Tutorial for Complex OncoPrint Generator

Noushin Farnoud (rahnaman@mskcc.org)

Elli Papaemmanuil Lab @ MSKCC

Contents

List of Figures

2 About

This is the documentation of the generate_complex_oncoprint suit of functions and describes the features you can add and/or control using simple input parameters. These functions are built upon ComplexHeatmap package of Bioconductor and aim to add more simplicity, flexibility and functionality for OncoPrint generation.

- For **generate_complex_oncoprint** suite, download from Papaemmanuil lab's Github repository:
  https://github.com/papaemmelab/analysisTools/tree/master/src/tools/R/Oncoprint
  If you can not access this repository, you can reach out to me directly to request it (Noushin Farnoud: rahnaman@mskcc.org).

- For **ComplexHeatmap**, download from:
  http://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html, or for the latest version directly from Github:

```
library(devtools)
install_github("jokergoo/ComplexHeatmap")
```

The examples presented in this tutorial are based on ComplexHeatmap version 2.5.4 (Github) using R version 3.6.3 (R Session info: Section 13).

3 Quick Start

Let's jump start with a simple example to generate Figure 1. You have a list of SNVs and INDELs and would like to generate an OncoPrint that depicts these mutations and color them according to their effect (e.g., missense, nonesense).

We start by loading an example dataset (also included in the directory), and selecting variants that have been annotated as oncogenic or likely oncogenic:

```
MUT  <- read_excel("./example_data/Mutations.xlsx")
MUT  <-  MUT %>% filter(Noush_annot2 %in% c("ONCOGENIC","LIKELY"))
```

3.1 Required Fields for MUT Dataframe

You notice that the loaded mutations dataframe **MUT** has many columns, although, the only **required MUT** columns for **generate_complex_oncoprint** are:

1. **TARGET_NAME**
2. **GENE**

3. **EFFECT**: For example, missense, frameshift_variant (see next page for details).

IMPORTANT

The only column that it's content must belong to one of these pre-defined values is **EFFECT** (i.e., the type of the mutation). I highly recommend that you make sure the specified EFFECTs in your mutation dataframe belong to one of the options that are listed at Section 11.

4 Example 1: Simple OncoPrint

Now let's see how we can generate our first OncoPrint. The minimal code would look something like this. Make sure you change your current working directory to the cloned oncoprint folder:

```
source("./generate_complex_oncoprint.R")
ht <-  generate_complex_oncoprint(muts= MUT,
        show.sample.names = TRUE,
        min.freq = 1,
        show.title= TRUE,
        title.str=  "Example1-YOHOOOO. I managed to get this without any error!...",
        File.Name= "Example_1_A", #NAME ofs the saved oncoprint
        save.path= "./example_oncoprints/", #DIR of the saved oncoprint figw
        )
```

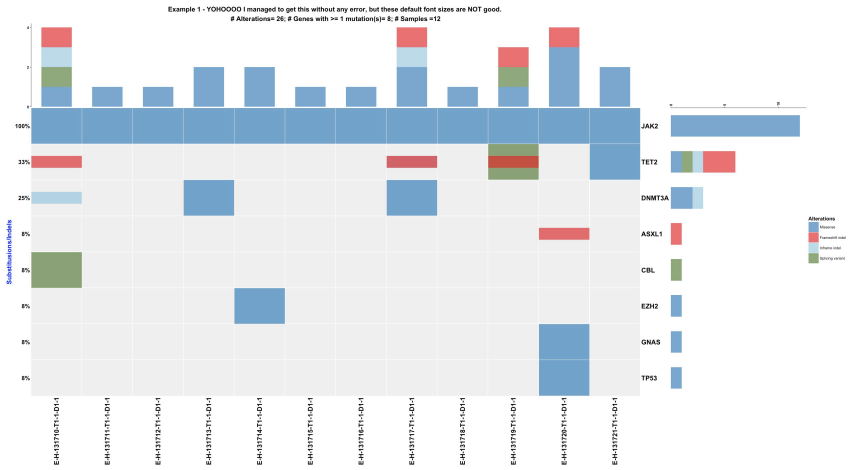
The mutation dataframe is passed to **generate_complex_oncoprint** via `mut` argument. As seen in Figure 1a, the default font sizes are too small and the grid looks a bit stretched. These are some of the main issues when you want to generate a publication quality figure. This tool allows you to easily control almost all of these features.

Now let's continue with a basic upgrade to the previous plot by setting up a few arguments to change the font sizes and the widths of the barplots (Figure 1b):

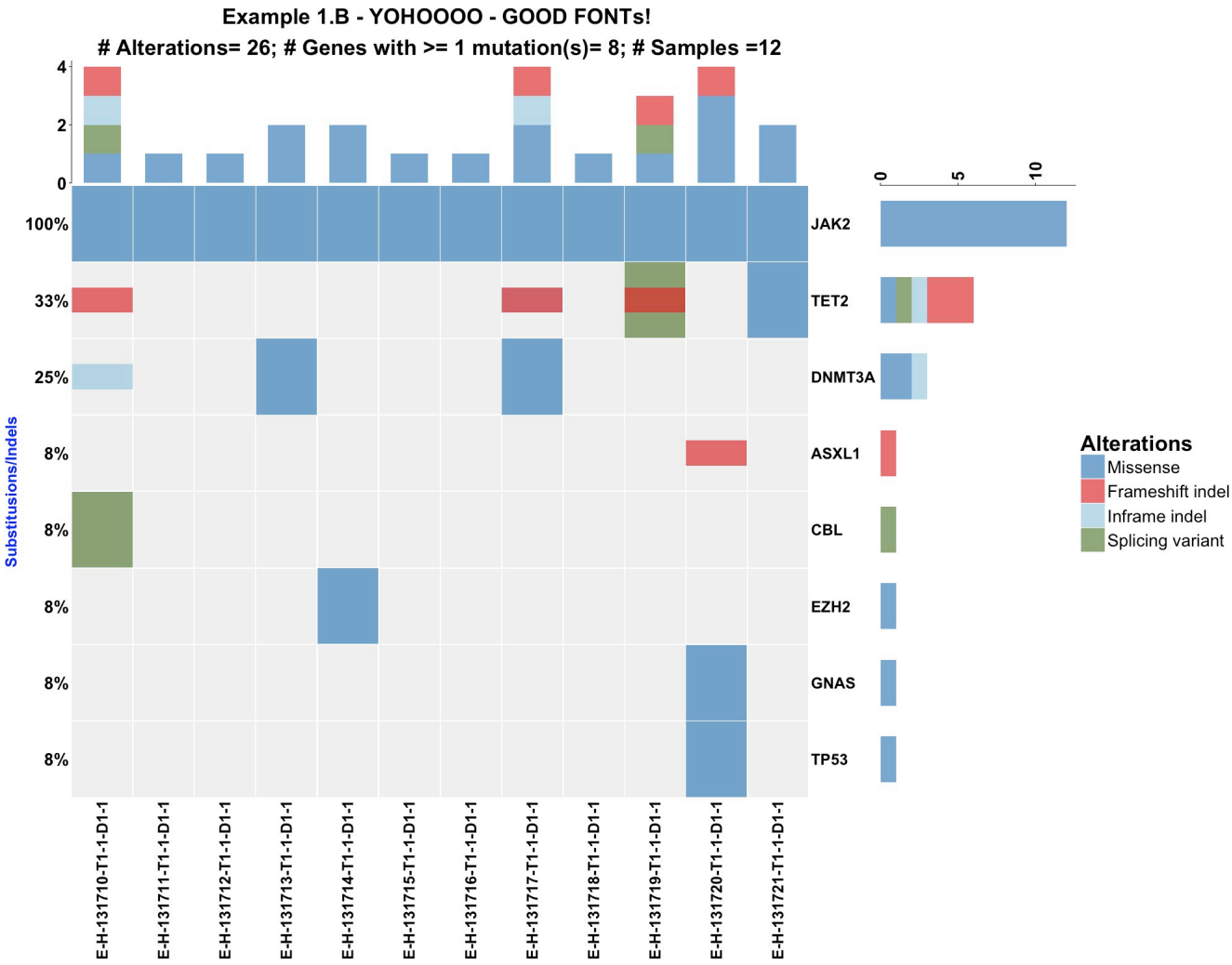
```
ht <-  generate_complex_oncoprint(muts= MUT, show.sample.names = TRUE, min.freq= 1,
        show.title= TRUE, title.str=  "Example 1.B - YOHOOOO- GOOD FONTs!",
        File.Name= "Example_1_B", save.path= "./example_oncoprints/",
        #*** NEW: Change the font sizes (see next section for details)
        cols.font= 20,
        rows.font= 18,
```

```
        pct.font= 20,
        legend.label.font= 20, legend.title.font= 25,
        fig.title.font= 25,
        barplot.font= 20,
        right.w= 8, top.w= 5 ,
        #*** Change the figure width and height to get a nice oncoprint
        w=2300, h=2400,
        )
```

For example, here `cols.font = 20` increases the size of the sample labels (i.e. column labels) (default 18). Next section provides a comprehensive depiction of major **generate_complex_oncoprint** arguments that facilitate the "beautification" process (Section 5).



(a)



(b)

Figure 1: Simple OncoPrint of mutations.

5 Controlling Fonts

Figure 2 depicts all major fonts as well as barplot sizes that can be easily controlled by setting specified arguments in input **generate_complex_oncoprint** function.

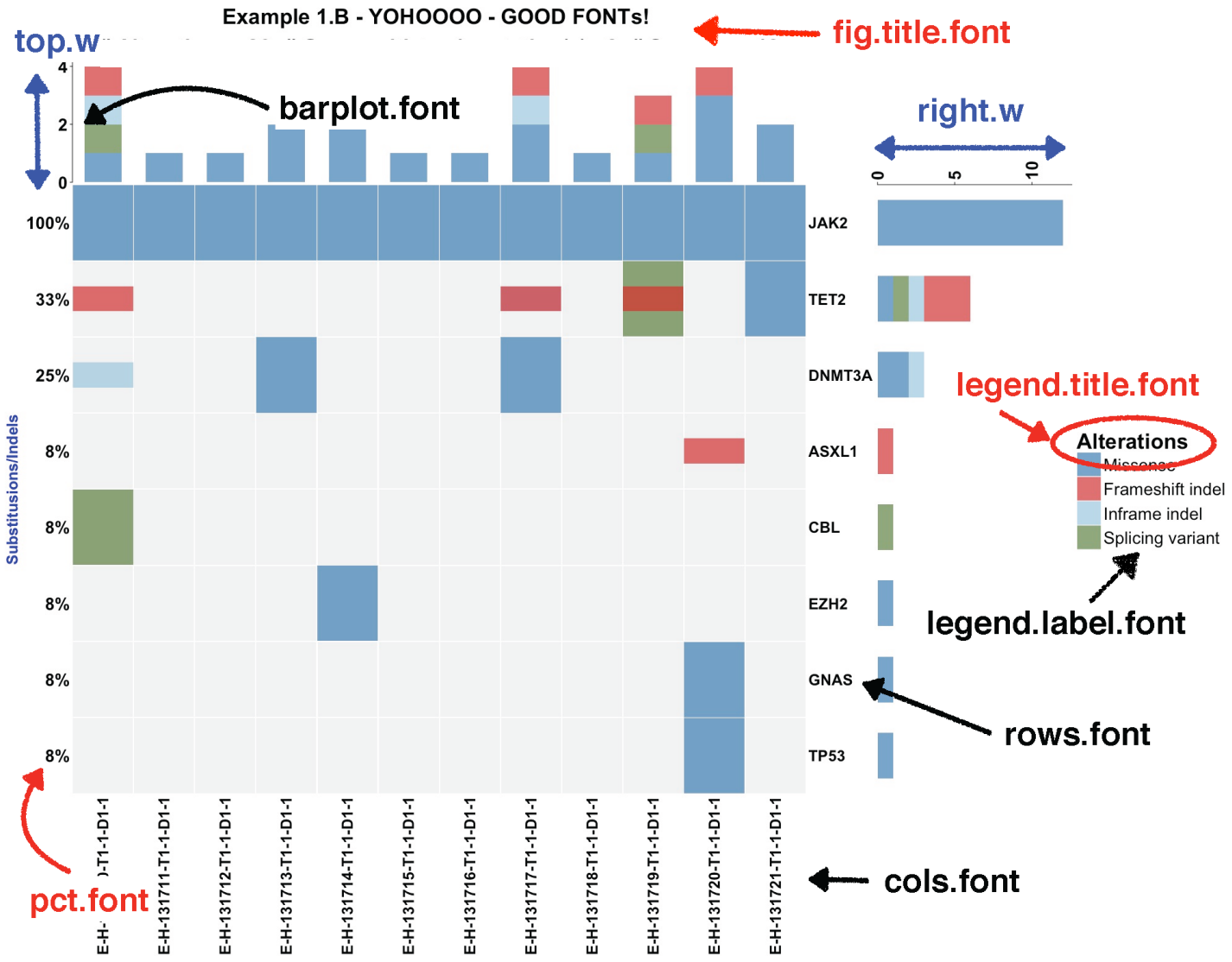


Figure 2: Useful **generate_complex_oncoprint** arguments to control OncoPrint appearance.

6 Example 2: Adding CNVs and SV

This tool allows addition of SVs and CNVs as separate sections to the mutations OncoPrint. Let's examine this by loading a new example:

```
rm(list=ls()) #clear all vars
load("./example_data/Example2.RData")
```

The loaded data includes dataframes for mutations (*MUTs*), as well as copy number variants (*cnvs*) and structural variants (*SVs*). Just assign these dataframes to `muts`, `svs` and `cnvs` arguments:

```
source("./generate_complex_oncoprint.R")
ht <- generate_complex_oncoprint(muts= MUTs, svs= SVs, cnvs=cnvs,
          min.freq= 1,
          title.str= "Example 2 - Mutuations + CNVs + SVs"
        )
```

6.1 Required Fields for CNVs and SVs Dataframes

1. **TARGET_NAME**: sample name.
2. **VAR_ID**: a string representative of the variant id; e.g., 1q, or CDKN2A. Avoid special characters.
3. **EFFECT**: type of the reported variant; e.g., AMP, FUSION.

NOTE: Notice how in Figure 3 the location of the alterations legend has changed from default position (right) to the bottom. Also, the number of rows that are shown in the legend ares limited to 2. These changes are achieved by setting these arguments:

```
...
           heatmap.legend.side= "bottom",
           num.rows.heatmap.lgd= 2,
           mut.legend.title.side = "leftcenter",
...
```

Often you need to play around with these to get a nice plot for the final figure! Try changing these parameters to see different outputs.

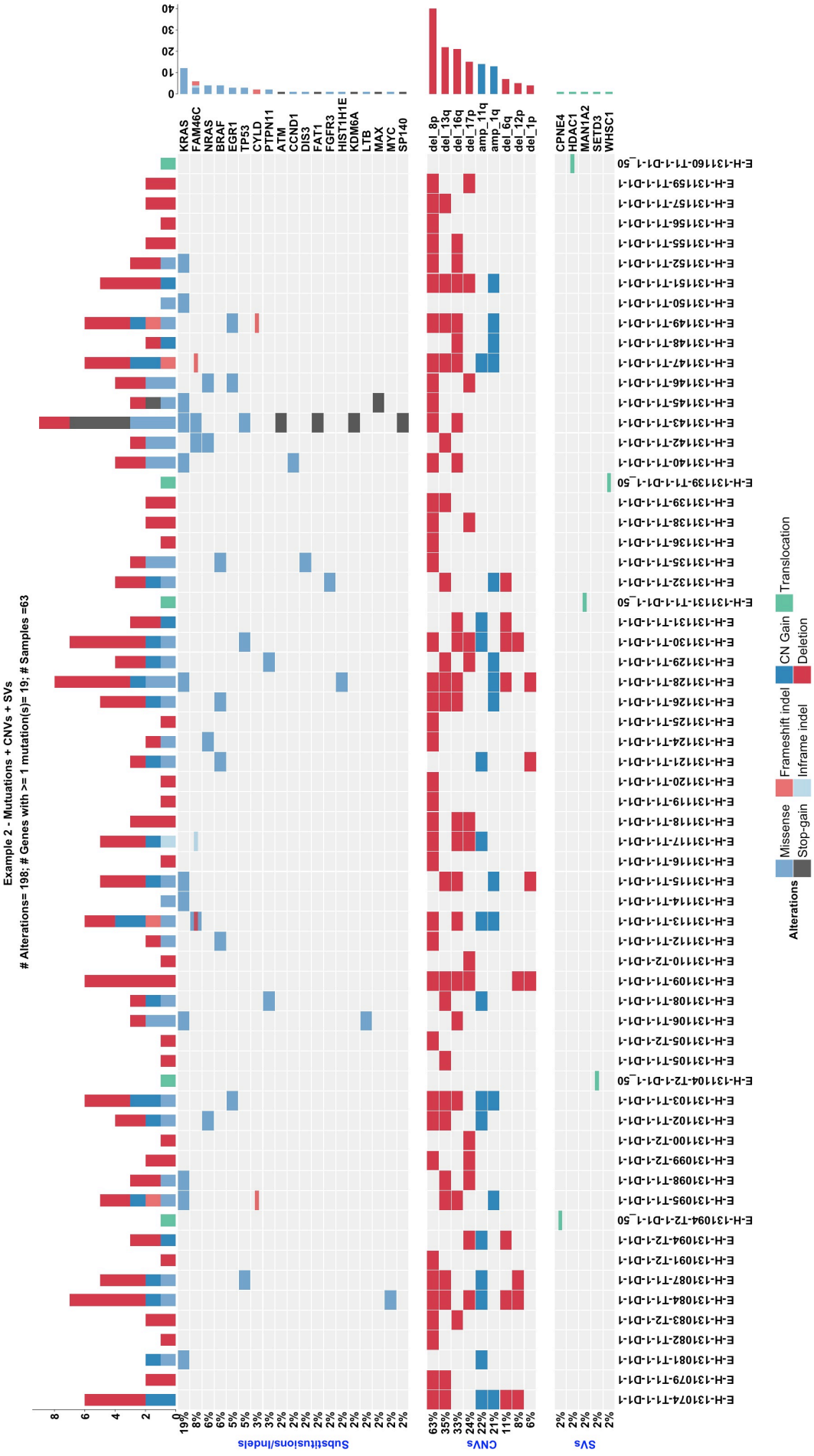


Figure 3: OncoPrint with individual segments for MUTATIONS, CNVs and SVs.

7 Example 3: Adding RESPONSE Annotation (built-in option)

generate_complex_oncoprint has default arguments to add RESPOSNE and INDIVIDUAL.ID annotation banners as well as any other single banner (total of 3 banners) to the OncoPrint.

IMPORTANT

The most important step for proper depiction of annotation ribbons in **generate_complex_oncoprint** is passing a complete sample-level lookup table via `lookup.table` argument. This lookup table allows the code to connect sample names (TARGET_NAME) in the alterations dataframe to their corresponding RESPONSE, INDIVIDUAL.ID, or any other sample-specific feature you like to visualize (See Figures 4-5).

7.1 Required Columns for RESPONSE Annotation

For adding the built-in **RESPONSE** annotation the `lookup.table` must include these columns:

1. **TARGET_NAME**
2. **RESPONSE**: to see a list of available options refer to Section 11.

7.2 Generate OncoPrint with RESPONSE

For simplicity, the code below only shows the new arguments that were set to depict RESPONSE annotation and control its appearance:

```
source("./generate_complex_oncoprint.R")
ht <-  generate_complex_oncoprint(muts= MUTs, cnvs= cnvs, svs= SVs ,
        ...
        #***Show RESPONSE banner
        show.response= TRUE,
        lookup.table= my.table, #******* lookup.table REQUIRED
        show.sample.names = TRUE,

        #***Control the location of the LEGENDs (for heatmap(mut) and RESPONSE)
        heatmap.legend.side= "right", mut.legend.title.side= "topcenter",
        annot.legend.side= "bottom", annot.title.side= "topleft",
```

```
        num.rows.annot.lgd= NULL,
        ....
        )
```

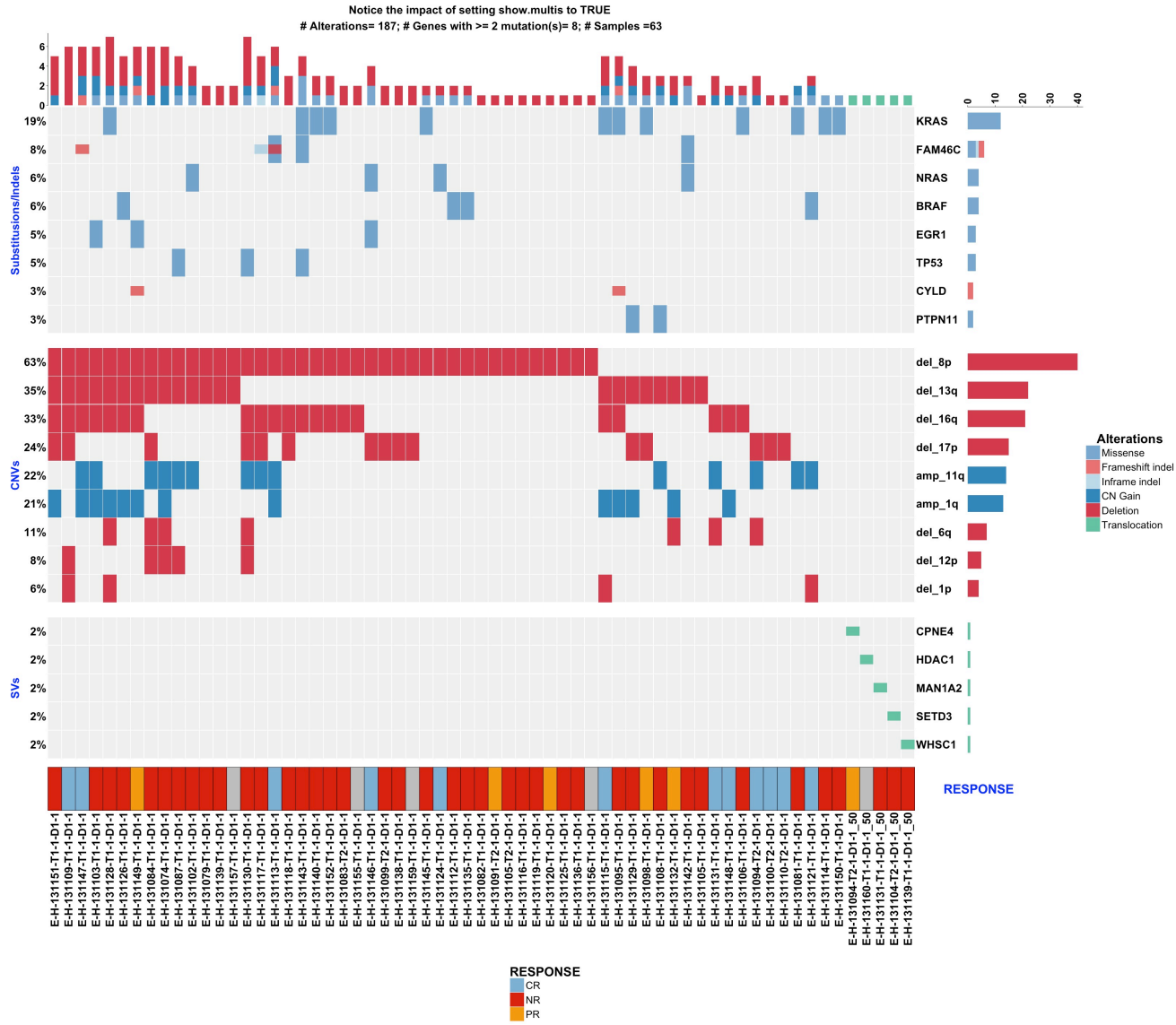


Figure 4: OncoPrint with RESPONSE annotation (built-in argument)

8 About TEMP Figures

Have you noticed the TEMP additional OncoPrints that are automatically generated and saved in your defined path?

It is VERY IMPORTANT to review the TEMPs and check a few samples to make sure the info is shown correctly, specially if you are adding annotation ribbons or are passing a pre-defined order of samples in **generate_complex_oncoprint**.

The issue is that sometimes due to factor levels you may not be aware of, or updates in your code or in ComplexHeatmap package, you may get inconsistencies. That is why I hard-coded the TEMP OncoPrint which is the very basic overview of the data and always depicts the sample names, even if you had set show.sample.names = FALSE for your main publication figure.

9 Example 4: Adding INDIVIDUAL Annotation (built-in option)

This is the same concept as presented in previous example (Sec. 7). It is useful when working with data from patients with multiple-samples and you like to visually detect whether patient samples are/not clustered together. The example will be added later.

10 Example 5: OncoPrint with Multi-hits and a New Feature Annotation

I often like to be able to visualize what genes have multiple hits in my OncoPrints. Also, sometimes you may have events that are overlapping and not clear. For example, due to this limitation you will not be able to easily distinguish samples with mono- versus multi- allelic *TP53* mutations in the default ComplexHeatmap OncoPrint.

Adding multis label is not as simple as it sounds and is prone to mess up clustering, but need you not worry Dear Papaemmanuil Lab! :)). I have added a new option to specifically mark "multi-hit" genes in Subs/Indel panel, as shown in Figure 5.

10.1 How to Depict Multi-hits

In Figure 5 you see that any gene with > 1 mutation is marked with a white dot by setting `show.multis= TRUE` argument (default FALSE). The size of this marker is also easily controllable via `multis.dot.size` argument (default 0.8). All you need to do for depicting multi-hits is adding the below lines:

```
...
show.multis= TRUE,
multis.dot.size = 0.9,
```

10.2 Add New Feature Annotation

Also, in Figure 5 you notice the addition of a new banner (RACE). That is done by:

```
....
show.response= TRUE,
# **** Add new banner for RACE
show.another.banner=TRUE, banner.name= "Race",
lookup.table= my.table, # REQUIRED for banner representation
...
```

*****IMPORTANT*****

Remember for adding any banner you must pass a lookup table. But for any new banner which is not part of the 2 built-in banners (RESPONSE and INDIVIDUAL), you must also tell **generate_complex_oncoprint** function which column of the lookup table (which feature) to use for annotation. This is simply done by setting show.another.banner = TRUE and passing the name of the feature column to banner.name argument (case-sensitive).

For example, in the code above for annotating RACE we passed `banner.name= "Race"` because that is the exact name of the lookup table column for this feature in this example.

This latest upgrade is very useful for adding any new annotation, such as *treatment*, *survival status*, etc. So I highly recommend to clone the latest version of **generate_complex_oncoprint** from github (See it on Github).

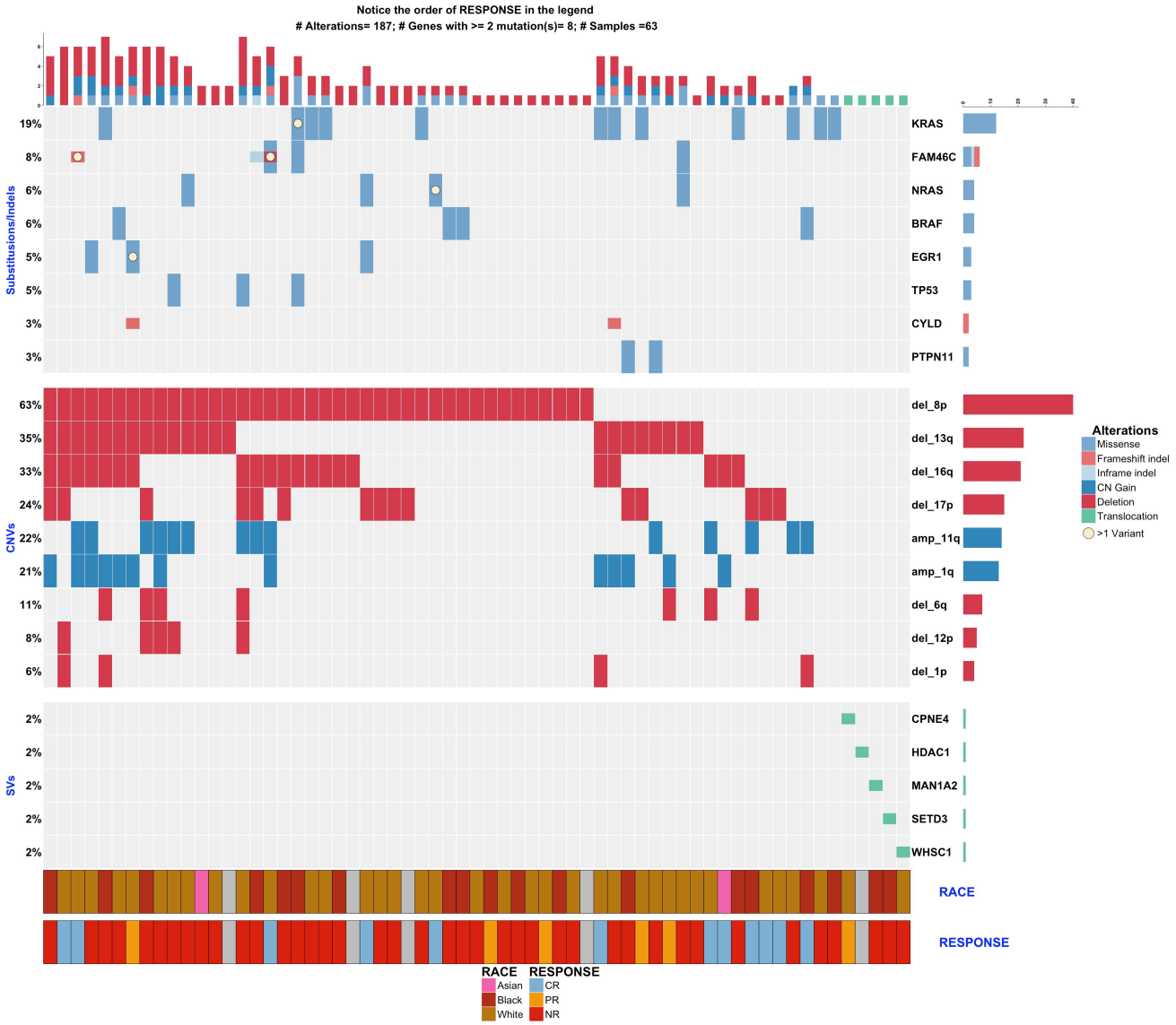


Figure 5: OncoPrint with multi-hit genes and an additional banner (Race)

11 List of Pre-defined EFFECT Options for Mutations, CNVs and SVs

Valid options for EFFECT field of all input alteration dataframes (Subsitutions/INDELs/CNVs/SVs):

```
non_synonymous_codon, missense, missense_codon,
stop_gained, stop_lost, stop_retained_variant, stop_gain,
splice_site_variant, initiator_codon_change,
inframe_codon_loss, inframe_codon_gain, inframe_deletion, inframe_insersion,
inframe_indel, inframe_variant,
complex_change_in_transcript, complex,
other_snvs, unknown,
frameshift_variant, frameshift_indel, frameshift_del,
amp, amplification, gain,
del, deletion, loss,
LOH,
inv, inversion,
tandem duplications, tandem_duplications,
fusion, translocation, trans,
complex_karyotype,
normal, normal_karyotype,
other_svs, N/E, inconclusive
```

Valid options for RESPONSE:

```
persistent,partial response,non-responder,
stable disease,responder,
CR,CR-i,PR,NR,N/A,N/E
```

You can modify response colors or add new options for response in heatmap_colors.R subfunction.

12 Frequently Asked Questions

12.1 Can I limit my OncoPrint to show genes that are only mutated in at least 5 samples?

Yes. You can control this via `min.freq` argument. For example, in this case you specify `min.freq= 5` in **generate_complex_oncoprint**. Refer to Figure 3 to see the code.

12.2 Can I specify my own order of samples or genes instead of default OncoPrint clustering?

Yes. The default ordering of the OncoPrint is hierarchical clustering. But you can override this by assigning a specified orders for samples (columns) and/or genes (rows) by the following two arguments.

```
        column_order = my.sample.order,
        row_order = my.gene.order,
```

Make sure these input lists are simple vectors of strings that correspond to the exact name of the samples/genes. Dataframes or factorized vectors would lead to errors.

12.3 Can I remove the sample names and the figure title for my publication OncoPrint?

Yes. You can remove the title of the figure and the sample names by setting `show.title= FALSE` and `show.sample.names = FALSE` (both argument are TRUE as default).

12.4 Can I change annotation legends positions?

Yes. You have the options to control it. See **generate_complex_oncoprint** variable description section for details.

13 Session Info

```
## sessionInfo()
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
## ## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## ## attached base packages:
## [1] grid stats graphics grDevices utils datasets methods base
## ## other attached packages:
## [1] prettyGraphs_2.1.6 argparse_2.0.1 ComplexHeatmap_2.5.4 randomcoloR_1.1.0.1 dplyr_1.0.0
plyr_1.8.6
## [7] readxl_1.3.1.9000 gsheet_0.4.5 reshape_0.8.8
## ## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.4.6 RColorBrewer_1.1-2 cellranger_1.1.0 pillar_1.4.4 compiler_3.6.3 tools_3.6.3 di-
gest_0.6.25
## [8] clue_0.3-57 jsonlite_1.7.0 Rtsne_0.15 lifecycle_0.2.0 tibble_3.0.1 pkgconfig_2.0.3 png_0.1-7
## [15] rlang_0.4.6 rstudioapi_0.11 parallel_3.6.3 curl_4.3 findpython_1.0.5 stringr_1.4.0 cluster_2.1.0
## [22] S4Vectors_0.24.4 IRanges_2.20.2 generics_0.0.2 vctrs_0.3.1 GlobalOptions_0.1.2 stats4_3.6.3 tidys-
elect_1.1.0
## [29] glue_1.4.1 R6_2.4.1 GetoptLong_1.0.2 purrr_0.3.4 magrittr_1.5 BiocGenerics_0.32.0 scales_1.1.1
## [36] ellipsis_0.3.1 shape_1.4.4 circlize_0.4.10 colorspace_1.4-1 V8_3.2.0 stringi_1.4.6 munsell_0.5.0
## [43] crayon_1.3.4 rjson_0.2.20
```