Detection of mutational patterns in cell free DNA (cfDNA) of colorectal cancer by custom amplicon sequencing

Benedikt Rauscher

2018-12-17

Abstract

Monitoring the mutational patterns of solid tumors during cancer therapy is a major challenge in oncology. Analysis of mutations in cell free (cf) DNA offers a non-invasive approach to detect mutations that may be prognostic for disease survival or predictive for primary or secondary drug resistance. A main challenge for the application of cfDNA as a diagnostic tool is the diverse mutational landscape of cancer. Here, we developed a flexible end-to-end experimental and bioinformatics workflow to analyze mutations in cfDNA using custom amplicon sequencing. Our approach relies on open software tools to select primers suitable for multiplex PCR using minimal cfDNA as input. In addition, we developed a robust linear model to identify specific genetic alterations from sequencing data of cfDNA. We used our method to design a custom amplicon panel suitable for detection of hotspot mutations relevant for colorectal cancer and analyzed mutations in serial cfDNA samples from 34 patients with advanced colorectal cancer. Our data demonstrates that recurrent and patient-specific mutational patterns can be reliably detected for the majority of patients. Furthermore, we show that the allele frequency of mutations in cfDNA correlates well with disease progression. Finally, we demonstrate that monitoring of cfDNA can outperform the predictive power of currently used tumor markers and reveal mechanisms of resistance to anti-EGFR antibody treatment.

Contents

Detection of mutational patterns in cell free DNA (cfDNA) of colorectal cancer by custom amplicon sequencing

1 About

This document contains computer code to reproduce analyses and figures presented in the corresponding study manuscript.

2 Dependencies

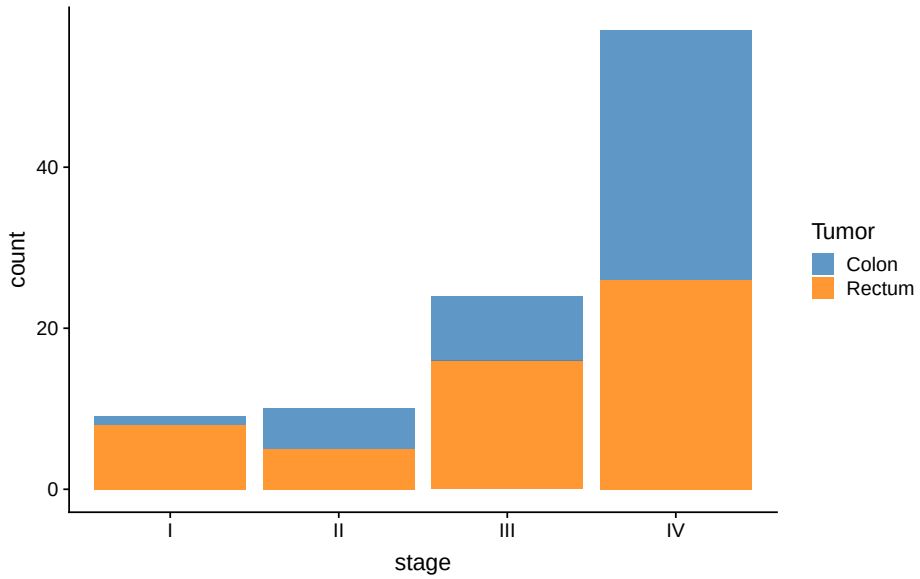
We load a number of packages which provide functionality that is required for downstream analyses.

```
library(GenomicRanges)
library(tidyverse)
library(openxlsx)
library(pheatmap)
library(ggsignif)
library(cowplot)
library(biomaRt)
library(gtools)
library(patchwork)
library(gtools)
library(Biostrings)
library(MASS)
library(BSgenome)
library(BSgenome.Hsapiens.UCSC.hg38)
library(genemodel)
library(treemap)
library(GenomicFeatures)
```

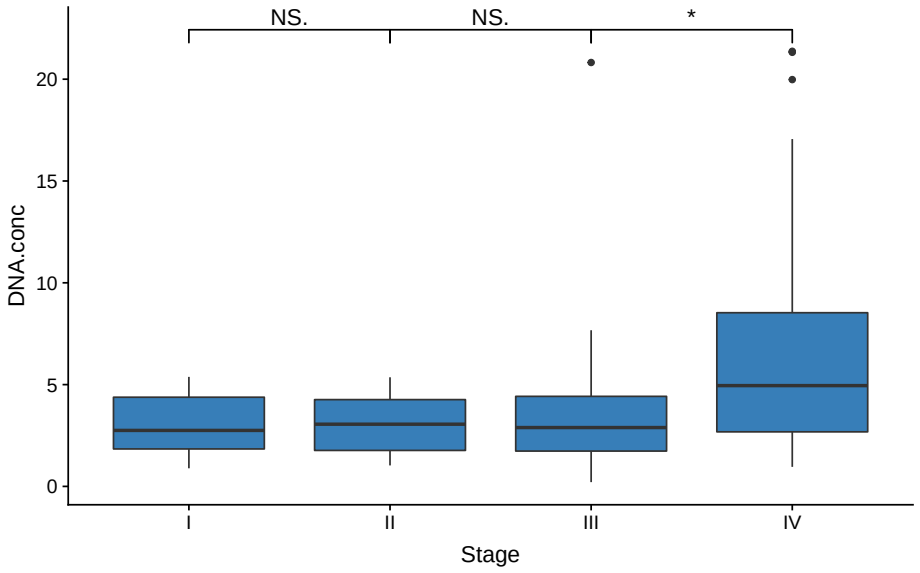
3 MALIBU Biobank statistics

We plot a number of summary statistics for the MALIBU Biobank. We compare levels of cell free DNA that we can extract with tumour stage and various gold standard tumour markers.

```
data('malibu', package='HDLB2018')

## Overview Stages and Location
ggplot (data= malibu, mapping = aes (x=Stage, fill = Tumor)) +
  geom_bar(alpha = 4/5) +
  scale_fill_manual(values=c('#377eb8', '#ff7f00')) +
  xlab('stage')
```

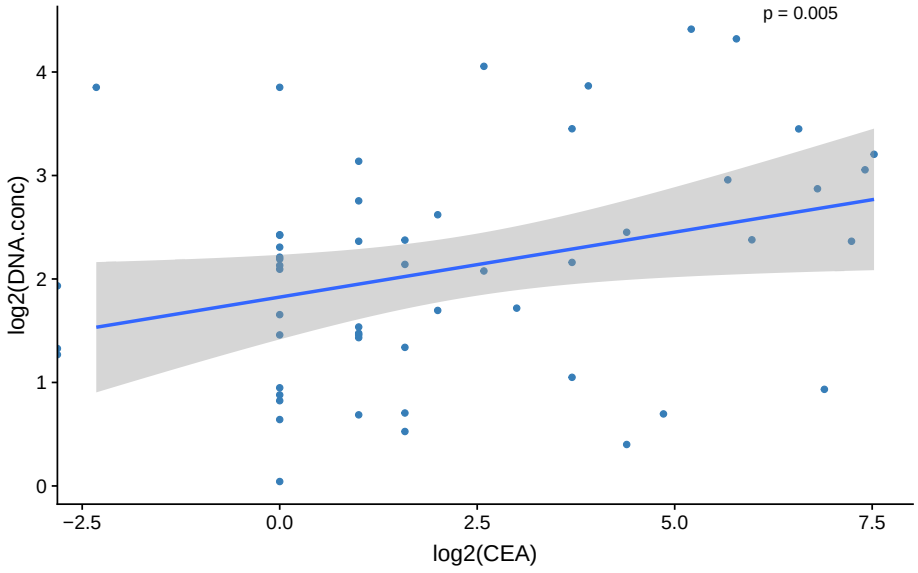



```
## Association DNA-conc and Stage
ggplot(data = malibu %>% dplyr::select(Stage, DNA.conc) %>% drop_na(),
       mapping = aes(x = Stage, y = DNA.conc)) +
  geom_boxplot(fill='#377eb8') +
  geom_signif(comparisons = list(c("I","II"),c("II","III"),c("III","IV")),
              map_signif_level = TRUE, textsize=5)
```

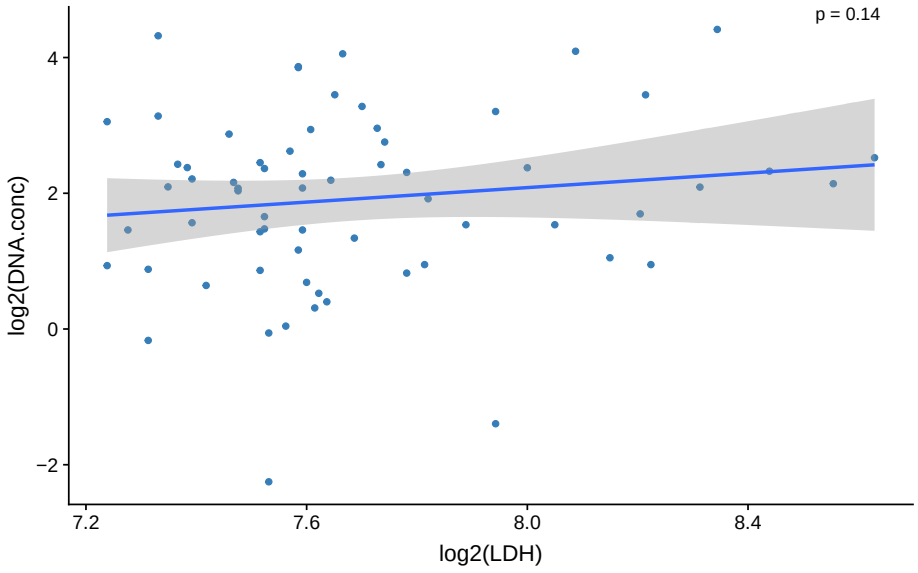



```
##Association DNA-conc and CEA
malibu2 <- filter (malibu, CEA < 200)
pval <- cor.test(log2(malibu2$DNA.conc+1), log2(malibu2$CEA + 1), method = "pearson", conf.level = 0.95, alte
  .$p.value %>% round(3)
pcc <- cor(log2(malibu2$DNA.conc+1), log2(malibu2$CEA + 1), method = "pearson")
ggplot(data = malibu2, mapping = aes(x = log2(CEA), y = log2(DNA.conc))) +
```

```
        geom_point(colour='#377eb8') +
        geom_smooth(method='lm') +
        annotate('text', x=Inf, y=Inf, hjust=2, vjust=1, label=paste('p =', pval))
```




```
##Asscociation DNA-conc and LDH
malibu3 <- filter (malibu, LDH <500)
pval <- cor.test(malibu3$DNA.conc, malibu3$LDH, method = "pearson", conf.level = 0.95, alternative='greater')
  .$p.value %>% round(2)
pcc <- cor(malibu3$DNA.conc, malibu3$LDH, method = "pearson")
ggplot(data = malibu3, mapping = aes(x = log2(LDH), y = log2(DNA.conc))) +
  geom_point(colour='#377eb8') +
  geom_smooth(method='lm') +
  annotate('text', x=Inf, y=Inf, hjust=1.5, vjust=1, label=paste('p =', pval))
```



4 Annotation of variant counts

First we read the determined read counts for all sequencing files. The files contain for each position of each amplicon the number of reads, the reference base and the number of times each nucleotide (or InDel) was sequenced.

We need to separate information for each nucleotide. Several statistics are provided separated by colons. The first field contains the base, the second one contains the count and the third the average mapping quality. The rest of the statistics are probably not that useful in our case so we discard them.

We need to separate information for each nucleotide. Several statistics are provided separated by colons. The first field contains the base, the second one contains the count and the third the average mapping quality. The rest of the statistics are probably not that useful in our case so we discard them.

```
## separate out the different base frequencies/qualities
base_counts <- base_counts %>% dplyr::select(-dummy) %>%
  gather(var, bf, -(filename:read_count)) %>% drop_na() %>%
  filter(!grepl('^N:', bf)) %>%
  separate(bf, c('nucleotide', 'nuc_count', 'quality', 'rest'),
           sep=':', extra='merge') %>%
  filter(nuc_count > 0) %>%
  arrange(filename, chr, position) %>%
  dplyr::select(-c(var, rest))

## convert numeric columns to proper data type
base_counts <- base_counts %>% mutate(position = as.integer(position),
                                      read_count = as.integer(read_count),
                                      nuc_count = as.integer(nuc_count),
                                      quality = as.numeric(quality))

## fix strange chromosome labels
base_counts <- base_counts %>% mutate(chr = gsub('chr', '', chr)) %>%
  filter(chr %in% c(1:22, c('X', 'Y')))
```

4.1 Annotation of amplicons

Next we need to annotate whether a variant falls into an amplicon. We use the BLAST-derivd mapping of the amplicons to annotate each variant. First we load the amplicon annotation file.

```
## load amplicon locations
data('location1', package='HDLB2018')
```

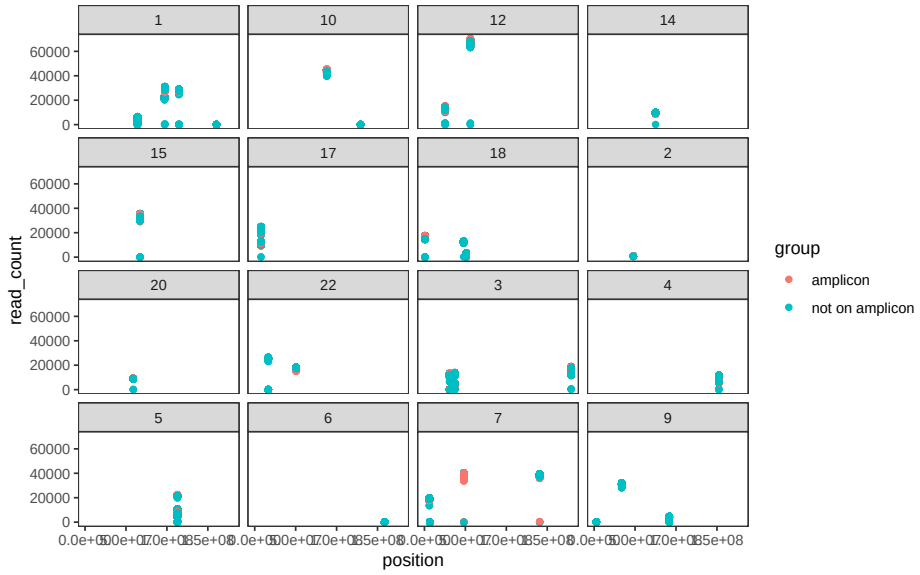
Next we generate genomic ranges for the amplicons and variants.

```
## amplicon genomic ranges
amplicon_gr <- GRanges(
  seqnames=as.character(location1$chr),
  ranges=IRanges(location1$amplicon_start,
```

```
                    end=location1$amplicon_end,
                    names=location1$amplicon),
  strand='+'
)

## variants genomic ranges
variants_gr <- GRanges(
  seqnames=as.character(base_counts$chr),
  ranges=IRanges(base_counts$position,
                 end=base_counts$position),
  strand='+'
)

## overlaps for each variant with amplicon range
var_ol <- findOverlaps(variants_gr, amplicon_gr)

## annotate amplicon in base count object
base_counts$amplicon <- 'none'
base_counts$amplicon[queryHits(var_ol)] <- location1$amplicon[subjectHits(var_ol)]
```

Do variants in general correspond to amplicon positions? How many don't?

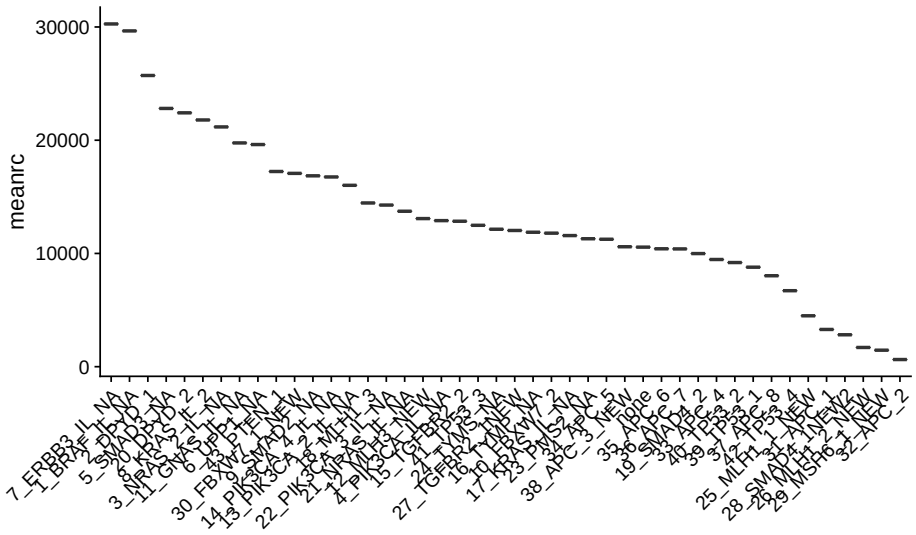
```
base_counts %>% filter(filename == '109-020615_S6_L001_R1_001_variant_counts.txt') %>%
  mutate(group=ifelse(amplicon != 'none', 'amplicon', 'not on amplicon')) %>%
  ggplot(aes(position, read_count)) +
  geom_point(aes(colour=group)) + facet_wrap(~chr) +
  theme_bw() + theme(panel.grid=element_blank())
```



4.2 Summary statistics for amplicon read counts

How well are different amplicons covered by read counts?

```
base_counts %>% group_by(amplicon) %>%
  summarise(meanrc = mean(as.integer(read_count))) %>% ungroup() %>%
  ggplot(aes(factor(amplicon, levels=arrange(., desc(meanrc)) %>%
                      .$amplicon), meanrc)) +
  geom_boxplot() + xlab('') +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



It seems that most amplicons are represented at some level of frequency. There are, however, considerable differenes. Some amplicons suck up a lot of reads and a few seem to not work very well. What is the average mapping quality per hotspot? Are there differences based on the position?

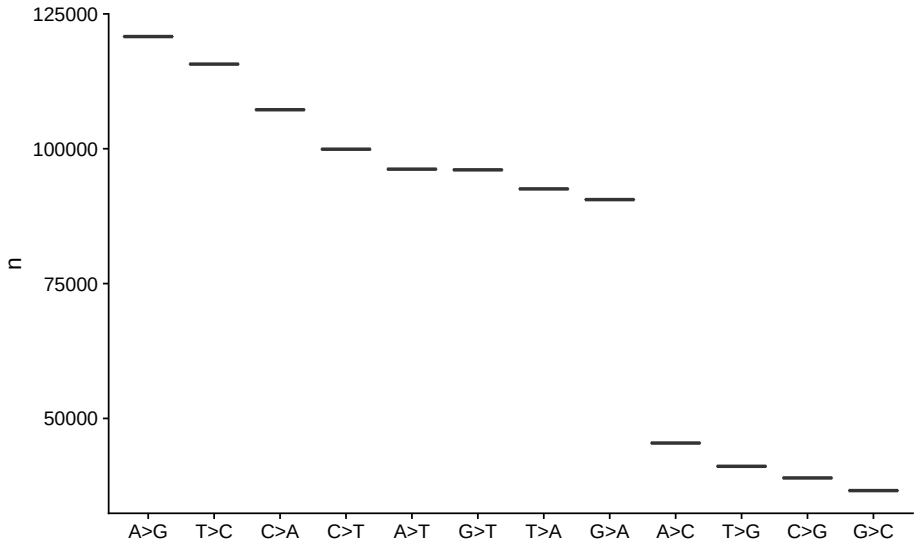
4.3 Calculation of mutations and their frequencies

Next we want to quantify the mutations and their frequencies that we observe for each amplicon.

```
## calculate raw variants, no filtering yet
raw_variants <- base_counts %>% dplyr::select(-quality) %>%
  filter(nucleotide != ref_base, amplicon != 'none') %>%
  arrange(filename, amplicon, position) %>%
  mutate(allele_frequency = as.integer(nuc_count)/read_count)
```

Are different nucleotide changes equally likely?

```
raw_variants %>% filter(!grepl('\\+|\\-', nucleotide)) %>%
  unite(DNA_change, ref_base, nucleotide, sep='>') %>%
  count(DNA_change) %>%
  ggplot(aes(factor(DNA_change,
                    levels=arrange(., desc(n)) %>% .$DNA_change), n)) +
  geom_boxplot() + xlab('')
```



4.4 Annotation of variants

Next we need to annotate the discovered variants. We need to add information about the genomic location of the nucleotide change that can then be mapped to variant annotations.

4.4.1 Annotation of DNA change

We annotate the DNA change ocurring with each genomic variant.

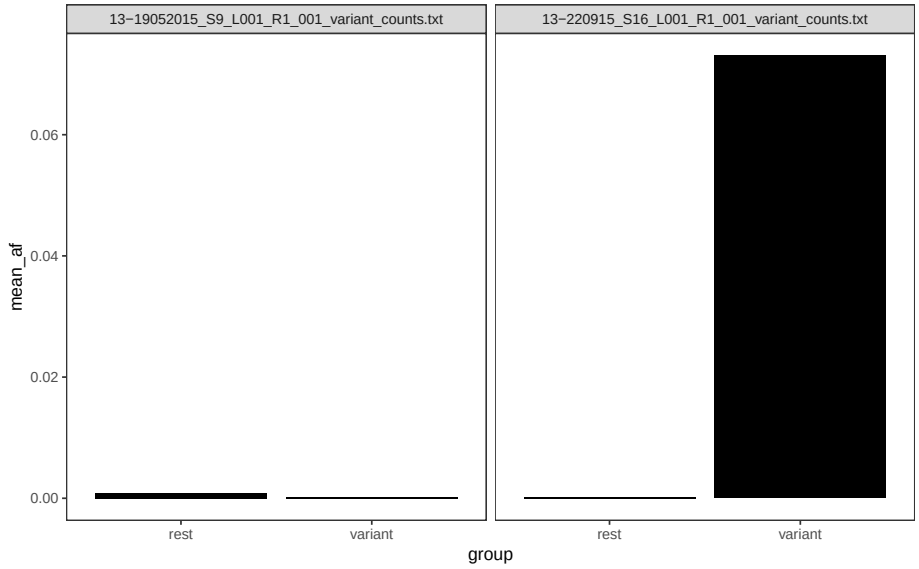
```
## annotate genomic variant location
annotated_variants <- raw_variants %>%
  mutate(DNA_change = ifelse(grepl('\\+', nucleotide),
                             gsub('\\+', 'ins', nucleotide),
                      ifelse(grepl('\\-', nucleotide),
                             gsub('\\-', 'del', nucleotide),
                             paste(ref_base, nucleotide, sep='>'))))
```

4.4.2 Sanity check: known mutations

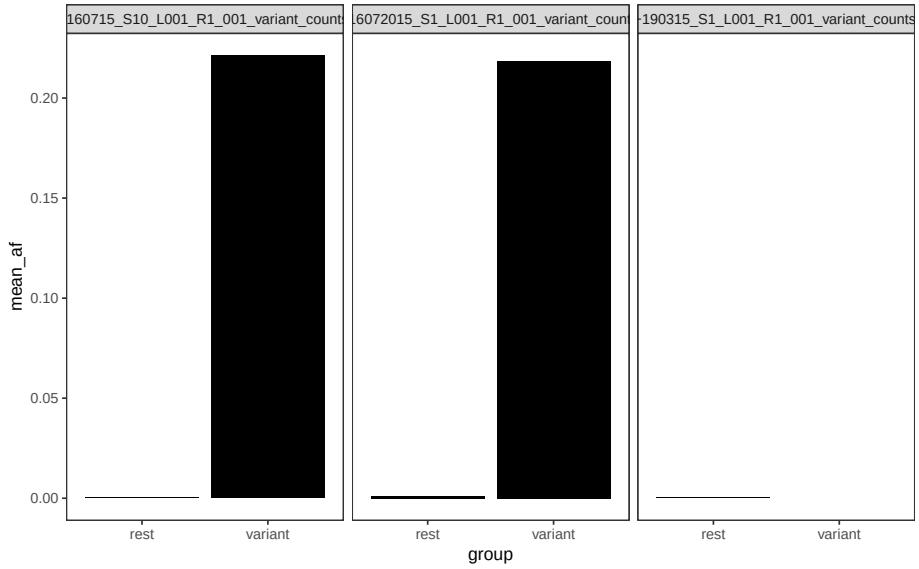
Can we, by manual inspection, see that previously characterized mutations are in fact present in the data?

```
check_variant <- function(patient, chr, pos, change){
  annotated_variants %>% filter(grepl(paste0('^', patient, '-'), filename),
                                chr == as.character(chr)) %>%
    mutate(group = ifelse((position == pos) & (DNA_change == change),
                          'variant', 'rest')) %>%
    group_by(filename, group) %>%
    summarise(mean_af = mean(allele_frequency)) %>% ungroup() %>%
    ggplot(aes(group, mean_af)) +
    geom_bar(stat='identity', fill='black') +
```

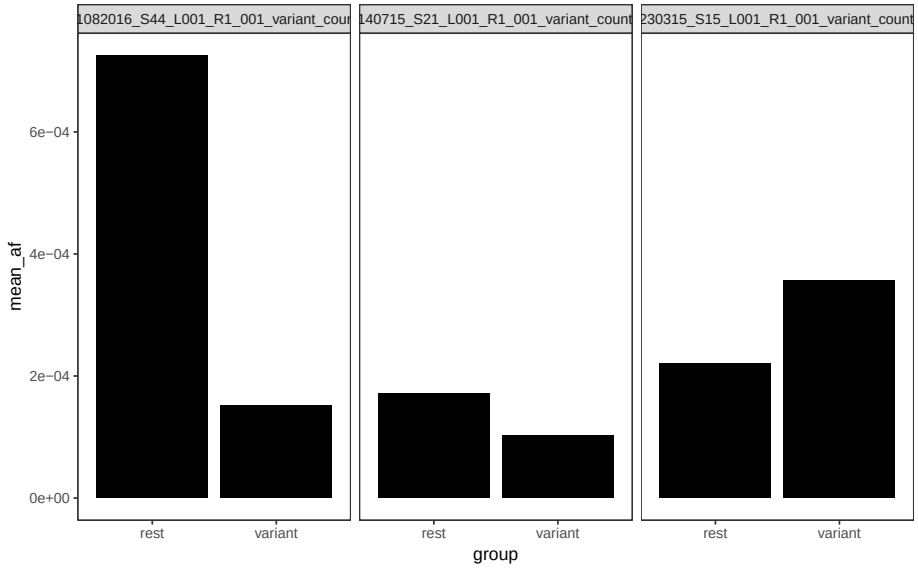
```
        facet_wrap(~filename) + theme_bw() +
        theme(panel.grid=element_blank())
    }

    ## patient 13 has NRAS Q61L (chr1, 114713908 T>A)
    check_variant('13', '1', 114713908, 'T>A')
```



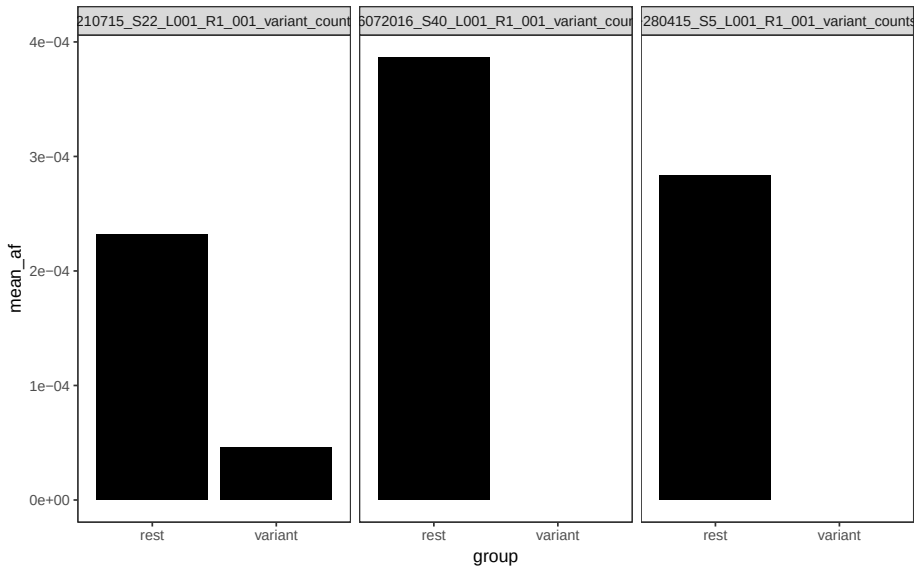

```
    ## patient 52 has KRAS G12D (chr12, 25245350, C>T)
    check_variant('52', '12', 25245350, 'C>T')
```




```
    ## patient 55 has KRAS G12V (chr12, 25245350, C>A)
    check_variant('55', '12', 25245350, 'C>A')
```




```
## patient 80 has KRAS G12D (chr12, 25245350, C>T)
check_variant('80', '12', 25245350, 'C>T')
```



4.4.3 Model-based selection of variants

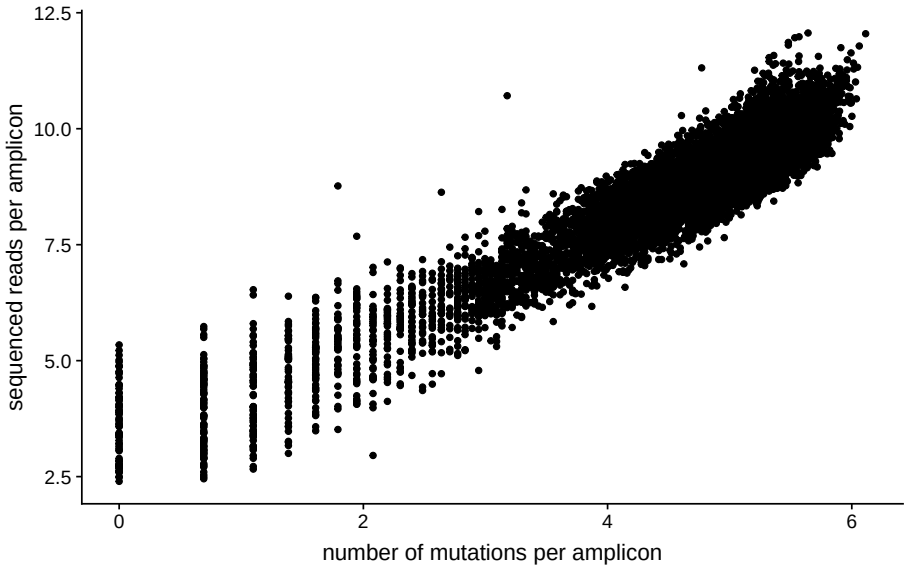
Over time it has become clear to me that the ideal mutation frequency and read-depth cut-offs varies from sample to sample and amplicon to amplicon. I thus think that a model-based approach is the best way to select the DNA changes that significantly separate from the mutation background (noise). We fit a robust linear model that models the allele frequency of each DNA change as a function of the amplicon position. We then select DNA changes that do not fit to the rest by looking at the robustly standardized residuals of each DNA change.

```
filter_vars_rlm <- function(df){
  if(nrow(df) < 10){
```

```
    res <- tibble()
  } else {
    ## robust fit
    fit <- rlm(I(log(allele_frequency/(1-allele_frequency))) ~ as.factor(position) + filename,
               data=mutate(df, allele_frequency = ifelse(allele_frequency == 1,
                                                         0.99, allele_frequency)))
    ## standardized residuals
    s_res <- resid(fit)
    res <- df[s_res > median(s_res) + 3*sd(s_res),]

    ## annotate mutations - sample quality and position bias
    betas <- broom::tidy(fit)
    res <- res %>%
      inner_join(betas %>% filter(grepl('position', term)) %>%
                   mutate(position = as.integer(gsub('^as.factor\\(position\\)', '', term))) %>%
                   dplyr::select(position, position_bias = estimate)) %>%
      inner_join(betas %>% filter(grepl('filename', term)) %>%
        mutate(filename = gsub('^filename', '', term)) %>%
        dplyr::select(filename, sample_bias = estimate))
  }
  return(res)
}

variants_to_keep <- annotated_variants %>%
  distinct(filename, amplicon, position, DNA_change,
           allele_frequency) %>%
  group_by(amplicon) %>% do(results=filter_vars_rlm(.)) %>% ungroup() %>%
  dplyr::select(-amplicon) %>% unnest(results)
```

A number of amplicons did not work for technical reason. We save this information in a variable so we can fall back on this knowledge when trying to understand discrepancies between biologically similar samples. We observe that the number of mutations that we detect per amplicon correlates well with the sequencing depth.

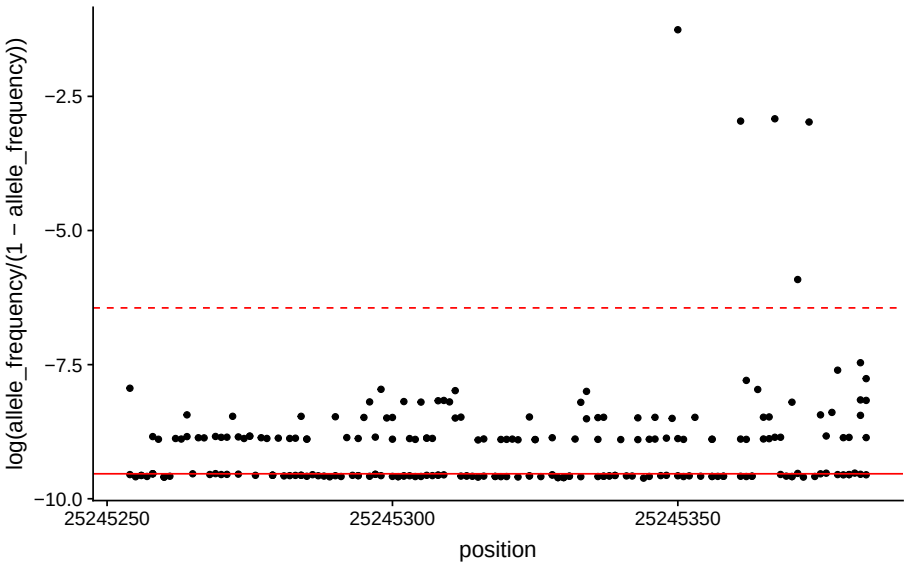
```
## correlation between sequencing depth and the number of mutations found
annotated_variants %>%
    distinct(filename, amplicon, position, DNA_change,
             allele_frequency) %>%
    group_by(filename, amplicon) %>%
  count(filename, amplicon) %>%
  left_join(base_counts %>%
              group_by(filename, amplicon) %>% summarise(read_count = mean(read_count)) %>% ungroup()) %>%
  ggplot(aes(log(n), log(read_count))) + geom_point() +
  xlab('number of mutations per amplicon') +
  ylab('sequenced reads per amplicon')
```




```
annotated_variants %>%
    distinct(filename, amplicon, position, DNA_change,
             allele_frequency) %>%
    group_by(filename, amplicon) %>%
    count(filename, amplicon) %>%
    left_join(base_counts %>%
                  group_by(filename, amplicon) %>%
                summarise(read_count = mean(read_count)) %>% ungroup()) %>%
  ungroup() %>%
  summarise(PCC = cor(log(n), log(read_count), method='pearson'),
            SCC = cor(log(n), log(read_count), method='spearman'))
```

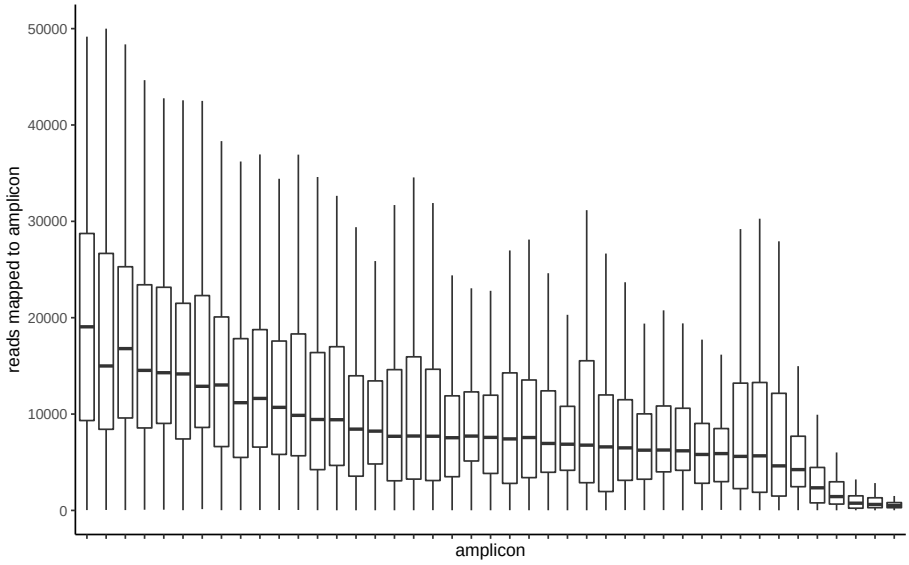
As an example we plot the amplicon for the KRAS G12D mutation for a patient that has a KRAS G12D mutation. A solid red line highlights the median allele frequency of detected variants and a dashed line indicates the cutoff that we inferred for mutation filtering.

```
## KRAS G12D example
kras_amplicon <- annotated_variants %>%
  filter(filename=='52-160715_S10_L001_R1_001_variant_counts.txt',
         amplicon=='17_KRAS_IL_NA') %>%
  arrange(desc(allele_frequency))

kras_fit <- rlm(I(log(allele_frequency/(1-allele_frequency))) ~ position, data = kras_amplicon)
kras_med <- median(log(kras_amplicon$allele_frequency/(1-kras_amplicon$allele_frequency)))
kras_cutoff <- kras_med + (3*sd(resid(kras_fit)))
kras_amplicon %>%
  ggplot(aes(position, log(allele_frequency/(1-allele_frequency)))) +
  geom_point() +
  geom_hline(yintercept = kras_med, colour = 'red') +
  geom_hline(yintercept = kras_cutoff, colour = 'red', linetype = 'dashed')
```



We visualize the number of reads mapped to each amplicon as a box plot. We further calculate a number of summary statistics.

```
## distribution of read counts across amplicons
av_df <- annotated_variants %>%
  group_by(filename, amplicon) %>%
  summarise(read_count = mean(read_count)) %>% ungroup() %>%
  group_by(amplicon) %>%
  mutate(med_count=median(read_count)) %>% ungroup() %>%
  mutate(amplicon = factor(amplicon,
                           levels=arrange(., desc(med_count)) %>%
                             .$amplicon %>% unique))
## box plot
av_df %>%
  ggplot(aes(amplicon, read_count)) +
  geom_boxplot(outlier.colour = NA) +
  theme_classic() + ylim(c(0,50000)) +
  theme(axis.text.x = element_blank()) +
  ylab('reads mapped to amplicon')
```




```
## summary statistics
av_df %>% group_by(amplicon) %>%
  summarise(med_read_count = median(read_count)) %>% ungroup() %>%
  summarise(var = var(med_read_count),
            min = min(med_read_count),
            max = max(med_read_count),
            med = median(med_read_count))
```

In general we find that our ability to detect variants for amplicons directly depends on the number of sequencing reads that we get for each of the amplicons.

4.4.4 Database annotation of variants

We now have all information that we need to use ENSEMBL/COSMIC to annotate the variants that we find in our data. For this we use the exported data from the COSMIC database (all primary tumour sample data). In this vignette we only include a small subset of the COSMIC database that corresponds to variants covered by our amplicons.

```
## load data
data('cosmic_mutant_subset', package='HDLB2018')

## annotate variants
annotated_variants <- annotated_variants %>%
  inner_join(variants_to_keep %>% filter(allele_frequency > 0.005)) %>%
  mutate(chr=as.character(chr)) %>%
  dplyr::select(variant_pos = position, everything()) %>%
  left_join(cosmic_mutant_subset) %>%
  distinct()
```

In the next steps we resolve the correct/exact variant that we measure at each amplicon position. We begin with single nucleotide polymorphisms.

```
## resolve point mutations
point_mutations <- annotated_variants %>%
  filter(description %in% c('Substitution - Missense',
                            'Substitution - coding silent',
                            'Substitution - Nonsense')) %>%
  separate(DNA_change_COSMIC, c('change_pos', 'new_base'),
           sep='>', remove=F) %>%
  mutate(new_base=substr(new_base, 1, 1),
         new_base = ifelse(strand == '-', system('tr "ACGT" "TGCA"', input=new_base, intern=T), new_base),
         change_pos=gsub('_\\d+', '', gsub('^c\\.\\d+', '', change_pos))) %>%
  filter(nucleotide == new_base, nchar(change_pos) == 1) %>%
  dplyr::select(-c(change_pos, new_base)) %>%
  ## exclude silent mutations - we don't care about them
  filter(description != 'Substitution - coding silent',
         !grepl('_ENST', gene_symbol))
```

We move on to insertion/deletion events.

```
## insertion/deletion events
indels <- annotated_variants %>%
    filter(grepl('del|ins', DNA_change)) %>%
    filter(description %in% c('Deletion - Frameshift',
                              'Deletion - In frame',
                              'Frameshift',
                              'Insertion - Frameshift',
                              'Insertion - In frame')) %>%
    extract(DNA_change_COSMIC, c('tag', 'new_base', 'tag2'),
            regex='(.+)((ins|del).+$)', remove=F) %>%
    mutate(new_base = ifelse(strand == '-', system('tr "ACGT" "TGCA"', input=new_base, intern=T), new_base)) %>%
    filter(DNA_change == new_base, !grepl('_ENST', gene_symbol)) %>%
    dplyr::select(-c(tag, tag2, new_base))
```

Now we can combine these variants.

```
## combine point mutations and indels
filtered_variants1 <- bind_rows(point_mutations, indels) %>% distinct()
```

5 Filtering determined variants

There are a number of germline mutations that come up. We check for these manually in the 1000 genome variation browser and exclude them from downstream analayses if there is sufficient cause.

```
germline <- tibble(
  amplicon = c('25_MLH1_1_NEW', '23_PMS2_NA'),
  variant_pos = c(37012077, 5987077),
  nucleotide = c('G', 'A')
)
```

```
## remove them from the data
filtered_variants <- filtered_variants1 %>% anti_join(germline)
```

5.1 Surprisingly frequent substitutions

A few SNPS in the genes such as TGFBR2 and MLH3 occur much more frequently than expected (more often than TP53/KRAS/APC). A closer look at these reveals that the nucleotide changes all occur within poly-X regions, strongly suggesting sequencing artefacts to be the cause for these findings. We hence remove them from the results.

```
## get the DNA sequence surrounding each mutation position
mutation_context <- filtered_variants %>%
  distinct(filename, amplicon, variant_pos, nucleotide, chr) %>%
  count(chr, amplicon, variant_pos, nucleotide) %>%
  arrange(desc(n)) %>% rowwise() %>%
  mutate(context = as.character(getSeq(Hsapiens, paste0('chr', chr),
                                       variant_pos-4, variant_pos+4))) %>%
  ungroup() %>%
  inner_join(filtered_variants %>% distinct(amplicon, chr, variant_pos, position_bias)) %>%
  left_join(filtered_variants %>% filter(grepl('^hct', filename)) %>%
              count(amplicon, variant_pos, DNA_change) %>%
             arrange(desc(n)) %>%
             dplyr::select(cell_line_n = n, everything()) %>%
             dplyr::select(-DNA_change))

exceptions <- mutation_context[c(1:10),] %>%
  bind_rows(mutation_context %>% filter(position_bias > 0.5)) %>%
  dplyr::select(-c(n, context))

filtered_variants <- filtered_variants %>% anti_join(exceptions)
```

We find two mutations that are related to a specific sequencing batch, ie they occur in every sample of this batch but not in any of the others. These mutations are APC P870S und Q1294*. We exclude these from downstream analyses as we find it most likely that they are technical artefacts.

```
filtered_variants <- filtered_variants %>%
  filter(! (gene_symbol == 'APC' & (AA_change %in% c('p.P870S', 'p.Q1294*'))))
```

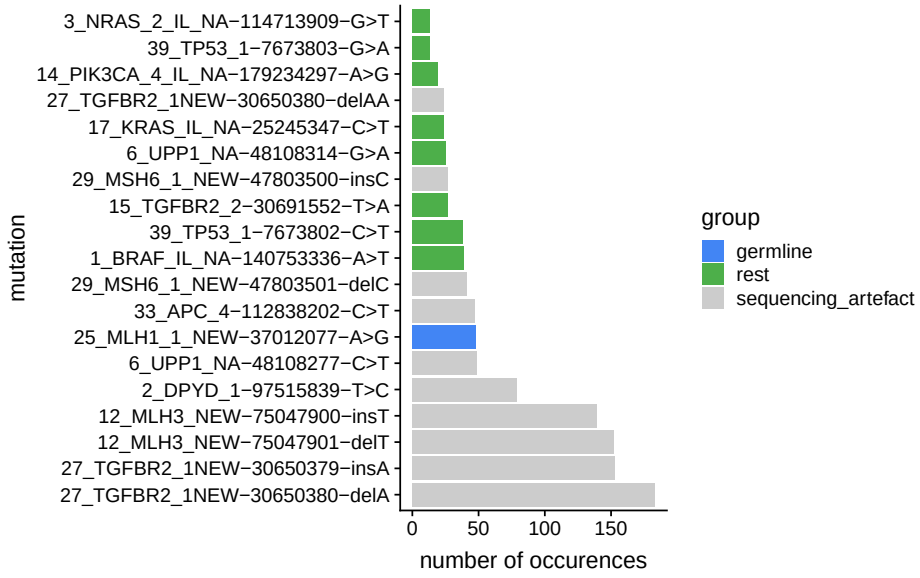
5.2 Statistical evaluation of resulting mutations

We perform a number of plots to get an overview of the mutations that we find and how frequently we find them. We also examine mutations that we might not be able to find to improve our understanding of the results that we find or might not find.

5.2.1 Most frequent mutations

First we generate a bar plot of the most frequent mutations in our datset.

```
filtered_variants1 %>% inner_join(exceptions) %>%
  mutate(group = 'sequencing_artefact') %>%
  bind_rows(filtered_variants1 %>% inner_join(germline) %>%
              mutate(group = 'germline')) %>%
  bind_rows(filtered_variants %>% mutate(group = 'rest')) %>%
  distinct(filename, amplicon, variant_pos, DNA_change, group) %>%
  count(amplicon, variant_pos, DNA_change, group) %>% arrange(desc(n)) %>%
  unite(mutation, amplicon, variant_pos, DNA_change, sep='-') %>%
  dplyr::slice(1:20) %>% dplyr::slice(-8) %>%
  mutate(mutation = factor(mutation, levels=mutation)) %>%
  ggplot(aes(mutation, n, fill=group)) + geom_bar(stat='identity') +
  coord_flip() + ylab('number of occurences') +
  scale_fill_manual(values=c('#4285f4', '#4daf4a', '#cccccc'))
```



5.2.2 What about APC?

It might be of note that among the most frequent mutations there is no APC mutation. As these are the most frequent mutations in colorectal cancer we need to look into this a bit. First we plot what fraction of all patients show an APC mutation.

```
## use the histo samples
n_histo <- filtered_variants %>% filter(grepl('h-', filename)) %>%
  .$filename %>% unique

## percentage of histo samples where an APC mutation is found
filtered_variants %>% filter(grepl('h-', filename), gene_symbol == 'APC') %>%
  count(filename) %>% summarise(n() / length(n_histo)) %>% .[[1]]
```

How many of the known APC mutations in COSMIC can we actually cover using the hotspots in our amplicon panel?

```
## filter hotspot genes
data('apc_loc', package='HDLB2018')

## ranges of hotspot
apc_hs_ranges <- GRanges(
  seqnames=as.character(apc_loc$chr),
  ranges=IRanges(as.integer(ifelse(apc_loc$strand == 'plus',
                                   apc_loc$start, apc_loc$end)),
                 end=as.integer(ifelse(apc_loc$strand == 'plus',
                                       apc_loc$end, apc_loc$start)),
                 names=apc_loc$hotspot),
  strand='+'
)
```

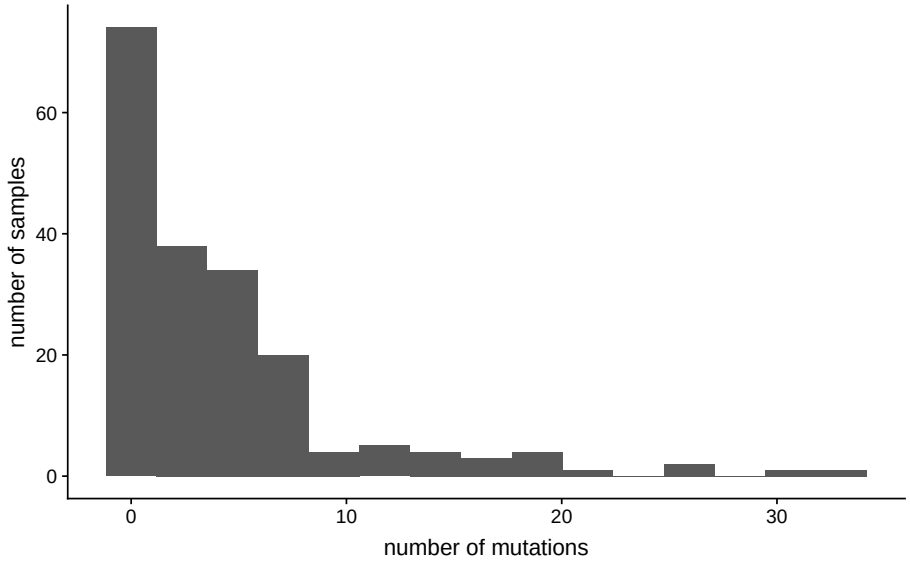
We generate a table with all APC mutations found in our data so we can generate a plot using cBioPortal.

```
filtered_variants %>% filter(gene_symbol == 'APC') %>%
  dplyr::select(gene_symbol, AA_change, filename, description,
                chr, variant_pos, ref_base, nucleotide) %>%
  `colnames<-`(c('Hugo_Symbol', 'Protein_Change', 'Sample_ID',
                 'Mutation_Type', 'Chromosome', 'Start_Position',
                 'Reference_Allele', 'Variant_Allele')) %>%
  mutate(End_Position = Start_Position,
         Mutation_Status = 'Somatic',
         Protein_Change = gsub('p\\.', '', Protein_Change)) %>%
  write_tsv('APC_mut_data.tsv')
```

5.2.3 How many mutations are found per sample?

We check how many mutations in total were found for each sample.

```
mut_counts_per_sample <- filtered_variants %>%
  distinct(filename, amplicon, variant_pos, DNA_change) %>%
  count(filename) %>% arrange(desc(n)) %>%
  bind_rows(raw_variants %>%
              filter(!filename %in% filtered_variants$filename) %>%
              distinct(filename) %>% mutate(n = 0))

## first a histogram
ggplot(mut_counts_per_sample, aes(n)) + geom_histogram(bins=15) +
  xlab('number of mutations') + ylab('number of samples')
```



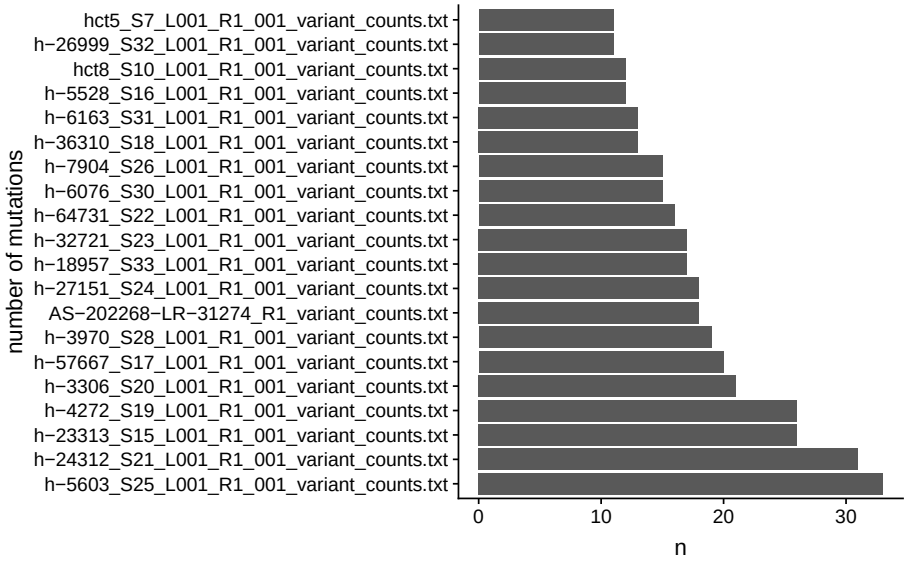
Which are the samples without any detectable mutations?

```
raw_variants %>% filter(!filename %in% filtered_variants$filename) %>%
  .$filename %>% unique
#>  [1] "109-020615_S6_L001_R1_001_variant_counts.txt"
#>  [2] "133-24092015_S24_L001_R1_001_variant_counts.txt"
#>  [3] "143-08072015_S27_L001_R1_001_variant_counts.txt"
#>  [4] "144-10022016_S11_L001_R1_001_variant_counts.txt"
#>  [5] "144-20042016_S18_L001_R1_001_variant_counts.txt"
#>  [6] "152-03022017_S39_L001_R1_001_variant_counts.txt"
#>  [7] "152-04032016_S19_L001_R1_001_variant_counts.txt"
#>  [8] "152-140715_S7_L001_R1_001_variant_counts.txt"
#>  [9] "167-28092015_S13_L001_R1_001_variant_counts.txt"
#> [10] "168-02112015_S7_L001_R1_001_variant_counts.txt"
#> [11] "168-210915_S13_L001_R1_001_variant_counts.txt"
#> [12] "168-29112016_S34_L001_R1_001_variant_counts.txt"
#> [13] "168-30082016_S35_L001_R1_001_variant_counts.txt"
#> [14] "19-210515_S17_L001_R1_001_variant_counts.txt"
#> [15] "25-120315_S14_L001_R1_001_variant_counts.txt"
#> [16] "330-13072016_S31_L001_R1_001_variant_counts.txt"
#> [17] "38a-27042016_S10_L001_R1_001_variant_counts.txt"
#> [18] "427b-14062016_S33_L001_R1_001_variant_counts.txt"
#> [19] "430-08082016_S17_L001_R1_001_variant_counts.txt"
#> [20] "430-16062016_S15_L001_R1_001_variant_counts.txt"
#> [21] "44-070515_S3_L001_R1_001_variant_counts.txt"
#> [22] "49-170315_S4_L001_R1_001_variant_counts.txt"
#> [23] "55-01082016_S44_L001_R1_001_variant_counts.txt"
#> [24] "55-140715_S21_L001_R1_001_variant_counts.txt"
#> [25] "80-280415_S5_L001_R1_001_variant_counts.txt"
#> [26] "83-090715_S23_L001_R1_001_variant_counts.txt"
#> [27] "AS-202266-LR-31274_R1_variant_counts.txt"
#> [28] "AS-202270-LR-31274_R1_variant_counts.txt"
#> [29] "AS-202282-LR-31274_R1_variant_counts.txt"
#> [30] "AS-202286-LR-31274_R1_variant_counts.txt"
```

```
#> [31] "AS-202292-LR-31274_R1_variant_counts.txt"
#> [32] "AS-202300-LR-31274_R1_variant_counts.txt"
#> [33] "AS-202324-LR-31274_R1_variant_counts.txt"
#> [34] "AS-202332-LR-31274_R1_variant_counts.txt"
#> [35] "AS-202340-LR-31274_R1_variant_counts.txt"
#> [36] "AS-202346-LR-31274_R1_variant_counts.txt"
#> [37] "AS-202348-LR-31274_R1_variant_counts.txt"
#> [38] "AS-202369-LR-31372_R1_variant_counts.txt"
#> [39] "AS-202371-LR-31372_R1_variant_counts.txt"
#> [40] "AS-202383-LR-31372_R1_variant_counts.txt"
#> [41] "AS-202425-LR-31372_R1_variant_counts.txt"
```

Most samples have from 1 - 10 mutations, which seems to be a reasonable number. However, there are other samples with many more mutations detected. One sample has as many as ~60 estimated mutations. We draw a bar plot with all samples with > 10 mutations.

```
mut_counts_per_sample %>% filter(n > 10) %>%
  filter(!grepl('Undetermined', filename)) %>%
  mutate(filename = factor(filename, levels=filename)) %>%
  ggplot(aes(filename, n)) + geom_bar(stat='identity') +
  coord_flip() + xlab('number of mutations')
```



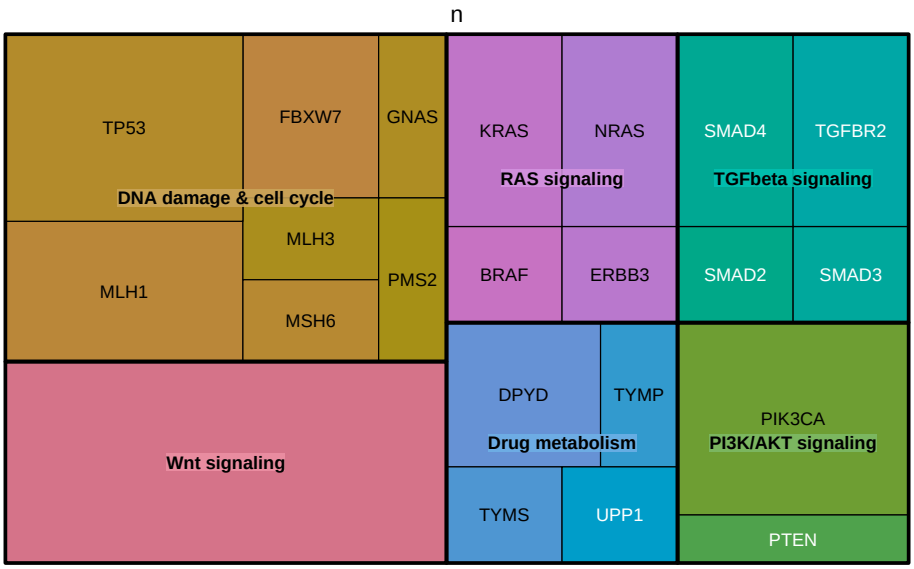
The most highly mutated samples are those derrived from histology slides. We hyptohesize that due to the high sequencing depth we can sequence many tumour sub-clones with heterogenous mutations that are present at low frequency.

5.2.4 A tree map visualizes amplicon coverage

We draw a tree map that shows how many amplicons are present for each gene in our panel.

```
amplicon_count <- annotated_variants %>%
  distinct(gene_symbol, amplicon) %>%
  filter(!is.na(gene_symbol),
```

```
            !grepl('_ENST.+$', gene_symbol),
            !grepl('_NM_', gene_symbol)) %>%
  count(gene_symbol) %>% arrange(desc(n)) %>%
  ## annotate pathways
  mutate(pathway = ifelse(gene_symbol %in% c('NRAS', 'KRAS', 'BRAF', 'ERBB3'), 'RAS signaling',
                   ifelse(gene_symbol %in% c('APC'), 'Wnt signaling',
                   ifelse(gene_symbol %in% c('PIK3CA', 'PTEN'), 'PI3K/AKT signaling',
                   ifelse(gene_symbol %in% c('SMAD4', 'SMAD3', 'SMAD2', 'TGFBR2'), 'TGFbeta signaling',
                   ifelse(gene_symbol %in% c('DPYD', 'UPP1', 'TYMP', 'TYMS'), 'Drug metabolism', 'DNA damage

treemap(amplicon_count, index=c('pathway', 'gene_symbol'), vSize='n', type='index')
```



6 Annotating patient information and sample data

For many of the older samples this information is contained in the file names. The new samples, however, have no interpretable file names and hence we need to annotate them based on a separate annotation file.

```
data('sample_info', package='HDLB2018')
```

6.1 Split variant data into subsets

It is more convenient to work if patient samples, cell line samples and histology samples are kept in separate data frames.

```
## remove the 'p.' in front of each AA change
filtered_variants <- filtered_variants %>%
  mutate(AA_change = gsub('^p.', '', AA_change)) %>%
  filter(!grepl('Undetermined', filename))
```

```
## patient samples
### first the ones where file names do not begin with 'AS'
patient_samples_part1 <- filtered_variants %>% filter(!grepl('^h|^AS', filename)) %>%
  separate(filename, c('patient_nr', 'tag'), sep='-', remove=F) %>%
  separate(tag, c('date', 'tag'), sep='_', extra='merge') %>%
  dplyr::select(-tag) %>% mutate(date=as.Date(date, '%d%m%Y')) %>%
  mutate(date=gsub('^00', '20', date))
### now the ones that do begin with 'AS'
patient_samples_part2 <- filtered_variants %>% filter(grepl('^AS', filename)) %>%
  inner_join(sample_info %>%
               mutate(filename = gsub('.fastq.gz', '_variant_counts.txt',
                                      FASTQ_FILE)) %>%
               dplyr::select(filename, patient_nr, date) %>%
               filter(!grepl('hct', patient_nr)))
### combine them
patient_samples <- bind_rows(patient_samples_part1, patient_samples_part2)
patient_samples[grepl('\\.', patient_samples$date),]$date <- as.character(as.Date(patient_samples[grepl('\\.
## adjust patient nr
patient_samples <- patient_samples %>%
  mutate(patient_nr = gsub('_2|dil|a|b', '', patient_nr))

## histo samples
histology_samples <- filtered_variants %>% filter(grepl('^h-', filename)) %>%
  separate(filename, c('hist_id', 'tag'), sep='_', remove=F, extra='merge') %>%
  mutate(hist_id = gsub('-', '_', hist_id)) %>% dplyr::select(-tag)

## cell line samples
### file names do not begin with 'AS'
cl_samples_part1 <- filtered_variants %>% filter(grepl('^hct', filename)) %>%
    separate(filename, c('sample_name', 'tag'), sep='_', remove=F, extra='merge') %>%
    dplyr::select(-tag) %>% mutate(batch = 'batch_1')
### file names do begin with 'AS'
cl_samples_part2 <- filtered_variants %>% filter(grepl('^AS', filename)) %>%
  inner_join(sample_info %>%
               mutate(filename = gsub('.fastq.gz', '_variant_counts.txt',
                                      FASTQ_FILE)) %>%
               dplyr::select(filename, sample_name=patient_nr, date) %>%
               filter(grepl('hct', sample_name))) %>%
  mutate(batch = 'batch_2')

### combine
cl_samples <- bind_rows(cl_samples_part1, cl_samples_part2)
```

6.1.1 Reproducibility

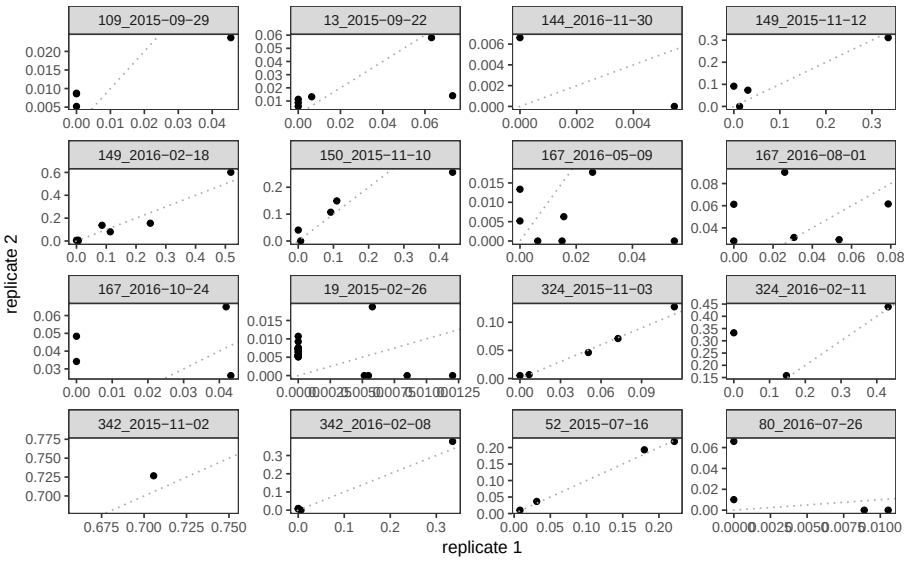
We have performed biological replicates for a number of samples. We can use these samples to judge the reproducibility of liquid biopsies calculating correlation coefficients for replicate pairs.

We first select the patient samples that have replicates.

```
## select samples with replicates
has_reps <- patient_samples %>%
  distinct(patient_nr, filename, date) %>%
  arrange(patient_nr, date) %>%
  count(patient_nr, date) %>% arrange(desc(n)) %>%
  filter(n >= 2) %>% dplyr::select(-n)

## annotation replication number
has_reps <- patient_samples %>% inner_join(has_reps) %>%
  distinct(patient_nr, filename, date) %>%
  group_by(patient_nr, date) %>% mutate(rep = paste0('rep', 1:n())) %>%
  ungroup() %>% arrange(patient_nr, date)
```

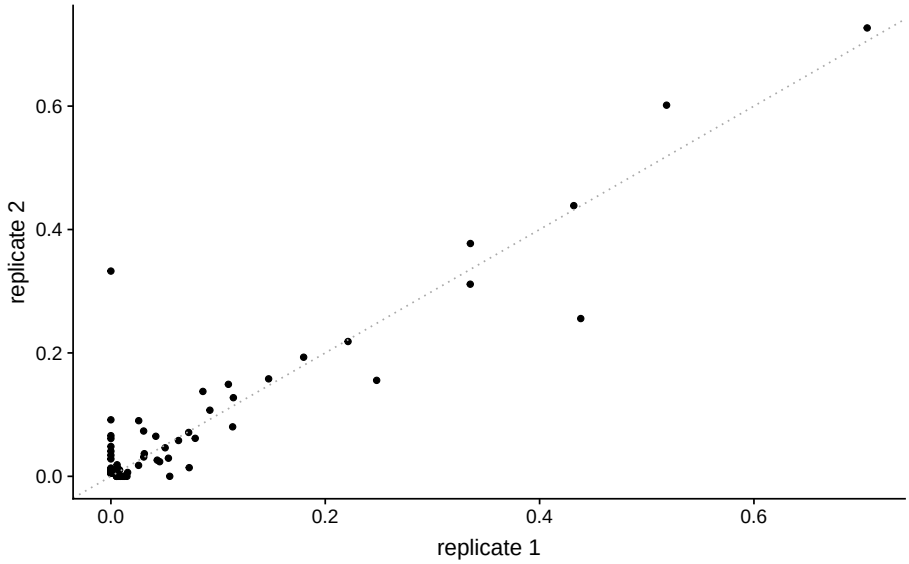
Now we can investigate how well reprducible the signal (mutations) is across replicates.

```
## data frame containing a table of allele frequencies across replicates
rep_af <- patient_samples %>% inner_join(has_reps) %>%
  dplyr::select(amplicon, variant_pos, DNA_change, patient_nr,
                date, rep, allele_frequency) %>%
  distinct() %>% unite(sample, patient_nr, date, sep='_') %>%
  spread(rep, allele_frequency) %>%
  mutate_at(vars(rep1, rep2, rep3), funs(ifelse(is.na(.), 0, .)))

## scatter plot of replicates
rep_af %>% ggplot(aes(rep1, rep2)) + geom_point() +
  geom_abline(slope=1, linetype='dotted', color='#aaaaaa') +
  facet_wrap(~sample, scales='free') + theme_bw() +
  theme(panel.grid=element_blank()) +
  xlab('replicate 1') + ylab('replicate 2')
```




```
## scatter plot of replicates all samples
rep_af %>% ggplot(aes(rep1, rep2)) + geom_point() +
```

```
  geom_abline(slope=1, linetype='dotted', color='#aaaaaa') +
  theme(panel.grid=element_blank()) +
  xlab('replicate 1') + ylab('replicate 2')
```




```
rep_af %>% summarise(PCC = cor(rep1, rep2, method='pearson'),
                     SCC = cor(rep1, rep2, method='spearman'))
```

It seems that there is in fact robust signal in the data. However, there is also noise. An important observation is that there are dropouts, i.e. we sometimes do not detect likely true mutations - likely due to insufficient read coverage. This needs to be taken into account when interpreting the results.

6.2 Treatment and tumour progress

For each patient we know what treatment they received and what the progress of their tumour was (according to tumour imaging) at any given sampling time point.

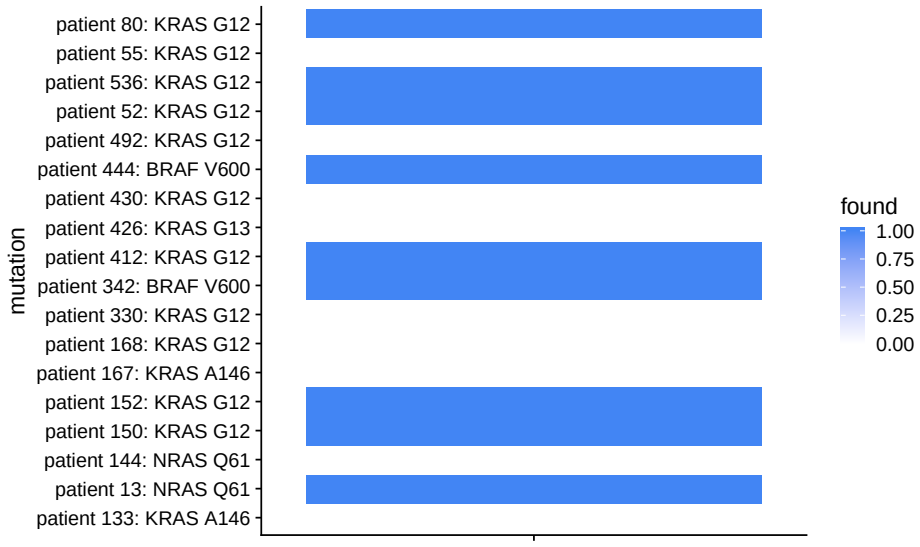
```
## mutations known by Sanger sequencing
data('known_mutations', package='HDLB2018')

## treatment that each patient received
data('patient_treatment', package='HDLB2018')

## tumour progress at sampling time points
data('tumour_progress', package='HDLB2018')

## histo sample to patient nr
data('histo_map', package='HDLB2018')
```

7 Quality control

7.1 Identification of mutations in LB samples known by Sanger-sequencing

Applying the cut-offs determined above, we want to investigate how well we can recover known mutations determined by Sanger sequencing.

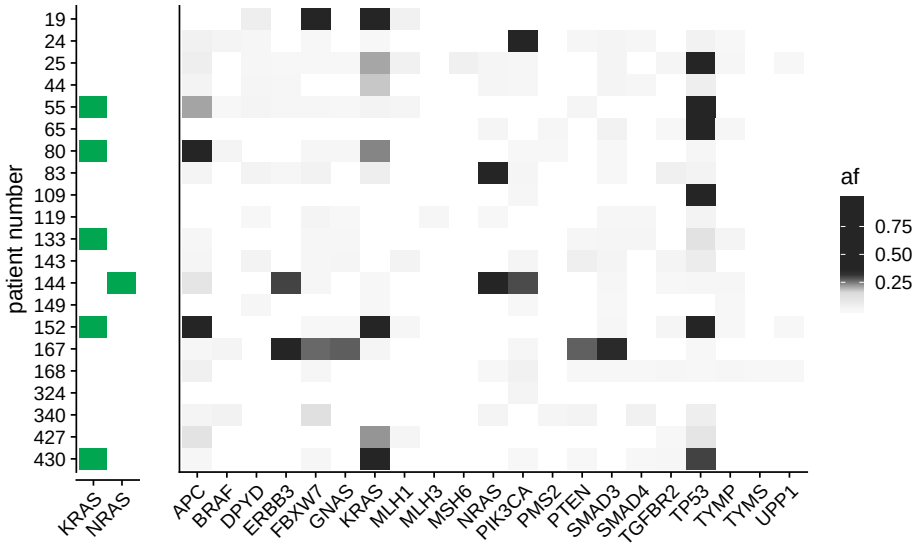
```
## convenience functon removing the last char
cut_last <- function(str){
  return(substr(str, 1, nchar(str)-1))
}

## filter patient sample variants and check if we find the sanger mutations
known_mutations %>% mutate(AA_change = cut_last(AA_change)) %>%
  mutate(patient_nr = as.character(patient_nr)) %>%
  left_join(patient_samples %>%
              dplyr::select(patient_nr, date, gene_symbol, AA_change) %>%
              mutate(found=1, AA_change = gsub('^p.', '', AA_change),
                     AA_change = cut_last(AA_change))) %>%
  mutate(found = ifelse(is.na(found), 0, found),
         patient=paste0('patient ', patient_nr, ':'),
         x=as.factor(1)) %>%
  unite(mutation, patient, gene_symbol, AA_change, sep=' ') %>%
  ggplot(aes(x, mutation, fill=found)) +
  geom_raster() +
  xlab('') + theme(axis.text.x = element_blank()) +
  scale_fill_gradientn(colors=c('#ffffff', '#4285f4'))
```



7.2 Comaparing histology samples and Sanger sequencing

Some mutations are known by Sanger sequencing. We should be able to recover these in available histology samples.

```
hist_muts <- histology_samples %>% inner_join(histo_map) %>%
  filter(!grepl('_NM_', gene_symbol)) %>%
  group_by(patient_nr, hist_id, gene_symbol) %>%
  summarise(af = max(allele_frequency)) %>% ungroup() %>%
  mutate(patient_nr = factor(patient_nr,
                             levels=rev(sort(unique(patient_nr))))) %>%
  ggplot(aes(gene_symbol, patient_nr, fill=af)) +
  geom_tile() +
  scale_fill_gradientn(colours = c('#f7f7f7',
                                   '#d9d9d9','#252525', '#252525',
                                   '#252525','#252525','#252525')) +
  theme(axis.text.x = element_text(angle=45, hjust=1),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        plot.margin = unit(c(0, 0, 0, 0), "cm")) +
  xlab('') + ylab('')

## sanger mutations
sanger_mut <- known_mutations %>% filter(! patient_nr %in% (167:168)) %>%
  filter(patient_nr %in% histo_map$patient_nr) %>%
    mutate(has = 1) %>% right_join(histo_map) %>%
    mutate(gene_symbol = ifelse(is.na(gene_symbol), 'KRAS', gene_symbol),
           has = as.factor(ifelse(is.na(has), 0, has)),
           patient_nr = factor(patient_nr,
                               levels = sort(histo_map$patient_nr,
                                             decreasing=T))) %>%
  dplyr::select(-c(hist_id, AA_change)) %>%
  spread(gene_symbol, has)

sanger_mut[is.na(sanger_mut)] <- '0'
sanger_plot <- sanger_mut %>% gather(gene_symbol, has, -patient_nr) %>%
  ggplot(aes(gene_symbol, patient_nr, fill=as.factor(has))) +
  geom_tile() + scale_fill_manual(values = c('#ffffff', '#00A650')) +
  theme(legend.position = 'none',
        axis.text.x = element_text(angle=45, hjust=1),
        plot.margin = unit(c(0, 0, 0, 0), "cm")) + xlab('') +
  ylab('patient number')

## generate plot
sanger_plot + hist_muts + plot_layout(widths=c(0.1, 1))
```



7.3 Identification of known mutations in cell line samples

The HT29 and HCT116 colorectal cancer cell lines contains several mutations in our hotspots that we know should be there. We observe how well we can detect these variants using the liquid biopsy protocol. First we parse COSMIC mutant data and select all variants in HT29 that should overlap with our sequencing hotspots. Then we assess how well we can detect these SNPs in the liquid biopsy experiment.

```
## read COSMIC mutant export containg HT29/HCT116 variants
data('ht29_variants', package='HDLB2018')
```

Some mutations won't be found because they are not covered by our hotspots.

```
## filter hotspot genes
data('hs_loc', package='HDLB2018')

## ranges of hotspot
hs_ranges <- GRanges(
  seqnames=as.character(hs_loc$chr),
  ranges=IRanges(as.integer(ifelse(hs_loc$strand == 'plus', hs_loc$start, hs_loc$end)),
                 end=as.integer(ifelse(hs_loc$strand == 'plus', hs_loc$end, hs_loc$start)),
                 names=hs_loc$hotspot),
  strand='+'
)
## ranges of HT29 variants
ht29_ranges <- GRanges(
  seqnames=ht29_variants$chr,
  ranges=IRanges(as.integer(ht29_variants$start), end=as.integer(ht29_variants$end),
                 names=ht29_variants$symbol),
  strand='+'
)

ol <- findOverlaps(ht29_ranges, hs_ranges)
```

```
## Filter HT29 variants and keep only the ones that overlap with a hotspot
ht29_variants <- ht29_variants[queryHits(ol),]
ht29_variants <- ht29_variants %>%
  dplyr::select(gene_symbol=symbol, AA_change=`Mutation AA`, everything()) %>%
  mutate(AA_change = gsub('^p.', '', AA_change))
```

7.3.1 Known cell line variants are discovered with high confidence

There are 2 variants in HT29 (TP53 and BRAF) that we should theoretically find with our amplicon panel. Do we find them? We have a lot of runs with a name like hct*. These are HT-29 cells mixed with varying amounts of HCT116 cells. Therefore, all HT-29 mutations should be detectable from these samples. For the HCT116 cells it depends on our sensitivity.

hct1 90%ht29; 10% hct116 hct2 95%ht29; 5% hct116 hct3 98%ht29; 2%hct116 hct4 99%ht29; 1% hct116 hct5 99,5%ht29; 0,5%hct116 hct6 99,9%ht29;0,1%hct116 hct7 90%ht29; 10% hct116 hct8 95%ht29; 5% hct116 hct9 98%ht29; 2%hct116 hct10 99%ht29; 1% hct116 hct11 99,5%ht29; 0,5%hct116 hct12 99,9%ht29;0,1%hct116

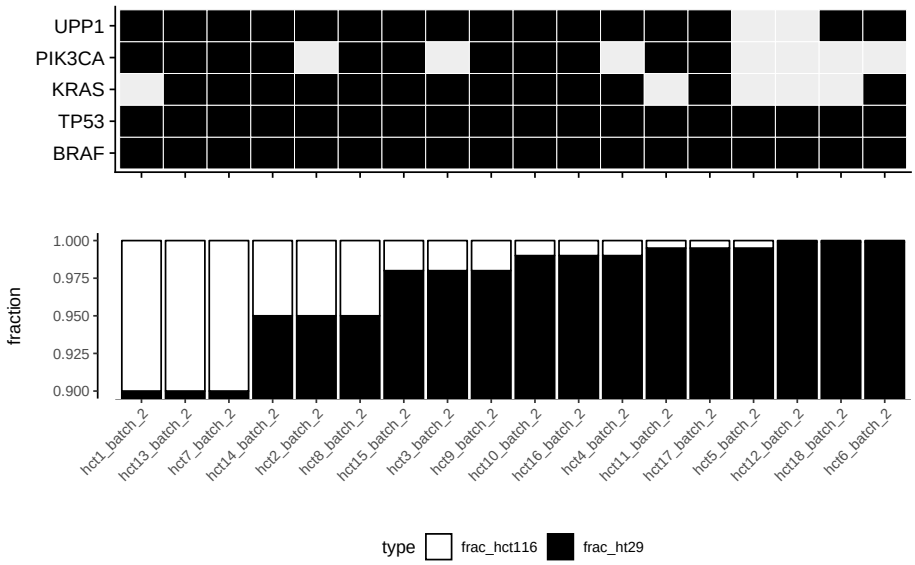
```
cellline_samples <- cl_samples %>%
  dplyr::select(start = variant_pos, end = variant_end, symbol = gene_symbol,
                everything()) %>%
  mutate(end = as.integer(end)) %>%
  inner_join(ht29_variants %>% mutate(start = as.integer(start),
                                      end = as.integer(end))) %>%
  distinct()

## plot heatmap with findings of mutations
ht29_lb_mat_data <- cellline_samples %>%
  dplyr::select(-c(start, end, position_bias, sample_bias)) %>%
  distinct() %>%
  unite(sample_id, sample_name, batch, sep='_') %>%
  dplyr::select(sample_id, symbol, AA_change) %>%
  mutate(found=1) %>%
  distinct() %>%
  spread(sample_id, found) %>%
  mutate_at(vars(-c(symbol, AA_change)), funs(ifelse(is.na(.), 0, .))) %>%
  gather(sample_id, found, -c(symbol, AA_change)) %>%
  mutate(sample_id = factor(sample_id, levels=mixedsort(unique(sample_id)))) %>%
  separate(sample_id, c('sample_name', 'batch'), sep='_',
           extra='merge', remove=F) %>% dplyr::select(-batch)
```

We plot a heatmap with all mutations found and a bar plot indicating the amount of HCT116 present in each samples and we order the matrix by that to make it more interpretable.

```
## ht29 fractions per sampel name
cl_fractions <- list(
  list(sample_name = 'hct1', frac_ht29 = '0.90'),
  list(sample_name = 'hct2', frac_ht29 = '0.95'),
  list(sample_name = 'hct3', frac_ht29 = '0.98'),
  list(sample_name = 'hct4', frac_ht29 = '0.99'),
  list(sample_name = 'hct5', frac_ht29 = '0.995'),
```

```
    list(sample_name = 'hct6', frac_ht29 = '0.999'),
    list(sample_name = 'hct7', frac_ht29 = '0.90'),
    list(sample_name = 'hct8', frac_ht29 = '0.95'),
    list(sample_name = 'hct9', frac_ht29 = '0.98'),
    list(sample_name = 'hct10', frac_ht29 = '0.99'),
    list(sample_name = 'hct11', frac_ht29 = '0.995'),
    list(sample_name = 'hct12', frac_ht29 = '0.999'),
    list(sample_name = 'hct13', frac_ht29 = '0.90'),
    list(sample_name = 'hct14', frac_ht29 = '0.95'),
    list(sample_name = 'hct15', frac_ht29 = '0.98'),
    list(sample_name = 'hct16', frac_ht29 = '0.99'),
    list(sample_name = 'hct17', frac_ht29 = '0.995'),
    list(sample_name = 'hct18', frac_ht29 = '0.999')
  )
  ## add hct116 (1-x)
  cl_fractions <- cl_fractions %>% bind_rows() %>%
    mutate(frac_ht29 = as.numeric(frac_ht29), frac_hct116 = 1-frac_ht29)

  ## levels for sample ordering
  cl_frac_lvls <- cl_fractions %>% arrange(frac_ht29, sample_name) %>% .$sample_name

  ## mutation matrix
  ht29_mut_plot <- ht29_lb_mat_data %>%
    filter(grepl('batch_2', sample_id)) %>%
    mutate(sample_name = factor(sample_name, levels=cl_frac_lvls)) %>%
    arrange(sample_name, sample_id) %>%
    mutate(sample_id = factor(sample_id, levels=unique(sample_id)),
           gene_symbol = factor(symbol, levels=c('BRAF', 'TP53', 'KRAS',
                                                 'PIK3CA', 'MSH6', 'UPP1'))) %>%
    ggplot(aes(x=sample_id, y=gene_symbol, fill=found)) +
    geom_tile(colour = '#ffffff') +
    scale_fill_gradientn(colours=c('#eeeeee', 'black')) +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    theme(legend.position = 'none', axis.text.x = element_blank()) +
    ylab('') + xlab('')

  ## fraction bp
  cl_bp <- cl_fractions %>%
    inner_join(distinct(ht29_lb_mat_data, sample_name, sample_id)) %>%
    mutate(sample_name = factor(sample_name, levels=cl_frac_lvls)) %>%
    filter(grepl('batch_2', sample_id)) %>%
    arrange(sample_name, sample_id) %>%
    mutate(sample_id = factor(sample_id, levels=unique(sample_id))) %>%
    gather(type, fraction, -c(sample_name, sample_id)) %>%
    ggplot(aes(sample_id, fraction, fill=type)) +
    geom_bar(stat='identity', colour='black') +
    theme_classic() + coord_cartesian(ylim=c(0.9, 1)) +
    scale_fill_manual(values=c('white', 'black')) +
    theme(legend.position = 'bottom',
          axis.text.x=element_text(angle=45, hjust=1)) + xlab('')
```

```
## draw on canvas
ht29_mut_plot + cl_bp + plot_layout(ncol=1)
```



8 Global landscape of discovered mutations

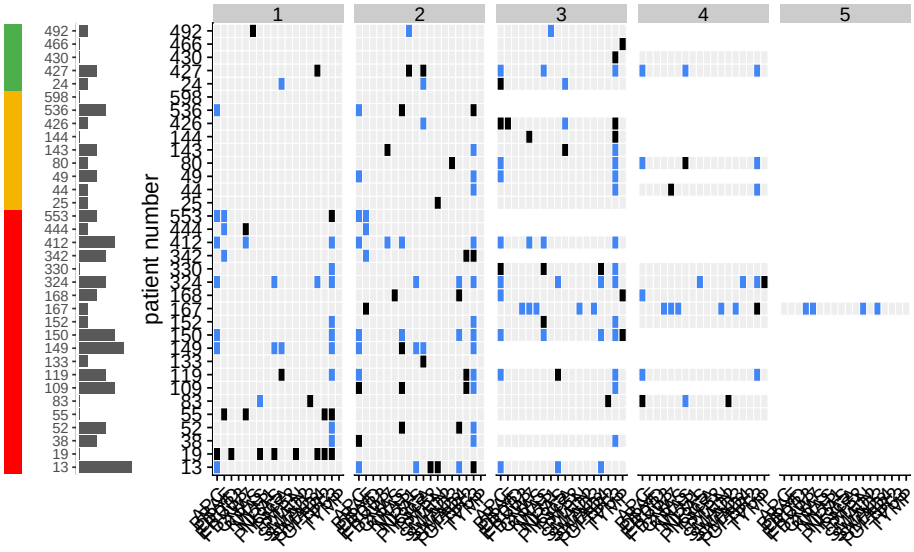
We generate a global landscape heat map that displays mutations across samples over time. We further add tumour progress and number of mutations (combined time point 1 and time point 2) for each sample.

```
all_samples <- base_counts %>% distinct(filename)
all_samples_p2 <- all_samples %>% filter(grepl('^AS', filename)) %>%
    inner_join(sample_info %>%
                   mutate(filename = gsub('.fastq.gz', '_variant_counts.txt',
                                          FASTQ_FILE)) %>%
                   dplyr::select(filename, patient_nr, date) %>%
                   filter(!grepl('hct', patient_nr)))
all_samples_p1 <- all_samples %>%
  filter(!filename %in% all_samples_p2$filename) %>%
  extract(filename, c('patient_nr', 'date'),
          regex = '^(\\d{2,3})[ab]*-(\\d+)_S.+$', remove=F) %>%
  mutate(date = ifelse(nchar(date) == 6, gsub('15$', '2015', date), date),
         date = as.Date(date, '%d%m%Y'))


sample_time_points <- all_samples_p1 %>% filter(!is.na(patient_nr)) %>%
  bind_rows(all_samples_p2 %>%
              mutate(date = ifelse(grepl('\\.', date), as.character(as.Date(date, format = '%d.%m.%Y')), date
                     date = as.Date(date))) %>%
  distinct(patient_nr, date) %>%
  group_by(patient_nr) %>%
  arrange(date) %>% mutate(time_point = 1:n()) %>%
```

```
  ungroup() %>% arrange(patient_nr)

## a list of all mutated genes in each sample
all_mut_genes <- patient_samples %>% filter(!grepl('_NM_', gene_symbol)) %>%
  distinct(patient_nr, date, gene_symbol) %>% mutate(mutated=1) %>%
  mutate(patient_nr = gsub('_\\d+$', '', gsub('dil|a|b', '', patient_nr))) %>%
  distinct() %>%
  unite(sample, patient_nr, date) %>%
  spread(sample, mutated) %>%
  mutate_at(vars(-gene_symbol), funs(ifelse(is.na(.), 0, .))) %>%
  gather(sample, mutated, -gene_symbol) %>%
  separate(sample, c('patient_nr', 'date'), sep='_') %>%
  left_join(known_mutations %>% dplyr::select(patient_nr, gene_symbol) %>%
              mutate(sanger=1, patient_nr = as.character(patient_nr))) %>%
  mutate(sanger=ifelse(is.na(sanger), 0, sanger))# %>%
  # mutate(mutated=ifelse(sanger==1 & mutated==0, 2,
  #                ifelse(sanger==1 & mutated == 1, 3, mutated)))


## average number of mutations per patient sample
all_mut_genes %>% filter(mutated == 1) %>%
  count(patient_nr, date) %>% summarise(mean = mean(n))

## patient factor levels to order by state
patient_lvls <- c(
  ## progress
  '13', '19', '38', '52', '55', '83', '109', '119', '133', '149', '150', '152', '167', '168', '324', '330',
  '412', '444', '553',
  ## stable
  '25', '44', '49', '80', '143', '144', '426', '536', '598',
  ## regress
  '24', '427', '430', '466', '492'
)

## patient status/progress annotation
patient_status <- tibble(patient_lvls, status = 0)
patient_status$status[1:20] <- 1 # progress
patient_status$status[21:29] <- 2 # stable
patient_status$status[30:34] <- 3 # regress
## 0 is unknown

landscape_plot <- all_mut_genes %>% mutate(date = as.Date(date)) %>%
    inner_join(sample_time_points)

## info for samples without mutations
filled <- sample_time_points %>% anti_join(landscape_plot) %>%
  filter(!patient_nr %in% c('51', '51dil', '168_2')) %>%
  mutate(test = map(patient_nr, ~ tibble(gene_symbol = unique(landscape_plot$gene_symbol),
                                         mutated = 0, sanger=0))) %>%
  unnest(test)
```

```
        landscape_plot <- landscape_plot %>% bind_rows(filled) %>%
            mutate(time_point= as.factor(time_point)) %>%
            dplyr::select(-c(sanger, date)) %>%
            spread(time_point, mutated) %>%
            mutate_at(vars(-c(gene_symbol, patient_nr)), funs(ifelse(is.na(.), 0, .))) %>%
            gather(time_point, mutated, -c(gene_symbol, patient_nr)) %>%
            left_join(sample_time_points %>% mutate(time_point=as.character(time_point))) %>%
            mutate(mutated=ifelse(is.na(date), NA, mutated)) %>% dplyr::select(-date) %>%
            mutate(patient_nr = factor(patient_nr, levels=patient_lvls)) %>%
            group_by(patient_nr, gene_symbol) %>% mutate(n_mut = sum(mutated, na.rm=T)) %>% ungroup() %>%
            mutate(mutated = ifelse((mutated != 0) & (n_mut > 1), 2, mutated)) %>%
            ggplot(aes(patient_nr, gene_symbol, fill=mutated)) +
            geom_tile(colour='#ffffff') + facet_wrap(~time_point, nrow=1) +
            scale_fill_gradientn(colours=c('#eeeeee', 'black', '#4285f4'),
                                 na.value = '#ffffff') +
            theme(axis.text.x=element_text(angle=45, hjust=1),
                  legend.position = 'none',
                  plot.margin = unit(c(0, 0, 0, 0), "cm")) +
            ylab('') + xlab('patient number') +
            coord_flip()

        ## plot patient status
        status_plot <- patient_status %>%
          mutate(patient_lvls = factor(patient_lvls, levels=patient_lvls), y=1) %>%
          ggplot(aes(patient_lvls, y, fill=status)) + geom_tile() +
          theme_minimal() +
          scale_fill_gradientn(colours=c('red', '#f4b400', '#4daf4a')) +
          ylab('') + xlab('') +
          theme(panel.grid=element_blank(), legend.position = 'none',
                axis.text.x = element_blank(),
                axis.text.y = element_blank(),
                plot.margin = unit(c(0, 0, 0, 0), "cm")) +
          coord_flip()

        ## number of mutations per patient at tp 1+2
        num_mut_df <- all_mut_genes %>% mutate(date=as.Date(date)) %>%
          right_join(sample_time_points) %>% filter(time_point %in% 2) %>%
          filter(!patient_nr %in% c('51', '51dil', '168_2')) %>%
          mutate(mutated = ifelse(is.na(mutated), 0, mutated)) %>%
          group_by(patient_nr) %>% summarise(n = sum(mutated)) %>% ungroup() %>%
          mutate(patient_nr = factor(patient_nr, levels=patient_lvls))
        num_mut_plot <- num_mut_df %>% ggplot(aes(patient_nr, n)) + geom_bar(stat='identity') +
          ylab('') + xlab('') + theme_classic() +
          theme(panel.grid=element_blank(),
                axis.text.x=element_blank(),
                axis.line.x = element_blank(),
                axis.ticks.x = element_blank(),
                plot.margin = unit(c(0, 0, 0, 0), "cm")) +
          coord_flip()

        ## combine plots
```

```
status_plot + num_mut_plot + landscape_plot +
  plot_layout(nrow=1, widths = c(0.025, 0.075, 0.9))
```



8.1 Tumour progression and mutation load

It seems that tumours that are progressing in sample time point 1 show more mutations than the others in time point 0. Is this a significant trend? Can we anticipate tumour progression by cfDNA mutation load?

```
## patients and their progress
patients_prog <- c('13', '19', '38', '52', '55', '83',
                   '109', '119', '133', '149', '152',
                   '168', '324', '330', '342')
patients_stable <- c('44', '49', '80', '143', '144', '150', '167')
patients_regress <- c('24', '427', '430')


all_mut_genes %>% mutate(date=as.Date(date)) %>%
  inner_join(sample_time_points) %>%
  filter(time_point %in% 1, !mutated %in% c(0, 2)) %>%
  count(patient_nr)

## I annotate progress/regress/stable for each sample
sample_time_points <- sample_time_points %>%
  mutate(patient_nr = as.integer(patient_nr)) %>%
  arrange(patient_nr, time_point) %>% drop_na()

## I ordered this list manually
sample_progress <- c(c('stable', 'progress', 'progress'), ## p 13
                     c('regress', 'progress'), # p19
                     c('regress', 'regress', 'progress'), ## p 24
                     c('regress', 'stable', 'progress'), ## p 25
```

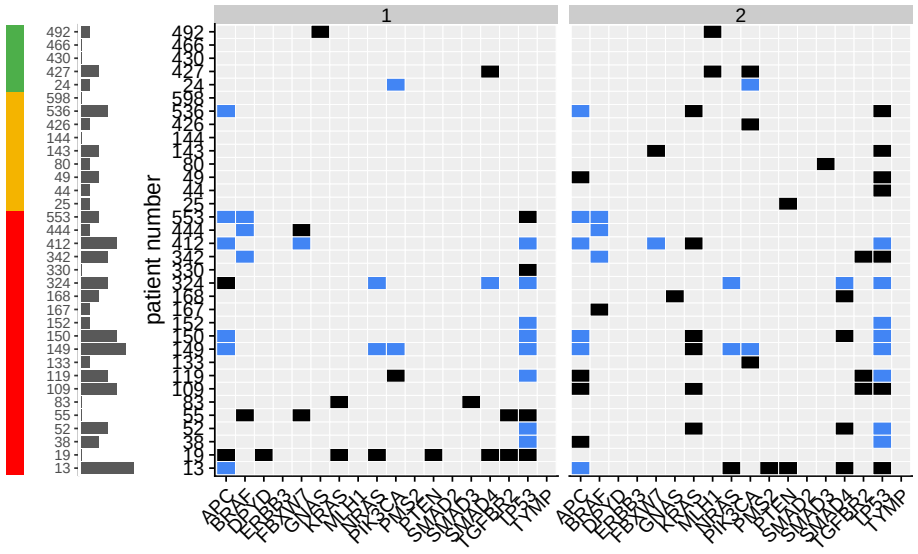
```
                        c('regress', 'progress', 'progress'), ## p 38
                        c('stable', 'stable', 'progress', NA), ## p 44
                        c('stable', 'stable', 'progress'), ## p 49
                        c('stable', 'progress'), ## p 52
                        c('stable', 'progress', NA, NA), ## p 55
                        c('stable', 'stable', NA, NA), ## p 80
                        c('stable', 'progress', NA, NA), ## p 83
                        c('stable', 'progress', NA), ## p 109
                        c('regress', 'progress', 'stable', 'progress'), ## p 119
                        c('stable', 'progress'), ## p 133
                        c('stable', 'stable', 'stable'),  ## p 143
                        c('regress', 'stable', 'stable'), ## p 144
                        c('regress', 'progress'), ## p 149
                        c('stable', 'progress', NA), ## p 150
                        c('stable', 'progress', NA, NA), ## p 152
                        c('stable', 'progress', 'stable', NA, NA), ## p 167
                        c('stable', 'progress', 'stable', NA), ## p 168
                        c('stable', 'progress', NA, NA), ## p 324
                        c('regress', 'progress', 'regress', 'regress'), ## p 330
                        c('stable', 'progress'), ## p 342
                        c('progress', NA, 'progress'), ## p 412
                        c('stable', 'progress', NA), ## p 426
                        c('regress', 'regress', 'progress', 'progress'), ## p 427
                        c('stable', 'regress', 'stable', 'stable'), ## p 430
                        c('stable', 'progress'), ## p 444
                        c('regress', 'regress', NA), ## p 466
                        c('regress', 'regress', 'regress'), ## p 492
                        c('stable', 'stable'), ## p 536
                        c('regress', 'progress'), ## p 553
                        c('regress', 'stable') ## p 598
)

sample_time_points <- sample_time_points %>% mutate(progress = sample_progress)
```

We repeat the same showing only samples from time points 1 and 2.

```
## samples for time points 1 and 2 only
stp2 <- sample_time_points %>%
                group_by(patient_nr) %>% top_n(2, -time_point) %>% ungroup() %>%
  mutate(patient_nr = as.character(patient_nr)) %>%
  dplyr::select(-progress)

landscape_plot <- all_mut_genes %>% mutate(date = as.Date(date)) %>%
    inner_join(stp2)

## info for samples without mutations
filled <- stp2 %>% anti_join(landscape_plot) %>%
  filter(!patient_nr %in% c('51', '51dil', '168_2')) %>%
  mutate(test = map(patient_nr, ~ tibble(gene_symbol = unique(landscape_plot$gene_symbol),
                                         mutated = 0, sanger=0))) %>%
  unnest(test)
```

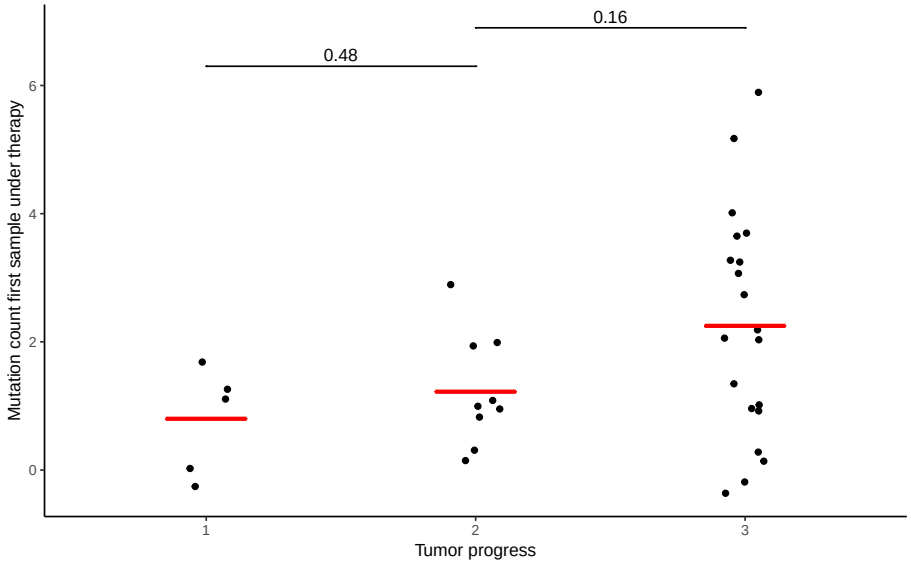
```
landscape_plot <- landscape_plot %>% bind_rows(filled) %>%
    mutate(time_point= as.factor(time_point)) %>%
    dplyr::select(-c(sanger, date)) %>%
    spread(time_point, mutated) %>%
    mutate_at(vars(-c(gene_symbol, patient_nr)), funs(ifelse(is.na(.), 0, .))) %>%
    gather(time_point, mutated, -c(gene_symbol, patient_nr)) %>%
    left_join(stp2 %>% mutate(time_point=as.character(time_point))) %>%
    mutate(mutated=ifelse(is.na(date), NA, mutated)) %>% dplyr::select(-date) %>%
    mutate(patient_nr = factor(patient_nr, levels=patient_lvls)) %>%
    group_by(patient_nr, gene_symbol) %>% mutate(n_mut = sum(as.integer(mutated), na.rm=T)) %>% ungroup() %>%
    mutate(mutated = ifelse((mutated != 0) & (n_mut > 1), 2, mutated)) %>%
    filter(!is.na(mutated)) %>% mutate(mutated = as.integer(mutated)) %>%
    ggplot(aes(patient_nr, gene_symbol, fill=mutated)) +
    geom_tile(colour='#ffffff') + facet_wrap(~time_point, nrow=1) +
    scale_fill_gradientn(colours=c('#eeeeee', 'black', '#4285f4'),
                         na.value = '#ffffff') +
    theme(axis.text.x=element_text(angle=45, hjust=1),
          legend.position = 'none',
          plot.margin = unit(c(0, 0, 0, 0), "cm")) +
    ylab('') + xlab('patient number') +
    coord_flip()

## plot patient status
status_plot <- patient_status %>%
  mutate(patient_lvls = factor(patient_lvls, levels=patient_lvls), y=1) %>%
  ggplot(aes(patient_lvls, y, fill=status)) + geom_tile() +
  theme_minimal() +
  scale_fill_gradientn(colours=c('red', '#f4b400', '#4daf4a')) +
  ylab('') + xlab('') +
  theme(panel.grid=element_blank(), legend.position = 'none',
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        plot.margin = unit(c(0, 0, 0, 0), "cm")) +
  coord_flip()

## number of mutations per patient at tp 1+2
num_mut_df <- all_mut_genes %>% mutate(date=as.Date(date)) %>%
  right_join(stp2) %>% filter(time_point %in% 2) %>%
  filter(!patient_nr %in% c('51', '51dil', '168_2')) %>%
  mutate(mutated = ifelse(is.na(mutated), 0, mutated)) %>%
  group_by(patient_nr) %>% summarise(n = sum(mutated)) %>% ungroup() %>%
  mutate(patient_nr = factor(patient_nr, levels=patient_lvls))
num_mut_plot <- num_mut_df %>% ggplot(aes(patient_nr, n)) + geom_bar(stat='identity') +
  ylab('') + xlab('') + theme_classic() +
  theme(panel.grid=element_blank(),
        axis.text.x=element_blank(),
        axis.line.x = element_blank(),
        axis.ticks.x = element_blank(),
        plot.margin = unit(c(0, 0, 0, 0), "cm")) +
  coord_flip()
```

```
## combine plots
status_plot + num_mut_plot + landscape_plot +
  plot_layout(nrow=1, widths = c(0.025, 0.075, 0.9))
```



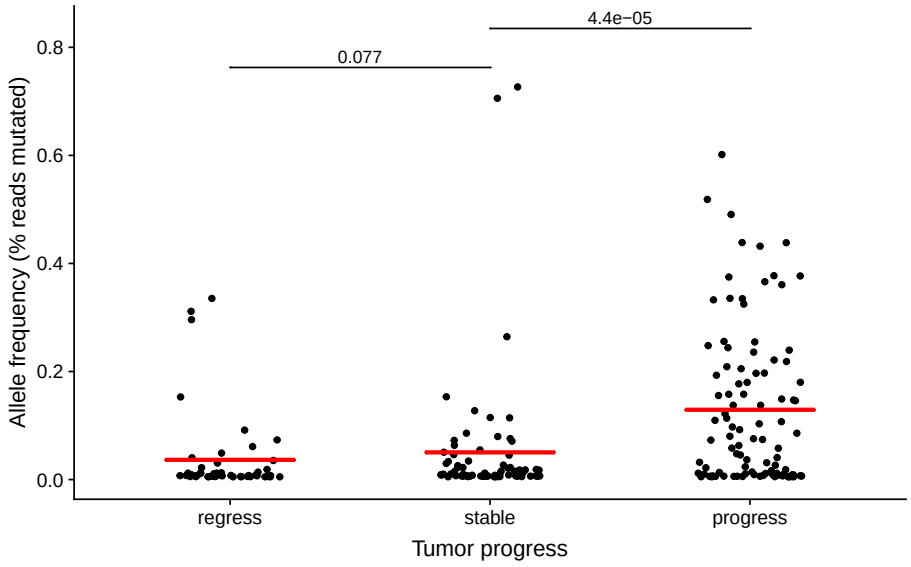
We can also make a plot that quantifies the relationship between mutation load and tumour progress state.

```
num_mut_df %>% inner_join(patient_status %>% dplyr::select(patient_nr = patient_lvls, status)) %>%
  mutate(status_rev = ifelse(status == 1, 3,
                      ifelse(status == 2, 2, 1))) %>%
  ggplot() + geom_jitter(aes(as.factor(status_rev), n), width = 0.1) +
  stat_summary(aes(as.factor(status_rev), n),
               fun.y = 'mean', fun.ymin='mean', fun.ymax = 'mean',
               colour = 'red', width=0.3, geom='crossbar') +
  theme_classic() +
  ggsignif::geom_signif(aes(as.factor(status_rev), n),
                        comparisons = list(c('1', '2'), c('2', '3')), tip_length = 0, step_increase = 0.1) +
  ylab('Mutation count first sample under therapy') +
  xlab('Tumor progress')
```

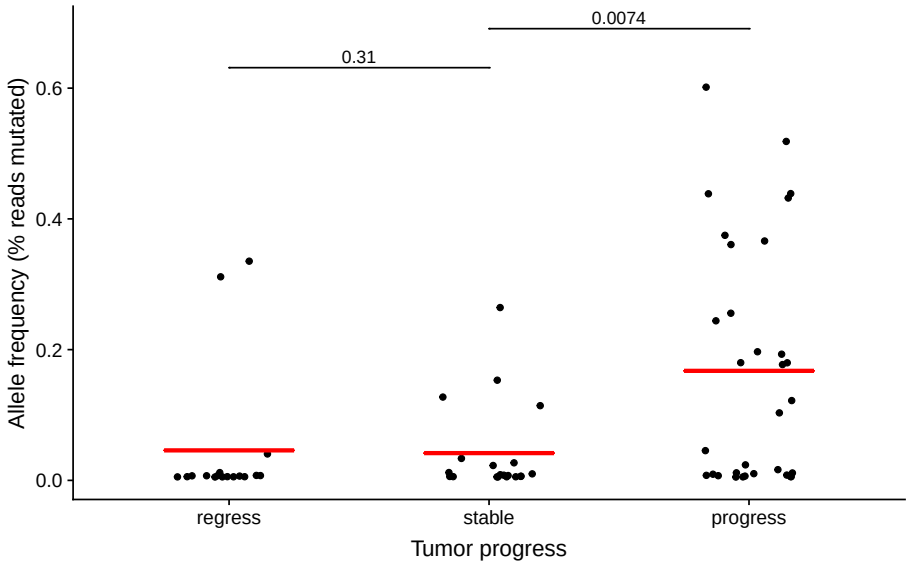


We want to check if there is a correlation between tumor progress and allele frequency. We generate a global boxplot and then we look at individual genes.

```
patient_samples %>% mutate(date = as.Date(date),
                           patient_nr = as.integer(patient_nr)) %>%
  inner_join(sample_time_points) %>%
  filter(!is.na(progress)) %>%
  mutate(progress = factor(progress, levels = c('regress', 'stable', 'progress'))) %>%
  ggplot(aes(progress, allele_frequency)) +
    geom_jitter(width = 0.2) +
    stat_summary(fun.y = 'mean', fun.ymin = 'mean',
                 fun.ymax = 'mean', colour = 'red', width = 0.5, geom='crossbar') +
  xlab('Tumor progress') + ylab('Allele frequency (% reads mutated)') +
  ggsignif::geom_signif(comparisons = list(c('regress', 'stable'), c('stable', 'progress')),
                        tip_length = 0, step_increase = 0.1)
```



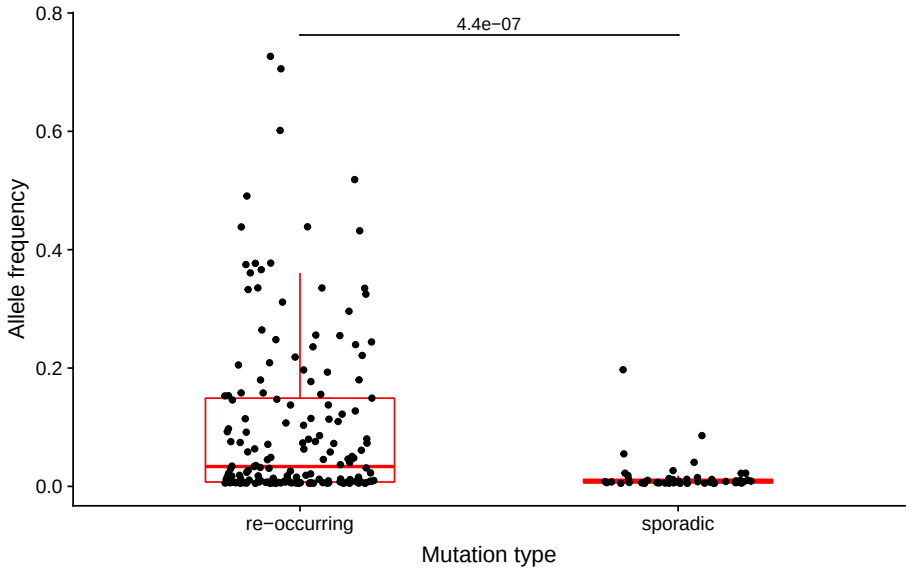
```
## for individual genes ?
patient_samples %>% mutate(date = as.Date(date),
                           patient_nr = as.integer(patient_nr)) %>%
  inner_join(sample_time_points) %>%
  filter(!is.na(progress)) %>%
  mutate(progress = factor(progress, levels = c('regress', 'stable', 'progress'))) %>%
  filter(gene_symbol == 'TP53') %>%
  ggplot(aes(progress, allele_frequency)) +
    geom_jitter(width = 0.2) +
    stat_summary(fun.y = 'mean', fun.ymin = 'mean',
                 fun.ymax = 'mean', colour = 'red', width = 0.5, geom='crossbar') +
  xlab('Tumor progress') + ylab('Allele frequency (% reads mutated)') +
  ggsignif::geom_signif(comparisons = list(c('regress', 'stable'), c('stable', 'progress')),
                        tip_length = 0, step_increase = 0.1)
```



Do re-occurring mutations have higher allele frequencies?

```
df <- patient_samples %>% mutate(date = as.Date(date),
                           patient_nr = as.integer(patient_nr)) %>%
    inner_join(sample_time_points) %>%
    filter(!is.na(progress)) %>%
    mutate(progress = factor(progress, levels = c('regress', 'stable', 'progress')))

reoccurring <- df %>% count(patient_nr, gene_symbol) %>%
  mutate(reoccurring = ifelse(n > 1, 're-occurring', 'sporadic')) %>%
  dplyr::select(-n)

df %>% inner_join(reoccurring) %>%
  ggplot(aes(reoccurring, allele_frequency)) +
  geom_boxplot(colour='red', width=0.5, outlier.colour = '#ffffff') +
  geom_jitter(width=0.2) +
  ggsignif::geom_signif(comparisons = list(c('re-occurring', 'sporadic')),
                        tip_length = 0) +
```

```
  xlab('Mutation type') + ylab('Allele frequency')
```



9 Analysis of the different patients

9.1 CEA Values

We load information about CEA values measured for each sample.

```
data('cea_values', package='HDLB2018')
```

9.2 Time course

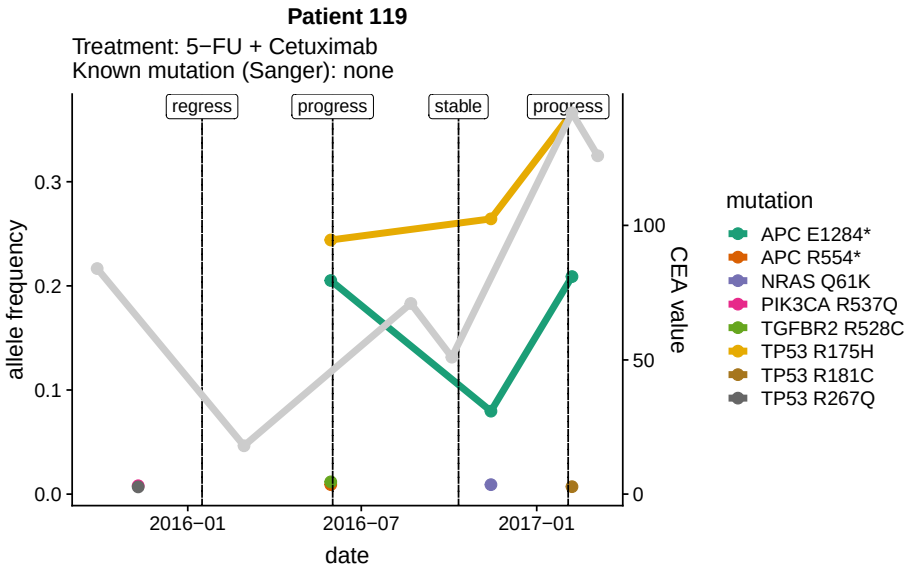
For each patient we generate a plot that visualizes the mutations and their frequencies in the context of time and progress. In addition we list the treatment that the patient received.

```
## first I need functions to scale values to specific ranges
scale_to_range <- function(vals, high){
  (high/max(vals, na.rm=T)) * (vals - max(vals, na.rm=T)) + high
}

scale_back <- function(vt, vals, high){
  (vt * max(vals, na.rm=T))/high
}

visualize_patient_timeline <- function(pnr){
  print(paste('Patient', pnr))
  ## generate a data frame with all require information
  df <- patient_samples %>% filter(patient_nr == pnr) %>%
```

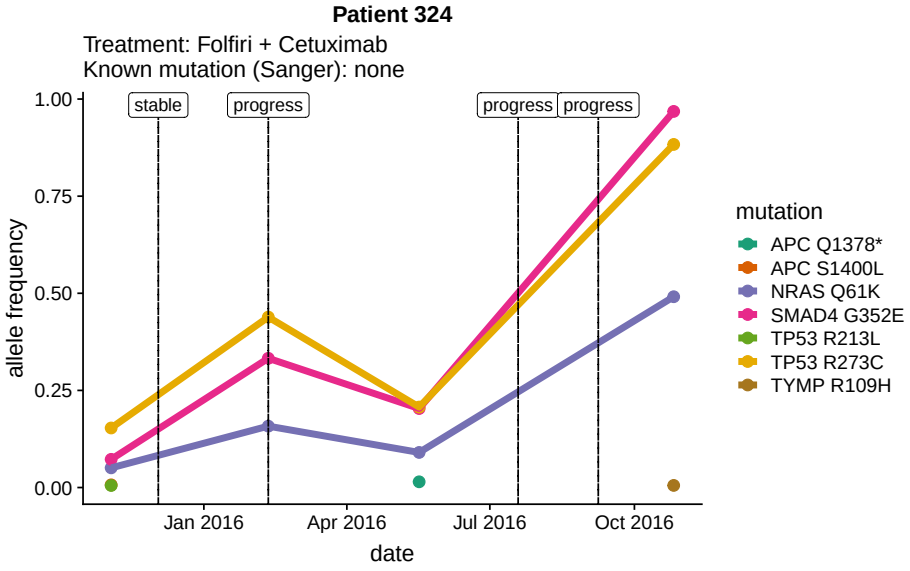
```
    filter(!grepl('_NM_', gene_symbol)) %>%
    dplyr::select(patient_nr, date, gene_symbol, AA_change, allele_frequency) %>%
    group_by(patient_nr, date, gene_symbol, AA_change) %>%
    summarise(allele_frequency = max(allele_frequency)) %>% ungroup() %>%
    mutate(date=as.Date(date)) %>%
    unite(mutation, gene_symbol, AA_change, sep=' ')

  muts18 <- df %>% inner_join(count(df, mutation)) %>%
    arrange(desc(n), desc(allele_frequency)) %>%
    .$mutation %>% unique %>% .[1:8]

  df <- df %>% filter(mutation %in% muts18) %>%
    full_join(patient_treatment %>% mutate(patient_nr = as.character(patient_nr)) %>%
                filter(patient_nr == pnr)) %>%
    full_join(tumour_progress %>% mutate(patient_nr = as.character(patient_nr)) %>%
                filter(patient_nr == pnr))

  km <- known_mutations %>% mutate(patient_nr = as.character(patient_nr)) %>%
    filter(patient_nr == pnr)
  km <- ifelse(nrow(km) > 0, paste(km$gene_symbol[1], km$AA_change[1]), 'none')

  ## CEA values
  cea <- cea_values %>% filter(patient_nr == pnr) %>%
    mutate(scaled = scale_to_range(CEA, max(df$allele_frequency, na.rm=T)))

  ## plot as ggplot
  p <- ggplot() + geom_point(data=filter(df, !is.na(mutation)), size=3,
                             aes(colour=mutation, x=date, y=allele_frequency)) +
    geom_line(data=filter(df, !is.na(mutation)),
              aes(x=date, colour=mutation, y=allele_frequency), lwd=2) +
    geom_rect(data=distinct(cea, date, CT) %>% filter(!is.na(CT)),
              ymin = -Inf, ymax = Inf, aes(xmin=date, xmax=date),
              colour = 'black', linetype = 'dashed') +
    geom_label(data=distinct(cea, date, CT) %>% filter(!is.na(CT)),
               aes(x = date, label = CT), y = Inf, vjust = 1) +
    labs(title = paste('Patient', pnr),
         subtitle = paste('Treatment:', df$treatment[1], '\nKnown mutation (Sanger):', km)) +
    ylab('allele frequency') +
    scale_color_brewer(palette = 'Dark2')

  if(!all(is.na(cea$CEA))){
    p <- p +
      geom_point(data = filter(cea, !is.na(CEA)),
              aes(x=date, y = scaled), size=3, colour='#cccccc') +
      geom_line(data = filter(cea, !is.na(CEA)),
                aes(x=date, y = scaled), lwd=2, colour='#cccccc') +
      scale_y_continuous(sec.axis = sec_axis(~ scale_back(., cea$CEA, max(df$allele_frequency, na.rm=T)),
                                             name = 'CEA value'))
  }
  return(p)
}
```

We plot the time line for four selected patients.

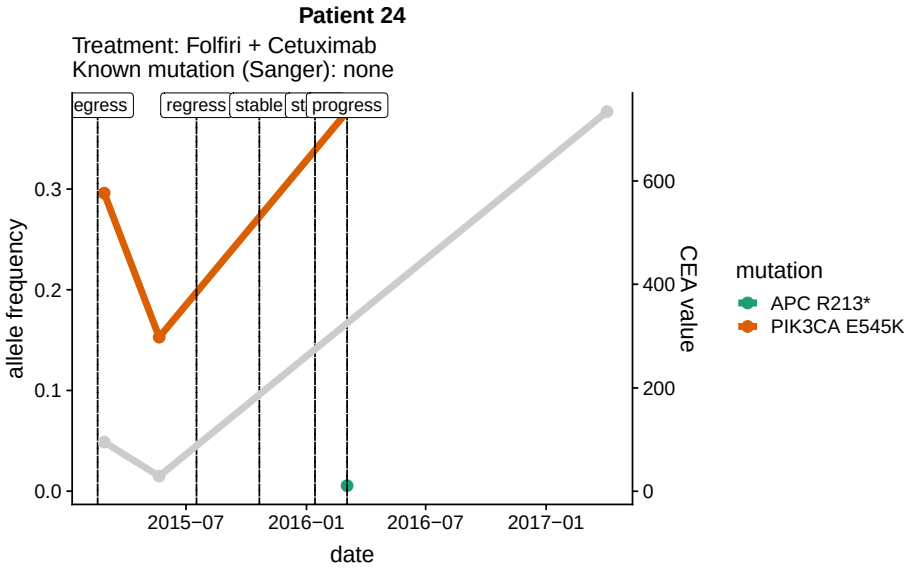
```
## patient 119
visualize_patient_timeline('119')
```



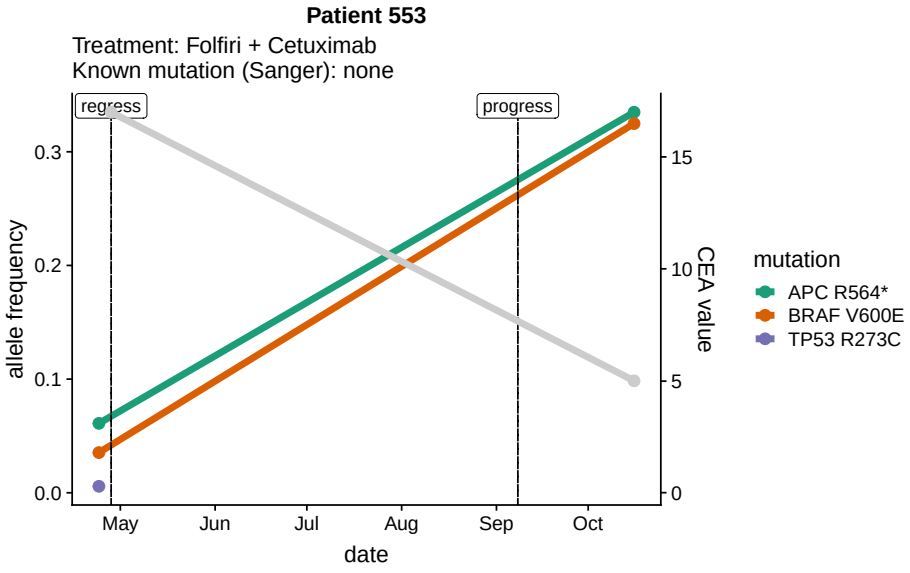

```
## patient 324
visualize_patient_timeline('324')
```




```
## patient 24
visualize_patient_timeline('24')
```




```
## patient 553
visualize_patient_timeline('553')
```



10 Session info

```
sessionInfo()
#> R version 3.4.1 (2017-06-30)
#> Platform: x86_64-apple-darwin15.6.0 (64-bit)
#> Running under: macOS Sierra 10.12.6
#>
#> Matrix products: default
#> BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
```

```
#> LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#> [1] parallel  stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] bindrcpp_0.2.2                   GenomicFeatures_1.30.3
#>  [3] AnnotationDbi_1.40.0             Biobase_2.38.0
#>  [5] treemap_2.4-2                    genemodel_1.1.0
#>  [7] BSgenome.Hsapiens.UCSC.hg38_1.4.1 BSgenome_1.46.0
#>  [9] rtracklayer_1.38.3               MASS_7.3-50
#> [11] Biostrings_2.46.0                XVector_0.18.0
#> [13] patchwork_0.0.1                  gtools_3.8.1
#> [15] biomaRt_2.34.2                   cowplot_0.9.3
#> [17] ggsignif_0.4.0                   pheatmap_1.0.10
#> [19] openxlsx_4.1.0                   forcats_0.3.0
#> [21] stringr_1.3.1                    dplyr_0.7.6
#> [23] purrr_0.2.5                      readr_1.1.1
#> [25] tidyr_0.8.1                      tibble_1.4.2
#> [27] ggplot2_3.0.0                    tidyverse_1.2.1
#> [29] GenomicRanges_1.30.3             GenomeInfoDb_1.14.0
#> [31] IRanges_2.12.0                   S4Vectors_0.16.0
#> [33] BiocGenerics_0.24.0              BiocStyle_2.6.1
#>
#> loaded via a namespace (and not attached):
#>  [1] colorspace_1.3-2          rprojroot_1.3-2
#>  [3] rstudioapi_0.7            bit64_0.9-7
#>  [5] fansi_0.2.3               lubridate_1.7.4
#>  [7] xml2_1.2.0                knitr_1.20
#>  [9] jsonlite_1.5              Rsamtools_1.30.0
#> [11] broom_0.5.0               gridBase_0.4-7
#> [13] shiny_1.1.0               compiler_3.4.1
#> [15] httr_1.3.1                backports_1.1.2
#> [17] assertthat_0.2.0          Matrix_1.2-14
#> [19] lazyeval_0.2.1            cli_1.0.0
#> [21] later_0.7.3               htmltools_0.3.6
#> [23] prettyunits_1.0.2         tools_3.4.1
#> [25] igraph_1.2.2              gtable_0.2.0
#> [27] glue_1.3.0                GenomeInfoDbData_1.0.0
#> [29] Rcpp_0.12.18              cellranger_1.1.0
#> [31] nlme_3.1-137              xfun_0.3
#> [33] rvest_0.3.2               mime_0.5
#> [35] XML_3.98-1.15             zlibbioc_1.24.0
#> [37] scales_1.0.0              hms_0.4.2
#> [39] promises_1.0.1            SummarizedExperiment_1.8.1
#> [41] RColorBrewer_1.1-2        yaml_2.2.0
#> [43] memoise_1.1.0             stringi_1.2.4
```

```
#> [45] RSQLite_2.1.1             RMySQL_0.10.15
#> [47] zip_1.0.0                 BiocParallel_1.12.0
#> [49] rlang_0.2.1               pkgconfig_2.0.1
#> [51] bitops_1.0-6              matrixStats_0.54.0
#> [53] evaluate_0.11             lattice_0.20-35
#> [55] bindr_0.1.1               GenomicAlignments_1.14.2
#> [57] labeling_0.3              bit_1.1-14
#> [59] tidyselect_0.2.4          plyr_1.8.4
#> [61] magrittr_1.5              bookdown_0.7
#> [63] R6_2.2.2                  DelayedArray_0.4.1
#> [65] DBI_1.0.0                 pillar_1.3.0
#> [67] haven_1.1.2               withr_2.1.2
#> [69] RCurl_1.95-4.11           modelr_0.1.2
#> [71] crayon_1.3.4              utf8_1.1.4
#> [73] rmarkdown_1.10            progress_1.2.0
#> [75] grid_3.4.1                readxl_1.1.0
#> [77] data.table_1.11.4         blob_1.1.1
#> [79] digest_0.6.15             xtable_1.8-2
#> [81] httpuv_1.4.5              munsell_0.5.0
```