatly facilitates both targeted and exploratory analysis of molecular cancer data well beyond that accessible via a web portal.

This section highlights how one can couple the GenomicDataCommons (https://bioconductor.org/packages/release/bioc/html/GenomicDataCommons.html), TCGAbiolinks (http://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html) and maftools (https://bioconductor.org/packages/release/bioc/html/maftools.html) bioconductor packages to quickly gain insight into public cancer genomics data-sets.

We will first use functions from the `GenomicDataCommons` package to identify and then fetch, using the `TCGAbiolinks` package, somatic variant results from the NCI-GDC and then provide a high-level assessment of those variants using the `maftools` package. The later package works with Mutation Annotation Format (https://wiki.nci.nih.gov/display/TCGA/Mutation+Annotation+Format+(MAF)+Specification) or **MAF** format files used by GDC and others to store somatic variants.

The workflow will be:

- Install packages if not already installed
- Load libraries
- Identify and download somatic variants for a representative TCGA dataset, in this case pancreatic adenocarcinoma.
- Use maftools to provide rich summaries of the data.

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("GenomicDataCommons", "TCGAbiolinks", "maftools"))
```

Once installed, load the packages, as usual.

```
library(GenomicDataCommons)
library(TCGAbiolinks)
library(maftools)
```

Now lets check on GDC status:

```
GenomicDataCommons::status()
```

```
## $commit
## [1] "e4b233ceb9a8183f93005e77f0754eae0418c073"
##
## $data_release
## [1] "Data Release 15.0 - February 20, 2019"
##
## $status
## [1] "OK"
##
## $tag
## [1] "1.19.0"
##
## $version
## [1] 1
```

If this statement results in an error such as `SSL connect error`, then please see the troubleshooting section here (https://bioconductor.org/packages/release/bioc/vignettes/GenomicDataCommons/inst/doc/overview.html#ssl-connection-errors).

3. Querying the GDC from R

We will typically start our interaction with the GDC by searching the resource to find data that we are interested in investigating further. In GDC speak this is called *"Querying GDC metadata"*. Metadata here refers to the extra descriptive information associated with the actual patient data (i.e. 'cases') in the GDC.

> **For example**: Our query might be '**find how many patients were studied for each major project**' or '**find and download all gene expression quantification data files for all pancreatic cancer patients**'. We will answer both of these questions below.

The are four main sets of metadata that we can query with this package, namely `cases()`, `projects()`, `files()`, and `annotations()`. We will start with `cases()` and use an example from the package associated publication (https://www.biorxiv.org/content/biorxiv/early/2017/04/04/117200.full.pdf) to answer our first question above (i.e. find the number of cases/patients across different projects within the GDC):

```
cases_by_project <- cases() %>%
  facet("project.project_id") %>%
  aggregations()
head(cases_by_project)
```

```
## $project.project_id
##                key doc_count
## 1            FM-AD     18004
## 2       TARGET-NBL      1127
## 3        TCGA-BRCA      1098
## 4       TARGET-AML       988
## 5        TARGET-WT       652
## 6         TCGA-GBM       617
## 7          TCGA-OV       608
## 8        TCGA-LUAD       585
## 9        TCGA-UCEC       560
## 10       TCGA-KIRC       537
## 11       TCGA-HNSC       528
## 12        TCGA-LGG       516
## 13       TCGA-THCA       507
## 14       TCGA-LUSC       504
## 15       TCGA-PRAD       500
## 16    NCICCR-DLBCL       489
## 17       TCGA-SKCM       470
## 18       TCGA-COAD       461
## 19       TCGA-STAD       443
## 20       TCGA-BLCA       412
## 21       TARGET-OS       381
## 22       TCGA-LIHC       377
## 23       TCGA-CESC       307
## 24       TCGA-KIRP       291
## 25       TCGA-SARC       261
## 26       TCGA-LAML       200
## 27       TCGA-ESCA       185
## 28       TCGA-PAAD       185
## 29       TCGA-PCPG       179
## 30       TCGA-READ       172
## 31       TCGA-TGCT       150
## 32   TARGET-ALL-P3       131
## 33       TCGA-THYM       124
## 34       TCGA-KICH       113
## 35        TCGA-ACC        92
## 36       TCGA-MESO        87
## 37        TCGA-UVM        80
## 38       TARGET-RT        75
## 39       TCGA-DLBC        58
## 40        TCGA-UCS        57
## 41       TCGA-CHOL        51
## 42     CTSP-DLBCL1        45
## 43     TARGET-CCSK        13
## 44 VAREPOP-APOLLO         7
```

Note that the **facet()** and **aggregations()** functions here are from the `GenomicDataCommons` package and act to group all cases by the project id and then count them up.

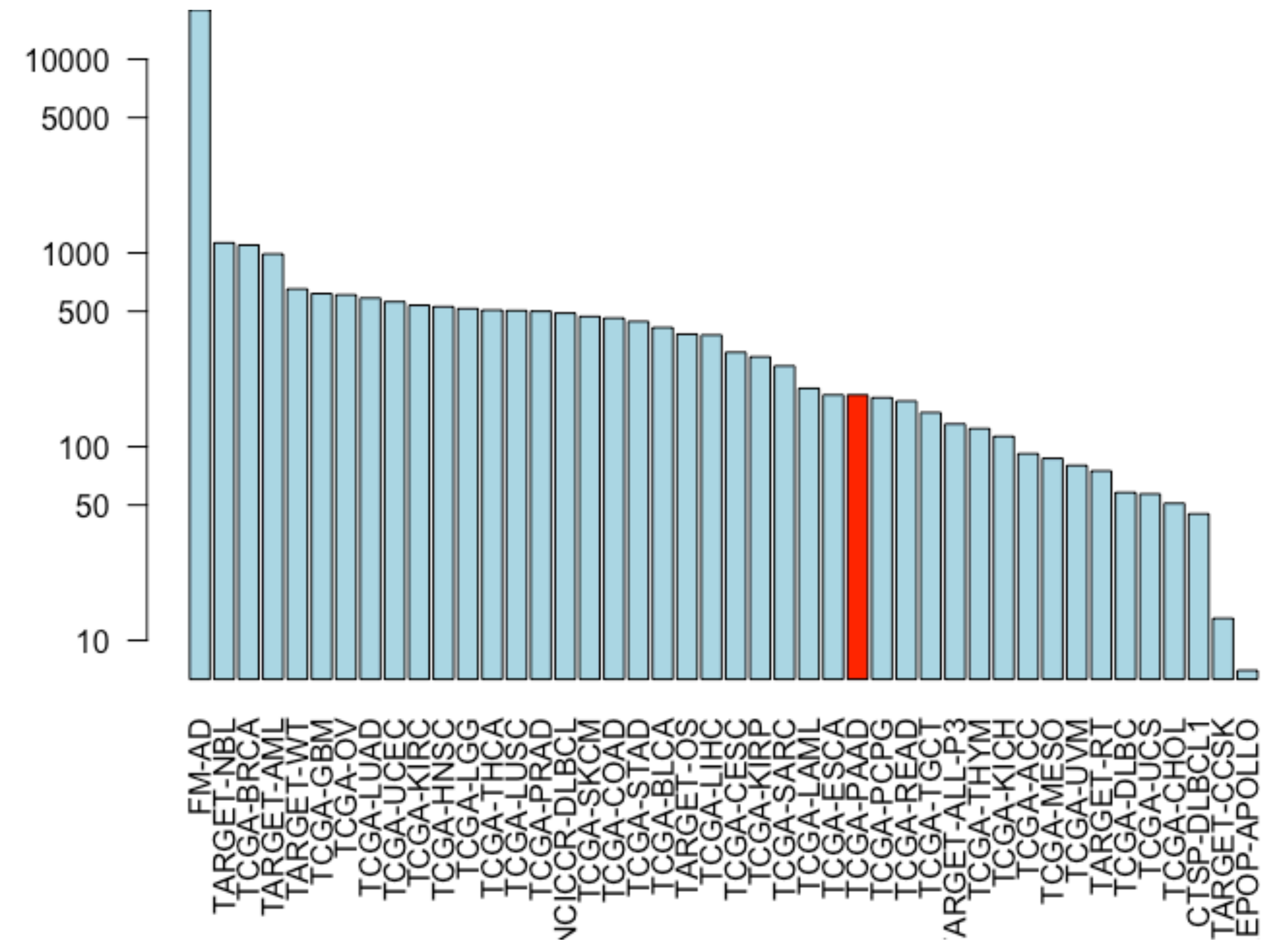
If you use the **View()** function on our new `cases_by_project` object you will find that the data we are after is accessible via `cases_by_project$project.project_id`.

Q9. Write the R code to make a barplot of the cases per project. Lets plot this data with a log scale for the y axis (`log="y"`), rotated axis labels (`las=2`) and color the bar coresponding to the TCGA-PAAD project.

```
x <- cases_by_project$project.project_id

# Make a custom color vector for our plot
colvec <- rep("lightblue", nrow(x))
colvec[___] <- "red"

# Plot with 'log' for y axis and rotate labels with 'las'
#par(___)
barplot(___, names.arg=___, log="y", col=colvec, las=2)
```



Lets explore some functions from the related `TCGAbiolinks` package. In particular we can use the **GDCquery()** function from this package to answer our second question from above - namely '*find all gene expression data files for all pancreatic cancer patients*':

```
query <- GDCquery(project="TCGA-PAAD",
                  data.category="Transcriptome Profiling",
                  data.type="Gene Expression Quantification")

ans <- getResults(query)
```

```
head(ans)
```

	data_release <chr>	data_type <chr>
1	12.0 - 15.0	Gene Expression Quantification
2	12.0 - 15.0	Gene Expression Quantification
3	12.0 - 15.0	Gene Expression Quantification
4	12.0 - 15.0	Gene Expression Quantification
5	12.0 - 15.0	Gene Expression Quantification
6	12.0 - 15.0	Gene Expression Quantification

6 rows | 1-3 of 29 columns

In RStudio we can now use the **View()** function to get a feel for the data organization and values in the returned `ans` object.

```
View(ans)
```

We should see that `file_records$results` contains a row for every RNA-Seq data file from the 'TCGA-PAAD' project. At the time of writing this was 546 RNA-Seq data files.

```
nrow(ans)
```

```
## [1] 546
```

We could download these with standard R tools, or for larger data-sets such as this one, use the packages **transfer()** function, which uses the GDC transfer client (a separate command-line tool) to perform more robust data downloads.

4. Variant analysis with R

Note we could go to the NCI-GDC web portal and enter the Advanced Search page (https://portal.gdc.cancer.gov/query) and then construct a search query to find MAF format somatic mutation files for our 'TCGA-PAAD' project.

After some exploration of the website I came up with the following query:
" `cases.project.project_id in ["TCGA-PAAD"] and files.data_type in ["Masked Somatic Mutation"] and files.data_format`

> **Q9**. How many MAF files for the TCGA-PAAD project were found from this advanced web search?

Lets do the same search in R with the help of the `TCGAbiolinks` package function **GDCquery_Maf()**. For brevity we will focus on only one of the MAF files for this project in GDC, namely the MuTect2 workflow variant calls.

```
maf.file <- GDCquery_Maf(tumor="PAAD", pipelines = "mutect")
```

And lets take a peak at the first 6 rows of this data:

```
head(maf.file)
```

Hugo_Symbol <chr>	Entrez_Gene_Id <int>	Center <chr>	NCBI_Build <chr>	Chromosome <chr>	Start_Position <int>
BCAN	63827	BI	GRCh38	chr1	156651635
TNN	63923	BI	GRCh38	chr1	175135891
PM20D1	148811	BI	GRCh38	chr1	205850012
CR1	1378	BI	GRCh38	chr1	207523807
MLK4	84451	BI	GRCh38	chr1	233372058
ITSN2	50618	BI	GRCh38	chr2	24310368

6 rows | 1-6 of 120 columns

> **Q10**. What argument could we use to write the MAF file into a csv document in your current working directory?

MAF analysis

The MAF file contents is now stored as a dataframe and the maftools package workflow, which starts with a MAF file or dataframe, can proceed, starting with reading the pancreatic cancer MAF file.

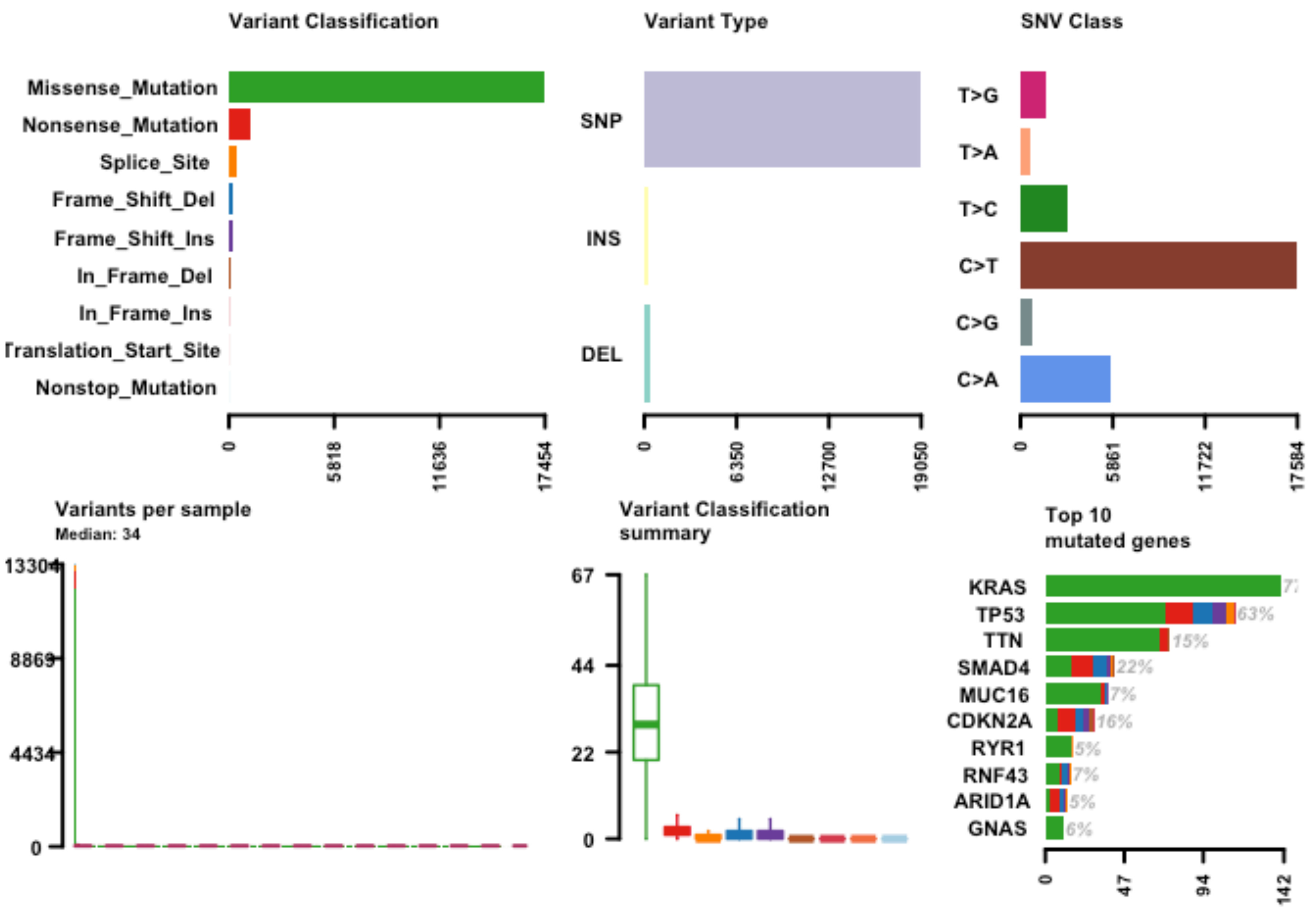
```
vars = read.maf(maf = maf.file, verbose = FALSE)
```

With the data now available as a **maftools** MAF object, a lot of functionality is available with little code. While the maftools package offers quite a few functions, here are a few highlights. Cancer genomics and bioinformatics researchers will recognize these plots:

Plotting MAF summary.

We can use `plotmafSummary()` function to plot a summary of the maf object, which displays number of variants in each sample as a stacked barplot and variant types as a boxplot summarized by Variant_Classification. We can add either mean or median line to the stacked barplot to display average/median number of variants across the cohort.

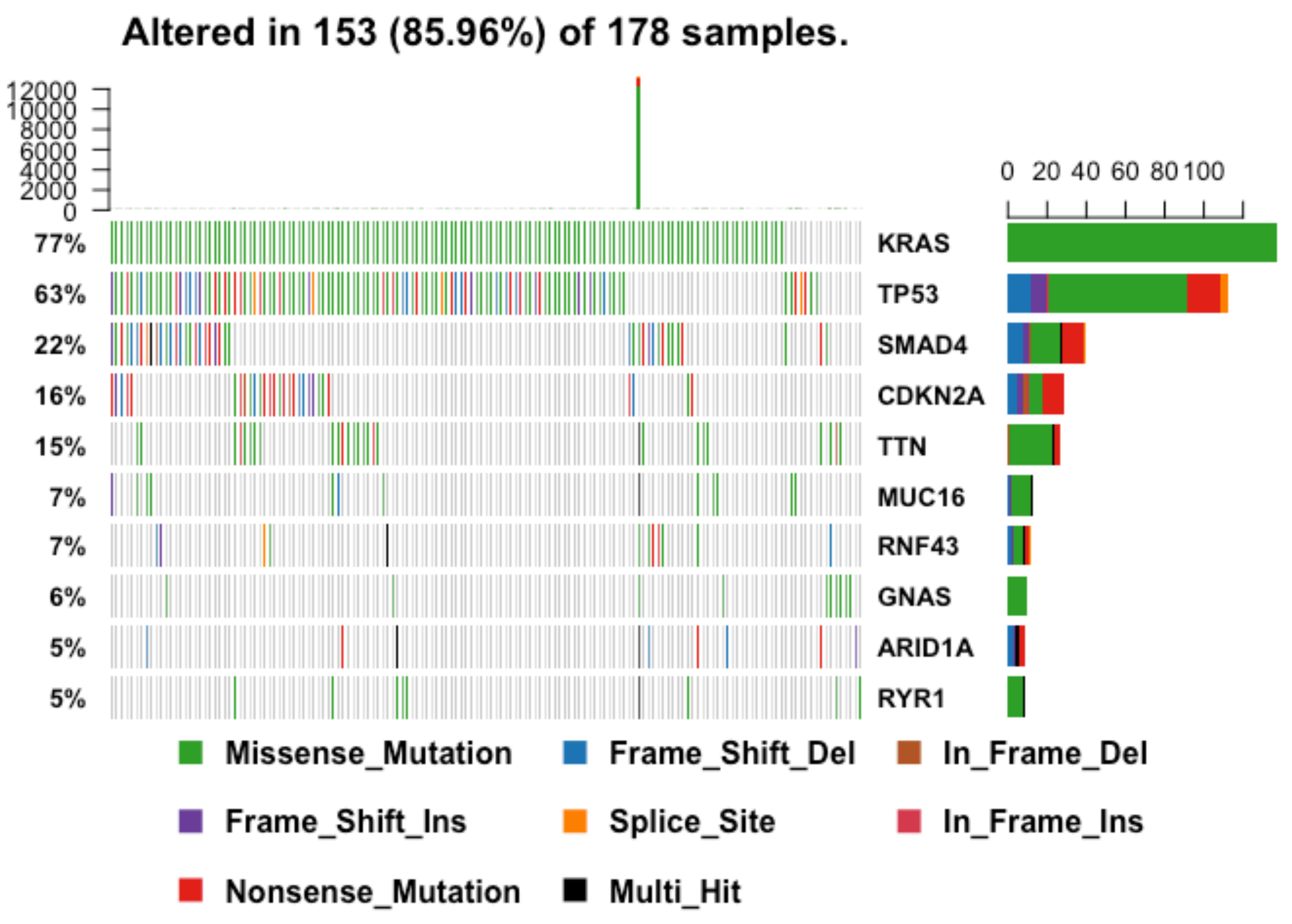
```
plotmafSummary(vars)
```



Drawing oncoplots

A very useful summary representation of this data can be obtained via so-called *oncoplots*, also known as *waterfall plots*.

```
oncoplot(maf = vars, top = 10)
```



You might need to run the **oncoplot()** command in the R Console and then zoom the display to see the full plot (as it is rather large and may not appear initially in your Rmarkdown document before Knitting. Another option is to send your plot to a PNG or PDF plot device directly, for example:

```
# Oncoplot for our top 10 most frequently mutated genes
pdf("oncoplot_panc.pdf")
oncoplot(maf = vars, top = 10, fontSize = 12)
dev.off()
```

```
## quartz_off_screen
##                 2
```

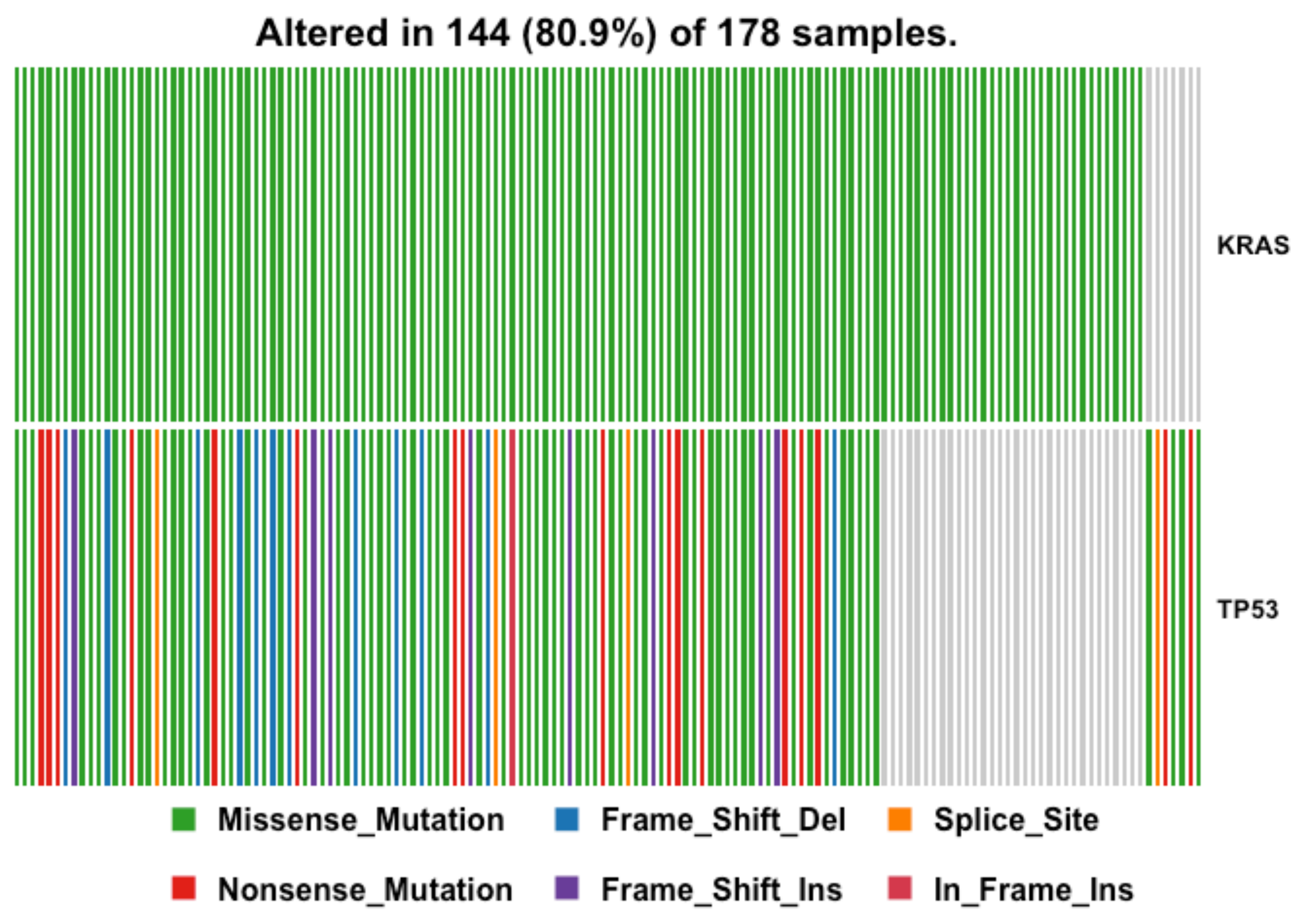
NOTE: The **oncoplot()** function is a wrapper around ComplexHeatmap's `OncoPrint()` function and there are lots and lots of possible customization options as usual with R graphics.

NOTE: Variants annotated as Multi_Hit are those genes which are mutated more than once in the same sample.

Oncostrip

We can visualize any set of genes using the **oncostrip()** function, which draws mutations in each sample similar to the graphic on the NCI-GDC web portal. Note that **oncostrip()** can be used to draw any number of genes using the input `top` or `genes` arguments

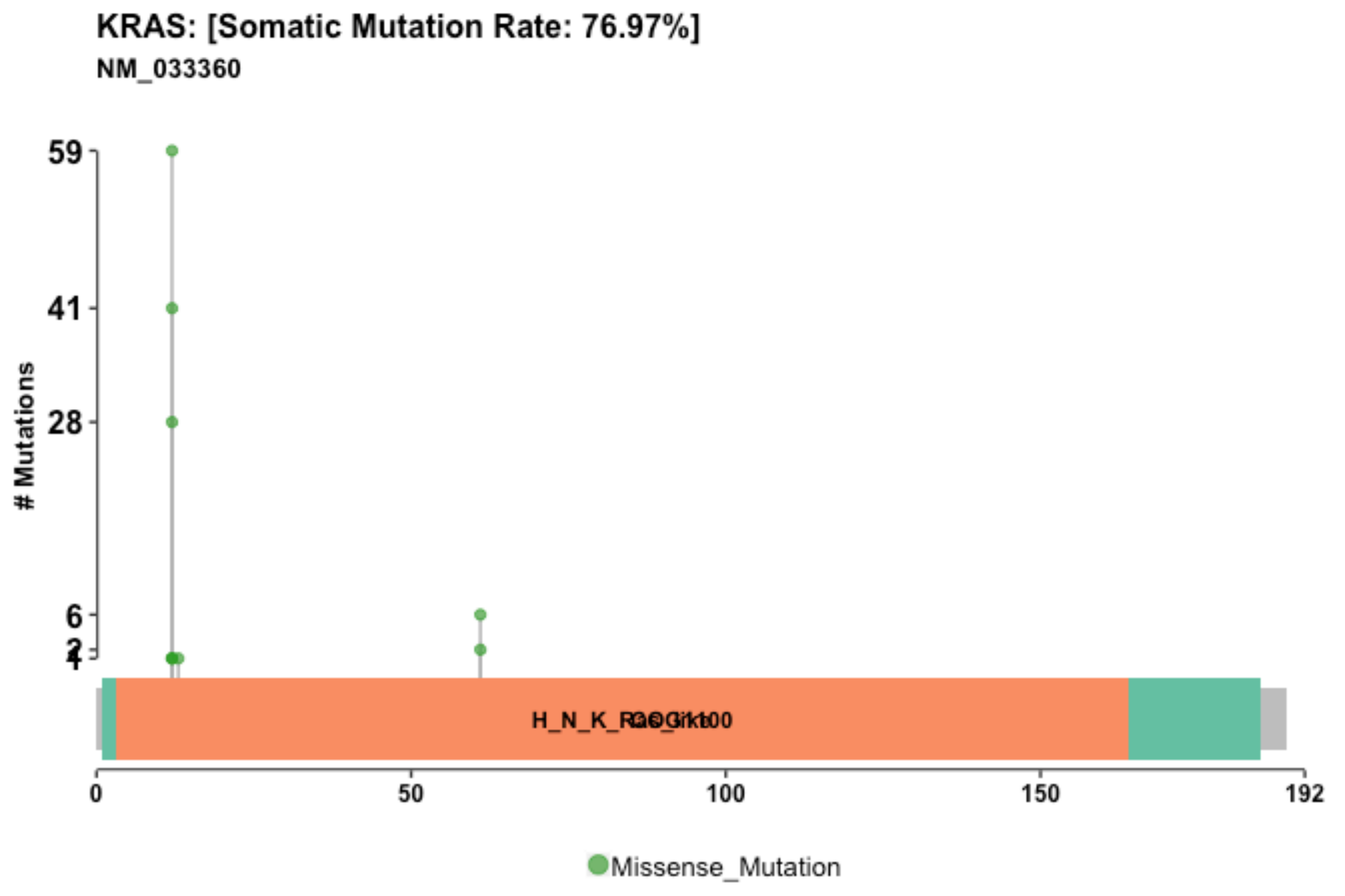
```
oncostrip(maf=vars, genes=c("KRAS", "TP53"))
```



Another plot focusing on KRAS in our particular dataset.

```
lollipopPlot(vars, gene='KRAS')
```

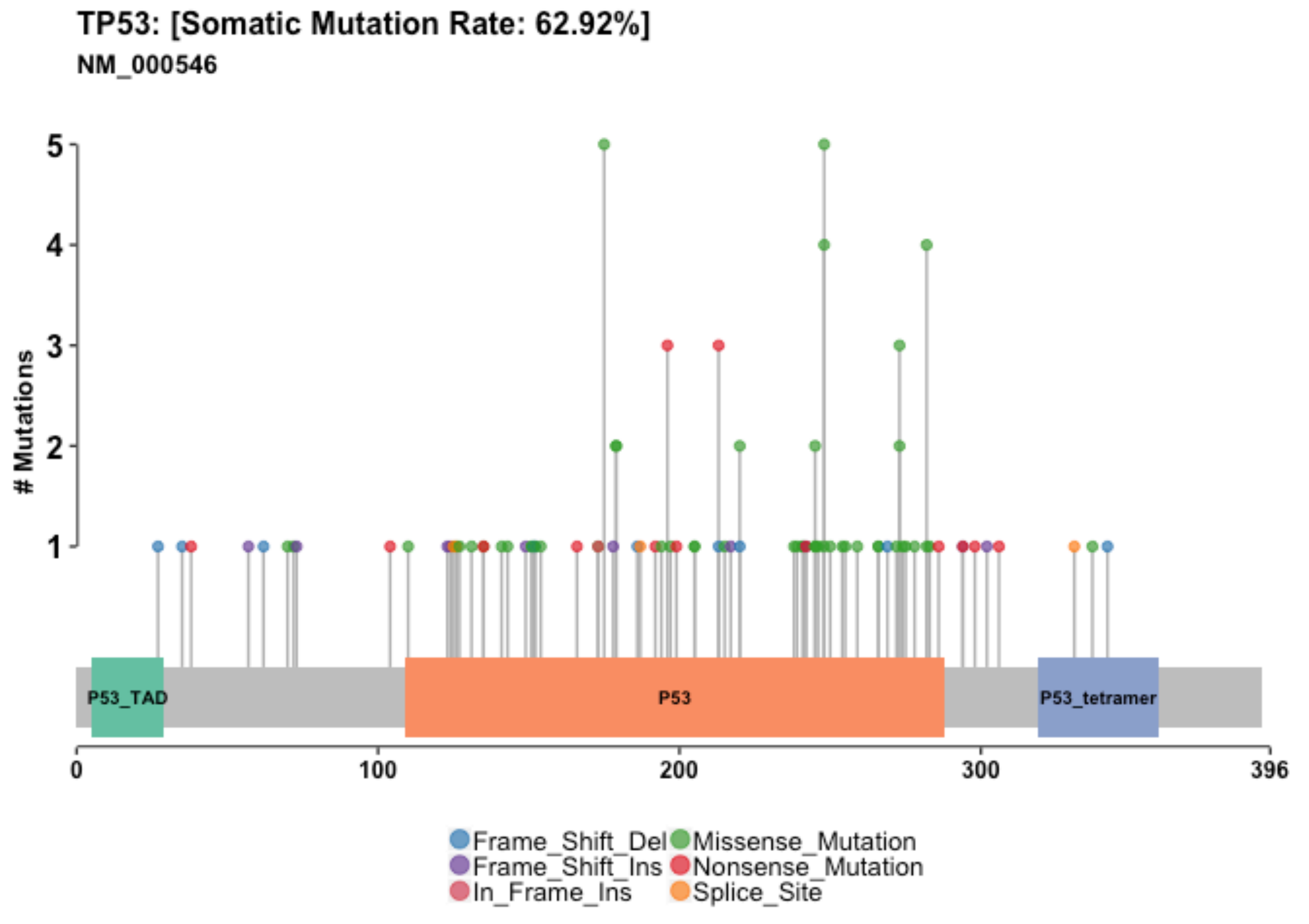
```
##    HGNC refseq.ID protein.ID aa.length
## 1: KRAS NM_004985  NP_004976       188
## 2: KRAS NM_033360  NP_203524       189
```



Q11. Adapt the code above to produce a lollipop plot for p53 (i.e. the 'TP53' gene)?

Your p53 plot should look like this:

```
##    HGNC    refseq.ID    protein.ID aa.length
## 1: TP53    NM_000546    NP_000537       393
## 2: TP53 NM_001126112 NP_001119584       393
## 3: TP53 NM_001126118 NP_001119590       354
## 4: TP53 NM_001126115 NP_001119587       261
## 5: TP53 NM_001126113 NP_001119585       346
## 6: TP53 NM_001126117 NP_001119589       214
## 7: TP53 NM_001126114 NP_001119586       341
## 8: TP53 NM_001126116 NP_001119588       209
```



Summary

Additional functionality is available for each of the `GenomicDataCommons`, `TCGAbiolinks` and `maftools` packages not to mention the 100's of other bioinformatics R packages that can now work with this type of data in both exploratory and targeted analysis modes.

The purpose of this hands-on session was to highlight how one can leverage three such packages to quickly gain insight into rapidly expanding public cancer genomics data-sets. Hopefully this will inspire your further exploration of these and other bioinformatics R packages.