

Summary statistics: Lymphoma predictive modeling

Sleiman Bassim, PhD

Project started in December 2017
Original version published in August 2018
Last updated on March 12, 2019

Contents

1 Exploratory Data Analysis	3
1.1 Data reformatting	3
1.1.1 On-array content of non-coding RNA	4
1.1.2 Regression analyses to quantify diagnosis connections	5
1.2 Featured data and groups of sample cases	6
2 Differential expression of microarray Affymetrix data	9
2.1 Cleaning and removing non-essential genes	12
2.1.1 Variance optimization for each array	12
2.1.2 Standard deviation optimization for each array	14
2.2 Distribution of p-values in pairwise comparisons	16
3 Clustering and network analyses	20
3.1 Network analysis for Spearman-related correlations (relaxed)	23
3.1.1 Nodal versus extra-nodal lymphoma	23
3.1.2 Relapsed versus no CNS relapsed cases	24
3.1.3 Lymphoma cases classified by Cell-of-origin subtypes	25
3.2 Network analysis for Pearson-related correlations (relaxed)	26
3.2.1 Nodal versus extra-nodal lymphoma	26
3.2.2 Relapsed versus no CNS relapsed cases	27
3.2.3 Lymphoma cases classified by Cell-of-origin subtypes	28
3.3 Network analysis for Spearman-related correlations (stringent)	29
3.3.1 Nodal versus extra-nodal lymphoma	29
3.3.2 Relapsed versus no CNS relapsed cases	30
3.3.3 Lymphoma cases classified by Cell-of-origin subtypes	31
3.4 Network analysis for Pearson-related correlations (stringent)	32
3.4.1 Nodal versus extra-nodal lymphoma	32
3.4.2 Relapsed versus no CNS relapsed cases	33
3.4.3 Lymphoma cases classified by Cell-of-origin subtypes	34
4 Machine Learning	35
4.1 Regularization	35
4.1.1 Uncertainty estimation for selected genes from expression networks	35
4.2 Selected machine and deep learning models	37
4.3 Available and tuned hyperparameters for each model	38
4.4 Machine learning performance benchmarks	38
4.4.1 Creating the baseline of models performance	38
4.4.2 Models performance with hyperparameter tuning	44
4.4.3 Genomic representation of selected genes	47
4.4.4 Prediction accuracy of each classifier at optimal parameters	48
4.4.5 Probability to classify samples with the best model	55
4.5 Importance scope of gene contribution	59
4.6 Accuracy measured across pipelines	64
4.7 Expression of important genes summarized by RMA scores	65
4.8 Expression of important genes summarized by limma scores	73
4.9 All classifying genes visualized by prognosis classes	77
4.9.1 Top classifier genes	77
4.9.2 Top importance genes	79
4.10 Version of machine learning models on high performance clusters	81
5 Clonal evolution	83
5.1 Distribution of variants by estimated allele frequency	83
5.2 Allele frequencies and variants using different callers	84
5.2.1 Individual 1	84
5.2.2 Individual 2	91
5.2.3 Individual 3	95
5.2.4 Individual 4	100

5.2.5	Individual 5	105
5.3	Location of called variants	109
5.4	Cellular prevalence and clustering of tumor clones	110
5.4.1	Variant allele frequency by individual using Varscan2	110
5.4.2	Cellular prevalence by individual using Varscan2	114
5.4.3	Variant allele frequency by individual using Bcftools	116
5.4.4	Cellular prevalence by individual using Bcftools	120
5.4.5	Variant allele frequency by individual using GATK	122
5.4.6	Cellular prevalence by individual using GATK	122
6	System information for this report	123

1 Loading packages.

```
pkgs <- c('gdata', 'lattice', 'latticeExtra', 'gplots',
  'ggplot2', 'dplyr', 'tidyR', 'RColorBrewer', 'igraph',
  'DescTools', 'scales', 'brotools', 'Hmisc', 'finalfit',
  'plyr', 'paletteer', 'ggrepel', 'ggExtra', 'ggpubr',
  'cowplot', 'ggridges', 'reshape')
lapply(pkgs, require, character.only = TRUE)
```

2 1 Exploratory Data Analysis

3 Data is from individuals with Lymphoma tumors, either undergone or not a Rituximab CHOP treatment.
4 Some individuals show relapse after treatment. Tumors migrate through nodal (lymphnodes) or extranodal
5 tissues. Tumors involve two different subtypes of cells of origin, ABC or GCB. **The first aim is to find**
6 **correlation genes that respond differently to treatment, nodal transmission, and cell subtypes.**

^aOR: Odds ratio. HR: Hazard ratio

```
#read.table("data/phenodata", sep = "\t", header = T) %>%
#  dplyr::select(SAMPLE_ID, Timepoint,
#  GROUP, SITE, Score, Prediction, ABClikelihood) %>%
#  brotools::describe()

print_summary_table <- function(features, dependent, df, execute = TRUE) {
  if ( execute == TRUE ) {
    x <- df %>%
      summary_factorlist(dependent, features, p=FALSE, add_dependent_label=TRUE)
    ## print latex table
    Hmisc::latex(x, file = "", booktabs = TRUE, title = "")
  } else {
    cat("LaTeX summary table printed\n")
  }
}

dfs <- read.table("data/phenodata", sep = "\t", header = T)
print_summary_table(features= c("Score", "ABClikelihood", "GROUP"),
  dependent= c("Prediction"),
  df = dfs,
  execute = F)
```

LaTeX summary table printed

Dependent: Prediction		ABC	GCB	U
10	Score	Mean (SD)	3156.3 (475.5)	506.4 (721.1)
1	ABClikelihood	Mean (SD)	1 (0)	0 (0)
2	GROUP	CNS DIAGNOSIS	4 (33.3)	6 (50.0)
3		CNS RELAPSE CHOP or EQUIVALENT	6 (60.0)	3 (30.0)
4		CNS RELAPSE RCHOP	17 (44.7)	13 (34.2)
5		NO RELAPSE	27 (28.1)	52 (54.2)
6		NORMAL ABC CONTROL	2 (100.0)	0 (0.0)
7		NORMAL GCB CONTROL	0 (0.0)	4 (100.0)
8		SYSTEMIC RELAPSE NO CNS	31 (48.4)	25 (39.1)
9		TESTICULAR NO CNS RELAPSE	9 (75.0)	0 (0.0)
				3 (25.0)

7 1.1 Data reformatting

8 In the first steps of the analysis, the samples will be labeled (supervised) into the following categories
9 (based on patients diagnosis).

```
metadata <- read.table("data/phenodata", sep = "\t", header = T) %>%
```

```

dplyr::select(SAMPLE_ID, Timepoint, GROUP, SITE, Score, Prediction, ABClikelihood) %>%
filter(Timepoint != "T2") %>%
mutate(Groups = case_when(GROUP %in% c("CNS_RELAPSE_RCHOP",
                                         "CNS_RELAPSE_CHOPorEQUIVALENT",
                                         "CNS_DIAGNOSIS") ~ "CNS",
                                         GROUP %in% c("TESTICULAR_NO_CNS_RELAPSE", "NO_RELAPSE") ~ "NOREL",
                                         GROUP == "SYSTEMIC_RELAPSE_NO_CNS" ~ "SYST",
                                         TRUE ~ "CTRL")) %>%
filter(Groups != "CTRL") %>%
mutate(ABCClassify = case_when(ABCliglihood >= .9 ~ "ABC",
                                         ABClielihood <= .1 ~ "GCB",
                                         TRUE ~ "U")) %>%
mutate(ABCScore = case_when(Score > 2412 ~ "ABC",
                                         Score <= 1900 ~ "GCB",
                                         Score == NA ~ "NA",
                                         TRUE ~ "U")) %>%
#
# mutate(Nodes = case_when(SITE == "LN" ~ "LN",
                                         SITE == "TO" ~ "LN",
                                         SITE == "SP" ~ "LN",
                                         TRUE ~ "EN")) %>%
mutate(Lymphnodes = case_when(Nodes == "LN" ~ 1, TRUE ~ 0))

# factorize
metadata$Groups <- as.factor(metadata$Groups)
metadata$ABCClassify <- as.factor(metadata$ABCClassify)
metadata$ABCScore <- as.factor(metadata$ABCScore)
metadata$Nodes <- as.factor(metadata$Nodes)
metadata$Lymphnodes <- as.factor(metadata$Lymphnodes)

## reorder sample names
ids <- read.table("data/sampleIDs")
metadata <- arrange(metadata, factor(SAMPLE_ID, levels = ids$V1))
meta.selected <- metadata %>%
  mutate(Contrast1 = as.factor(paste0(Groups, ".", Prediction))) %>%
  mutate(Contrast2 = as.factor(paste0(Groups, ".", Nodes)))

#brotools::describe(metadata)
print_summary_table(c("ABCScore", "ABCClassify", "GROUP",
                     "Contrast1", "Contrast2"), c("Nodes"), meta.selected, execute = F)

```

LaTeX summary table printed

Dependent: Nodes			EN	LN
4	ABCScore	ABC	34 (37.0)	58 (63.0)
5		GCB	36 (35.0)	67 (65.0)
6		U	16 (39.0)	25 (61.0)
1	ABCClassify	ABC	37 (35.9)	66 (64.1)
2		GCB	38 (32.5)	79 (67.5)
3		U	11 (68.8)	5 (31.2)
7	GROUP	CNS DIAGNOSIS	7 (63.6)	4 (36.4)
8		CNS RELAPSE CHOP or EQUIVALENT	5 (62.5)	3 (37.5)
9		CNS RELAPSE RCHOP	20 (51.3)	19 (48.7)
10		NO RELAPSE	30 (31.2)	66 (68.8)
11		NORMAL ABC CONTROL	2 (NA)	0 (0.0)
12		NORMAL GCB CONTROL	0 (0.0)	4 (100.0)
13		SYSTEMIC RELAPSE NO CNS	10 (15.6)	54 (84.4)
14		TESTICULAR NO CNS RELAPSE	12 (100.0)	0 (0.0)

1.1.1 On-array content of non-coding RNA

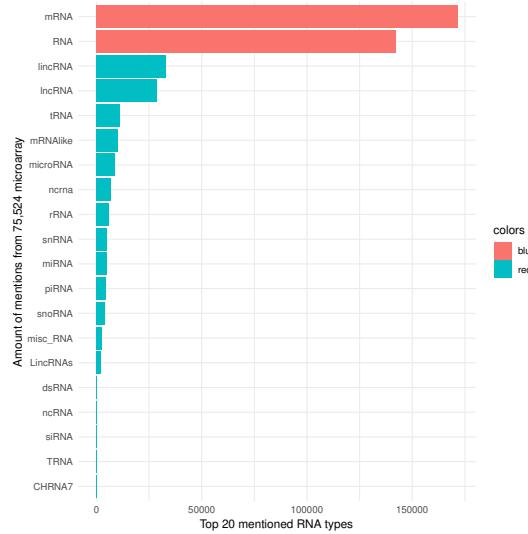
One microarray contains 70,524 probes, of which 12,969 genes do not contain any of the terms related to non-coding RNAs. The graph shows in red that the coding genes are categorized as mRNA or RNA. The non-coding RNAs are in blue and there is over 600 different mentions among the probes (large intergenic non-coding RNAs, lncRNAs make up most of the long ncRNAs). Even though they are less mentioned in gene annotations (smaller bars) they do however cover 70% of the annotated probes.

↑ Two or more mentions (patterns) can be recognized in one gene

```

colors <- c(rep("blue",2), rep("red",18))
read.table("./data/rna.patterns.annotated.array.txt", header = F) %>%
  mutate(percent = (V1/5 / (sum(V1)/5)) * 100) %>%
  slice(1:20) %>%
  ggplot(aes(x = reorder(V2, percent),
             y = V1,
             fill = colors)) +
  geom_bar(stat = "identity",
            position = "dodge") +
  coord_flip() +
  theme_minimal() +
  labs(x = "Amount of mentions from 75,524 microarray",
       y = "Top 20 mentioned RNA types")

```



1.1.2 Regression analyses to quantify diagnosis connections

Logistic regression of binomial factoring between nodal/extranodal diagnosis and individuals labels for cell-of-origin classification and CNS relapse or systemic relapse. Regression model summary with odds ratio with 95% confidence interval to quantify how much nodal and extranodal diagnosis is associated with the cell-of-origin ABC or GCB nature in DLBCL individuals with CNS, systemic or no relapse.

```

fit_summary_table <- function(features, dependent, df, method, execute = TRUE) {
  if ( execute == TRUE ) {
    if ( method == "glm" || method == "cox" ) {
      x <- df %>%
        finalfit(dependent, features)
    } else if ( execute == "glmer" ) {
      x <- df %>%
        finalfit(dependent, features,
                  mixed, random_effect)
    }
    ## print latex table
    Hmisc:::latex(x, file = "", booktabs = TRUE, title = "")
  } else {
    cat("LaTeX summary table printed\n")
  }
}

fit_summary_table(features= c("ABCScore", "ABCClassify", "GROUP"),
                  dependent= c("Nodes"),
                  df = metadata,
                  method = "glm",
                  execute = F)

```

LaTeX summary table printed

Dependent: Nodes		EN	LN	OR (univariable)	OR (multivariable)
4	ABCscore	ABC	34 (39.5)	58 (38.7) -	-
5		GCB	36 (41.9)	67 (44.7) 1.09 (0.61-1.96, p=0.771)	0.44 (0.06-3.23, p=0.408)
6		U	16 (18.6)	25 (16.7) 0.92 (0.43-1.97, p=0.820)	0.96 (0.25-4.74, p=0.952)
1	ABCclassify	ABC	37 (43.0)	66 (44.0) -	-
2		GCB	38 (44.2)	79 (52.7) 1.17 (0.67-2.04, p=0.591)	1.61 (0.24-10.98, p=0.615)
3		U	11 (12.8)	5 (3.3) 0.25 (0.08-0.76, p=0.018)	0.52 (0.07-2.97, p=0.473)
7	GROUP	CNS DIAGNOSIS	7 (8.1)	4 (2.7) -	-
8		CNS RELAPSE CHOP or EQUIVALENT	5 (5.8)	3 (2.0) 1.05 (0.15-7.08, p=0.960)	0.97 (0.13-6.76, p=0.979)
9		CNS RELAPSE RCHOP	20 (23.3)	19 (12.7) 1.66 (0.43-7.21, p=0.470)	1.71 (0.42-7.73, p=0.461)
10		NO RELAPSE	30 (34.9)	66 (44.0) 3.85 (1.08-15.64, p=0.042)	3.40 (0.91-14.2, p=0.074)
11		NORMAL ABC CONTROL	2 (2.3)	0 (0.0) 0.00 (NA-NA, p=0.995)	0.00 (NA-NA, p=0.995)
12		NORMAL GCB CONTROL	0 (0.0)	4 (2.7) 74.56 (0.00-NA, p=0.993)	79.25 (0.00-NA, p=0.993)
13		SYSTEMIC RELAPSE NO CNS	10 (11.6)	54 (36.0) 9.45 (2.42-NA, p=0.002)	8.07 (1.98-NA, p=0.004)
14		TESTICULAR NO CNS RELAPSE	12 (14.0)	0 (0.0) 0.00 (0.00-NA, p=0.988)	0.00 (0.00-NA, p=0.988)

22 Mixed effects multilevel logistic regression model fit to find connections between individuals (CNS relapse,
 23 systemic, and no relapse) and cell-of-origin predictions (ABC, GCB likelihoods), while considering nodal
 24 and extranodal involvement in the relapse (diagnosed tissue sites with cancer invasion).

```
mixed = c("GROUP")
random_effect = c("SITE")
fit_summary_table(features= c("Prediction", "GROUP"),
                   dependent= c("Nodes"),
                   df = metadata,
                   method = "glmer",
                   execute = F)
```

LaTeX summary table printed

Dependent: Nodes		EN	LN	OR (univariable)	OR (multilevel)
9	Prediction	ABC	34 (40.5)	58 (38.7) -	-
10		GCB	36 (42.9)	67 (44.7) 1.09 (0.61-1.96, p=0.771)	-
11		U	14 (16.7)	25 (16.7) 1.05 (0.48-2.32, p=0.908)	-
1	GROUP	CNS DIAGNOSIS	7 (8.1)	4 (2.7) -	-
2		CNS RELAPSE CHOP or EQUIVALENT	5 (5.8)	3 (2.0) 1.05 (0.15-7.08, p=0.960)	0.38 (0.00-NA, p=0.989)
3		CNS RELAPSE RCHOP	20 (23.3)	19 (12.7) 1.66 (0.43-7.21, p=0.470)	0.49 (0.00-NA, p=0.988)
4		NO RELAPSE	30 (34.9)	66 (44.0) 3.85 (1.08-15.64, p=0.042)	1.70 (0.00-NA, p=0.989)
5		NORMAL ABC CONTROL	2 (2.3)	0 (0.0) 0.00 (NA, p=0.995)	0.00 (0.00-Inf, p=1.000)
6		NORMAL GCB CONTROL	0 (0.0)	4 (2.7) NA (0.00-NA, p=0.993)	285412.87 (0.00-Inf, p=0.999)
7		SYSTEMIC RELAPSE NO CNS	10 (11.6)	54 (36.0) 9.45 (2.42-42.22, p=0.002)	1.76 (0.00-NA, p=0.989)
8		TESTICULAR NO CNS RELAPSE	12 (14.0)	0 (0.0) 0.00 (0.00-NA, p=0.988)	0.00 (0.00-Inf, p=1.000)

1.2 Featured data and groups of sample cases

25 Difference in cases being indexed based on their *cell-of-origin* association subtypes using either of the
 26 following features: prediction, ABCclassify, ABCScore.
 27

```
metadata %>%
  select(Prediction, ABCclassify, ABCScore) %>%
  summary

Prediction ABCclassify ABCScore
ABC :90    ABC:101    ABC:90
GCB :99    GCB:113    GCB:99
U   :39     U  : 16    U   :41
NA's: 2
```

28 Distribution of samples with different treatments.

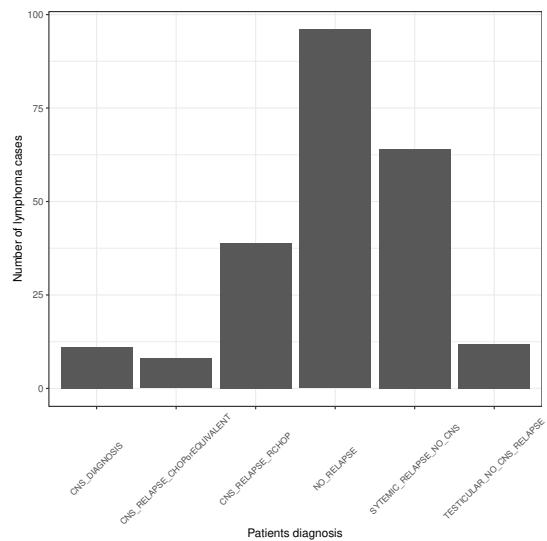
```
metadata %>%
```

```

select(GROUP) %>%
ggplot(aes(x = GROUP)) +
geom_histogram(stat = "count") +
labs(y = "Number of lymphoma cases",
x = "Patients diagnosis") +
theme_bw() +
theme(axis.text.x = element_text(vjust = .5,
angle = 45,
size = 8))

Warning: Ignoring unknown parameters: binwidth, bins, pad

```



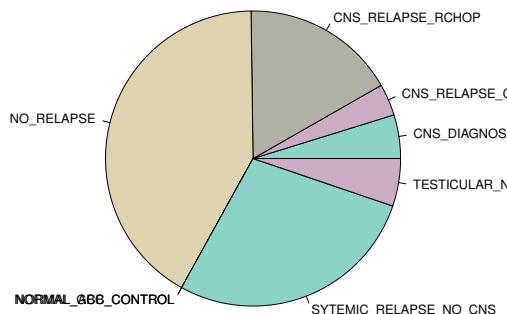
29

30 Or as a pie chart.

```

palette.pies <- brewer.pal(12, name = "Set3")
palette.pies.adj <- colorRampPalette(palette.pies)(length(unique(metadata$GROUP)))
pie(table(metadata$GROUP), col=palette.pies.adj)

```



31

32 Distribution of samples with different cells of origin subtypes.

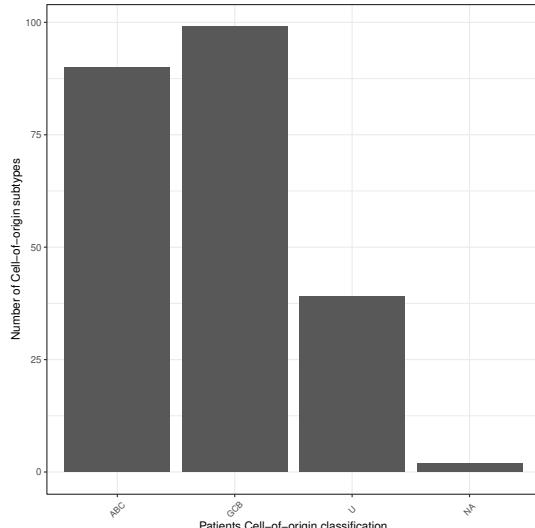
```
metadata %>%
```

```

select(Prediction) %>%
ggplot(aes(x = Prediction)) +
geom_histogram(stat = "count") +
labs(y = "Number of Cell-of-origin subtypes",
x = "Patients Cell-of-origin classification") +
theme_bw() +
theme(axis.text.x = element_text(vjust = .5,
angle = 45,
size = 8))

Warning: Ignoring unknown parameters: binwidth, bins, pad

```



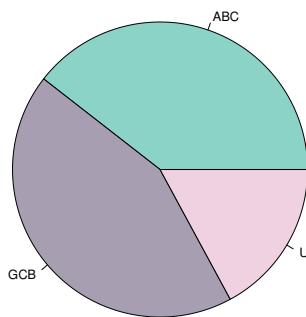
33

34 Or as pie chart.

```

palette.pies <- brewer.pal(12, name = "Set3")
palette.pies.adj <- colorRampPalette(palette.pies)(length(unique(metadata$Prediction)))
pie(table(metadata$Prediction), col=palette.pies.adj)

```



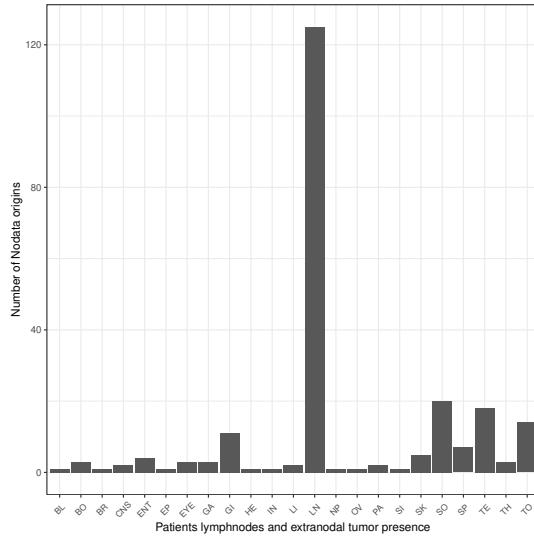
35

36 Distribution of samples with different lymphnodes and extranodal cancer metastasis.

```
par(mfrow=c(2,2))
```

```
metadata %>%
  select(SITE) %>%
  ggplot(aes(x = SITE)) +
  geom_histogram(stat = "count") +
  labs(y = "Number of Nodata origins",
       x = "Patients lymphnodes and extranodal tumor presence") +
  theme_bw() +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 8))

Warning: Ignoring unknown parameters: binwidth, bins, pad
```



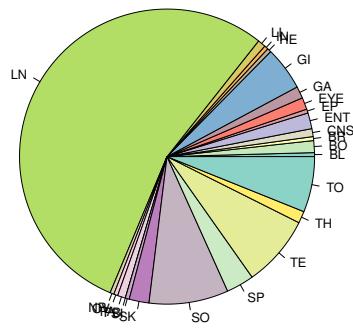
37

38 Or as a pie chart.

```

palette.pies <- brewer.pal(12, name = "Set3")
palette.pies.adj <- colorRampPalette(palette.pies)(length(unique(metadata$SITE)))
pie(table(metadata$SITE), col=palette.pies.adj)

```



39

2 Differential expression of microarray Affymetrix data

41 Genes have been fitted in a model that is based on an Empirical Bayes approach. Ranking of the genes
42 determine if they are statistically significant. Bonferroni correction is used to control the false discovery
43 rate (FDR). Moderated t-statistics, FDR, and fold change (log2) are implemented to reduce selection of
44 false positives.

- 45 • **adjpval** is the adjusted P-value to control the FDR using Bonferroni correction. **Genes selected**
46 **here based on their adjpval are also greater than or equal to the bstat threshold.**

- 47 • **avgex** is the average expression the ordinary arithmetic average of the log2-expression values for
 48 the probe, across all arrays. **Genes selected here based on their avgex are also greater than**
 49 **or equal to the bstat threshold.**
- 50 • **bstat** is the moderated t-statistics using an Empirical Bayes approach generating B-statistics scores.

```
expression <- read.table("data/summary.full.409794.txt", sep = "\t", header = T) %>%
  select(Design, Model, Bthreshold, adjPval, Category, Parameter, Transcripts) %>%
  filter(Category == "total")
summary(expression)

  Design                               Model
CNSvsNOREL      : 12    systemicRelapse      : 36
CNSvsNOREL_ABC: 12    systemicRelapseCOOprediction:108
CNSvsNOREL_EN : 12    systemicRelapseNodes     :108
CNSvsNOREL_GCB: 12
CNSvsNOREL_LN : 12
CNSvsSYST       : 12
(Other)          :180

  Bthreshold      adjPval      Category      Parameter
Min.   :-4.00      Min.   :0.1      total:252    adjpval:84
1st Qu.:-2.50      1st Qu.:0.1                avgex  :84
Median :-1.00      Median :0.1                bval   :84
Mean   :-1.25      Mean   :0.1
3rd Qu.: 0.25      3rd Qu.:0.1
Max.   : 1.00      Max.   :0.1

  Transcripts
Min.   : 0
1st Qu.: 0
Median : 6
Mean   : 398
3rd Qu.: 120
Max.   :6414
```

- 51 Number of transcripts when comparing B-statistics scores, which represent confidence in selecting each
 52 significantly expressed gene.

```
aggregate( Transcripts ~ Bthreshold, data=expression, FUN=range)

  Bthreshold Transcripts.1 Transcripts.2
1      -4           0        6414
2      -2           0         954
3       0           0        154
4       1           0         65
```

- 53 Number of transcripts when samples are classed into groups, which are based on clinical data (e.g.,
 54 cell-of-origin, CNS relapse, and nodal/extranodal tumor transmission).

```
aggregate( Transcripts ~ Model, data=expression, FUN=range)

  Model Transcripts.1 Transcripts.2
1      systemicRelapse      0        6414
2 systemicRelapseCOOprediction 0        5664
3      systemicRelapseNodes 0        4059
```

- 55 Number of transcripts found when comparing different sample cases indexed based on their clinical data.
 56

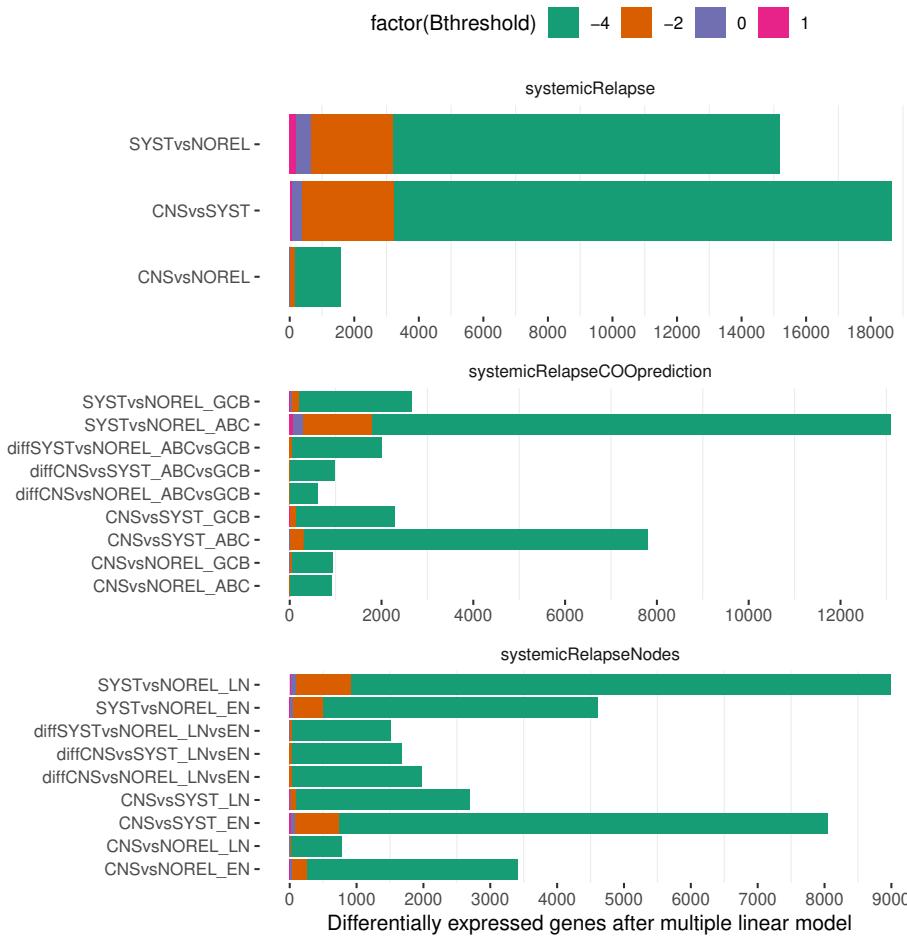
```
aggregate( Transcripts ~ Design, data=expression, FUN=range)
```

	Design	Transcripts.1	Transcripts.2
1	CNSvsNOREL	0	716
2	CNSvsNOREL_ABC	0	453
3	CNSvsNOREL_EN	0	1585
4	CNSvsNOREL_GCB	0	438
5	CNSvsNOREL_LN	0	374
6	CNSvsSYST	30	6414
7	CNSvsSYST_ABC	0	3770
8	CNSvsSYST_EN	1	3663
9	CNSvsSYST_GCB	2	1081
10	CNSvsSYST_LN	0	1302
11	diffCNSvsNOREL_ABCvsGCB	0	308
12	diffCNSvsNOREL_LNvsEN	0	977
13	diffCNSvsSYST_ABCvsGCB	0	498
14	diffCNSvsSYST_LNvsEN	0	831
15	diffSYSTvsNOREL_ABCvsGCB	0	982
16	diffSYSTvsNOREL_LNvsEN	0	738
17	SYSTvsNOREL	65	5299
18	SYSTvsNOREL_ABC	26	5664
19	SYSTvsNOREL_EN	1	2056
20	SYSTvsNOREL_GCB	0	1227
21	SYSTvsNOREL_LN	0	4059

57 Number of genes that respond to treatment, cell subtypes, and nodal transmission.

↑ Min adjusted p-value of 0.1

```
expression %>%
  ggplot(aes(
    x = Design,
    y = Transcripts,
    fill = factor(Bthreshold))) +
  theme_bw() +
  geom_bar(stat = "identity",
            position = "stack") +
  coord_flip() +
  facet_wrap(~ Model,
             ncol = 1,
             scales = "free") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "",
       y = "Differentially expressed genes after multiple linear model") +
  theme(legend.position = "top",
        strip.background = element_rect(linetype = "blank",
                                         fill = "white"),
        panel.border = element_rect(linetype = "blank",
                                    fill = NA),
        panel.grid.major = element_line(linetype = "blank")) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10))
```



58

2.1 Cleaning and removing non-essential genes

59

Subsetting the data by reducing the number of gene profiles improves interpretation and reduces noise. It is well established that many machine learning models used for classification can be sensitive to high number of *irrelevant* genes, others like support vector machines and random forests are less so (Statnikov 2008).

60

Each array contains probes of 75,523 functional and non-functional RNAs. Either ncRNA, mRNA, and non annotated genes. More than 53.32% of the probes are non-coding. For interpretation purpose, ncRNAs profiles were discarded before fitting the expressions. In addition, the variation from the mean of each transcript was assessed and the spread of expression were all used to discard top and bottom variants. Individual genes that vary widely from the mean of the array were removed thus reducing the spread of the expression across profiles. Transcripts with potential biased high expressions were thus flagged and discarded thus improving correlation of other transcripts. Subsetting was done after normalization of all datasets, all arrays. This would reduce technical errors appearing significant when comparing arrays between each others. Data was transformed (standardization protocol) before calculating means and variances. This helps a better signal recovery from a large dataset with potential expression bias.

61

2.1.1 Variance optimization for each array

62

Full probe list accounting for 75,523 genes (red horizontal line). The full line represents the variance after being adjusted by iteratively discarding top/low variant expression profiles. The dotted line represent the original variance before discarding genes.

63

The graph below shows that by discarding highly variant expressions and selecting only the top 1613 genes for example, the mean variance of the whole array (0.27) is higher than a ranked subset of 10,811 (0.09). Ideally, the reduction of the data is on both, the mean variance and mean standard deviation of the whole array.

[†] σ^2 is the average of the squared differences from the μ

[†] Each array correspond to a DLBCL case

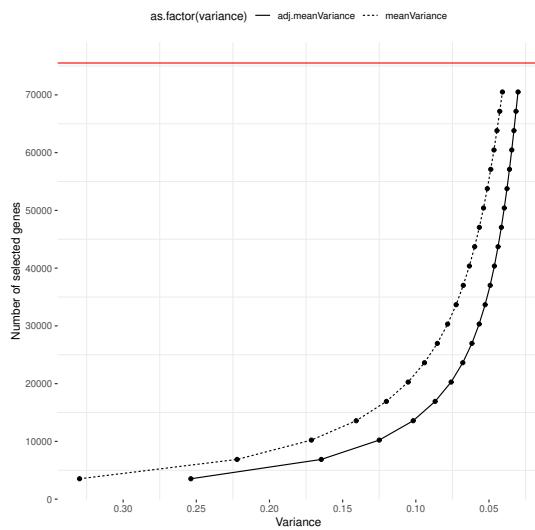
[†] The smaller the variance, the better

```
read.table("./data/summary.139102.adjusted.means.subsetting.txt", header = T) %>%
```

```

select(dimension, meanVariance, adj.meanVariance) %>%
gather("variance", "count", 2:3) %>%
ggplot(aes(x = count,
           y = dimension)) +
theme_bw() +
geom_line(aes(linetype = as.factor(variance))) +
geom_point() +
scale_x_continuous(trans = "reverse",
                   breaks = scales::pretty_breaks(n = 10)) +
scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
geom_hline(aes(yintercept = 75523), colour = "red") +
labs(y = "Number of selected genes",
     x = "Variance") +
theme(legend.position = "top",
      strip.background = element_rect(linetype = "blank",
                                       fill = "white"),
      panel.border = element_rect(linetype = "blank",
                                   fill = NA),
      panel.grid.major = element_line(linetype = "blank"))

```



82
83 Same plot description as above however we removed ncRNA which account for 53.32% of the probes.
84 The total number of transcripts is now 35,253 (46%, **red horizontal line**). The **blue horizontal line** repre-
85 sent the threshold that was selected for subsequent analysis.
86 By discarding 1198 transcripts from the 35,253 the top outliers with high variance are not included in the
87 clustering process. More rare expression signals will get distinguished. Also, the size of the dataset was
88 reduced to 29,207 by removing transcripts with little deviation from the mean of each array. The total
89 number of transcripts by array was kept above 25k to increase the sizes of the clusters (modules and
90 networks) in later analyses. For example, network analysis on 20k transcripts generated network sizes
91 between 200 and 500. At 29k networks have a total size over 700 nodes.

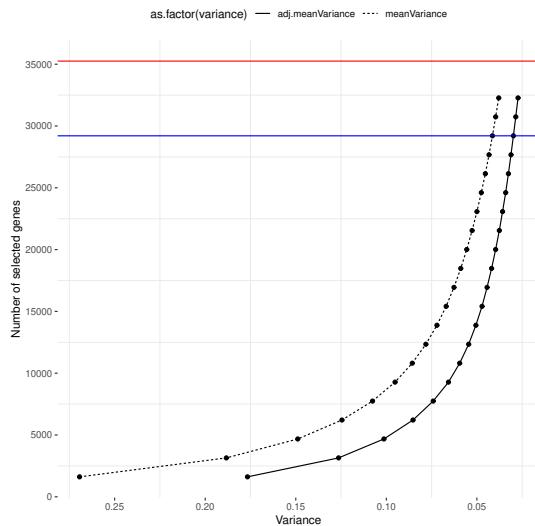
↑ 29,207 genes were selected
for clustering and nets

```
read.table("./data/summary.149317.adjusted.means.subsetting.txt", header = T) %>%
```

```

92 select(dimension, meanVariance, adj.meanVariance) %>%
93   gather("variance", "count", 2:3) %>%
94   ggplot(aes(x = count,
95             y = dimension)) +
96   theme_bw() +
97   geom_line(aes(linetype = as.factor(variance))) +
98   geom_point() +
99   scale_x_continuous(trans = "reverse",
100           breaks = scales::pretty_breaks(n = 8)) +
101   scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
102   geom_hline(aes(yintercept = 35253), color = "red") +
103   geom_hline(aes(yintercept = 29207), color = "blue") +
104   labs(y = "Number of selected genes",
105         x = "Variance") +
106   theme(legend.position = "top",
107         strip.background = element_rect(linetype = "blank",
108                                         fill = "white"),
109         panel.border = element_rect(linetype = "blank",
110                                         fill = NA),
111         panel.grid.major = element_line(linetype = "blank"))

```



92

93 2.1.2 Standard deviation optimization for each array

94 The spread of the gene expression scores is dependent on their variance, their deviation from each array's
95 mean (population mean). By removing potentially noisy expressions we are reducing the spread of the
96 arrays numbers, hence improving recognition of rare gene regulations. Below, the plot shows how the
97 standard deviation, **spread** of the data is getting smaller the more we discard genes with high and low
98 variance.

99 All array probes with all RNAs.

↑ Best if small spread between
2 SDs

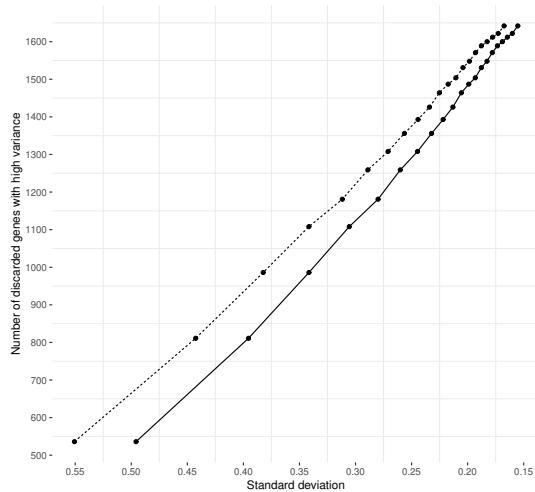
```
read.table("./data/summary.139102.adjusted.means.subsetting.txt", header = T) %>%
```

```

select(discarded, meanSD, adj.meanSD) %>%
gather("sd", "count", 2:3) %>%
ggplot(aes(x = count,
           y = discarded)) +
theme_bw() +
geom_line(aes(linetype = as.factor(sd))) +
geom_point() +
scale_x_continuous(trans = "reverse",
                    breaks = scales::pretty_breaks(n = 8)) +
scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
labs(y = "Number of discarded genes with high variance",
     x = "Standard deviation") +
theme(legend.position = "top",
      strip.background = element_rect(linetype = "blank",
                                       fill = "white"),
      panel.border = element_rect(linetype = "blank",
                                   fill = NA),
      panel.grid.major = element_line(linetype = "blank"))

```

as.factor(sd) — adj.meanSD meanSD



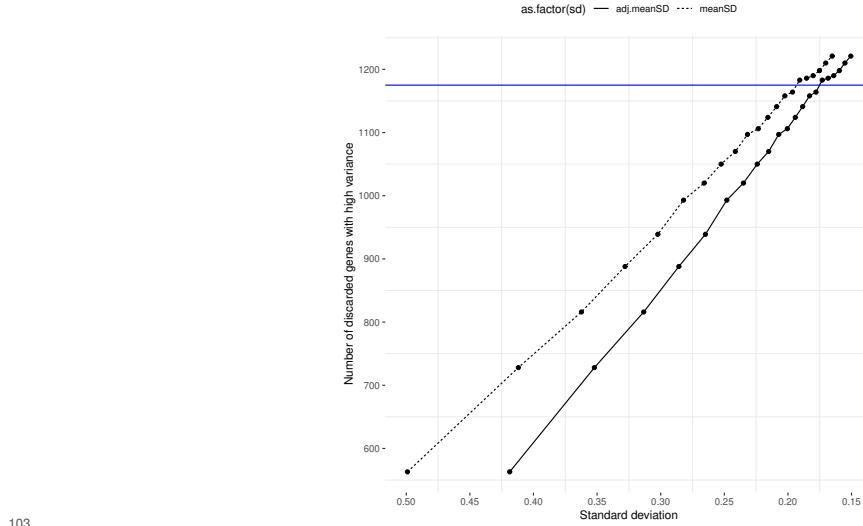
100

101 Without the ncRNAs. Blue horizontal line is the number of genes discarded when variance shrinking was
102 used to eliminate biased genes.

```

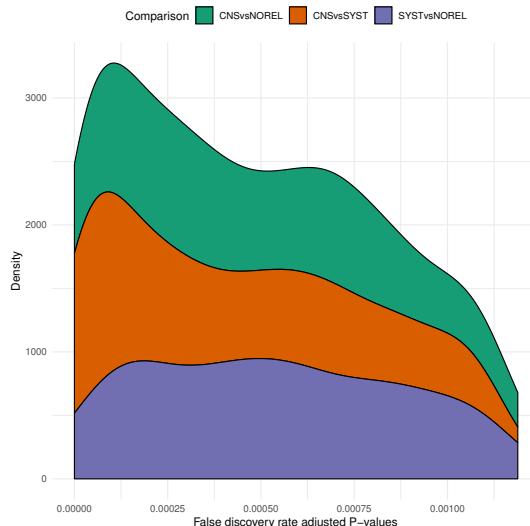
read.table("./data/summary.149317.adjusted.means.subsetting.txt", header = T) %>%
  select(discarded, meanSD, adj.meanSD) %>%
  gather("sd", "count", 2:3) %>%
  ggplot(aes(x = count,
             y = discarded)) +
  theme_bw() +
  geom_line(aes(linetype = as.factor(sd))) +
  geom_point() +
  geom_hline(aes(yintercept = 1175), colour = "blue") +
  scale_x_continuous(trans = "reverse",
                     breaks = scales::pretty_breaks(n = 8)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
  labs(y = "Number of discarded genes with high variance",
       x = "Standard deviation") +
  theme(legend.position = "top",
        strip.background = element_rect(linetype = "blank",
                                         fill = "white"),
        panel.border = element_rect(linetype = "blank",
                                    fill = NA),
        panel.grid.major = element_line(linetype = "blank"))

```



103
 104 **2.2 Distribution of p-values in pairwise comparisons**
 105 Distribution of p-values in pairwise comparisons between individuals recognized as CNS relapse, sys-
 106 temic, and did not relapse.

```
pvals <- read.table("./data/summary.lmfit.fdrAdjPval.txt", header = TRUE, fill = TRUE)
pvals %>%
  filter(Contrast == "systemicRelapse") %>%
  ggplot(aes(x = Pval,
             fill = Comparison)) +
  geom_density(position = "stack") +
  theme_minimal() +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")
```



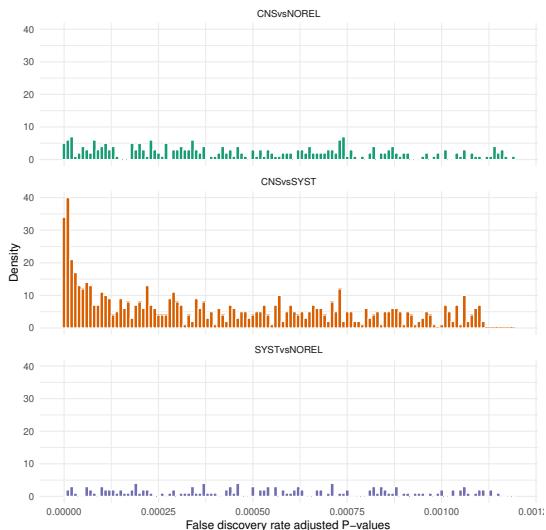
107
 108 Distribution of p-values in pairwise comparisons between individuals recognized as CNS relapse, sys-
 109 temic, and did not relapse.

```
pvals %>%
```

```

filter(Contrast == "systemicRelapse") %>%
ggplot(aes(x = Pval,
           fill = Comparison)) +
  geom_histogram(binwidth = 10^-5,
                 col=I("white")) +
  theme_minimal() +
  facet_wrap(~ Comparison,
             ncol = 1) +
  theme(legend.position = "none") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")

```

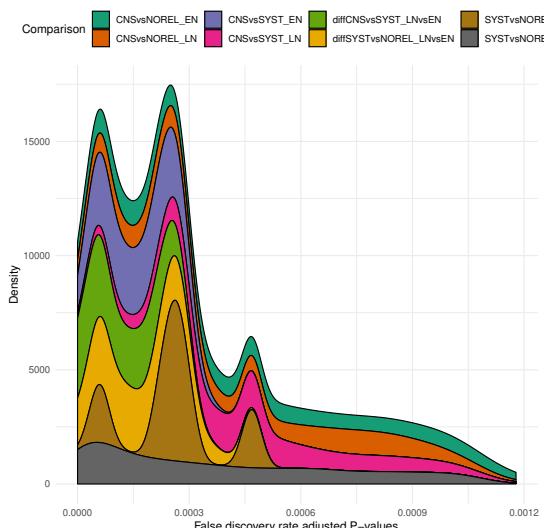


110
111 Distribution of p-values in pairwise comparisons between individuals recognized with lymph nodes and
112 extranodal implication.

```

pvals %>%
  filter(Contrast == c("systemicRelapseNodes")) %>%
  ggplot(aes(x = Pval,
             fill = Comparison)) +
  geom_density(position = "stack") +
  theme_minimal() +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")

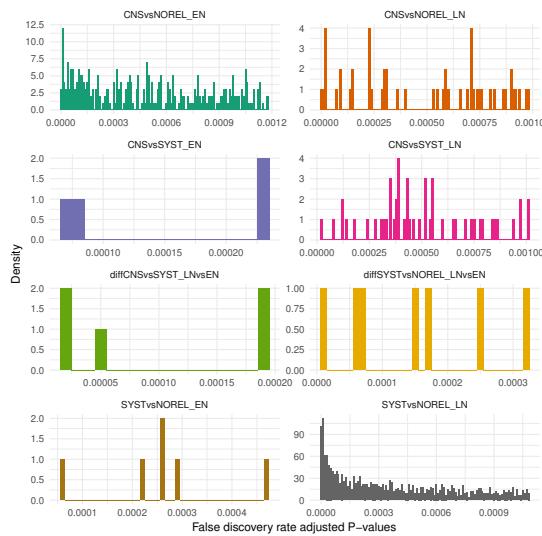
```



113

114 Distribution of p-values in pairwise comparisons between individuals recognized with lymph nodes and
115 extranodal implication.

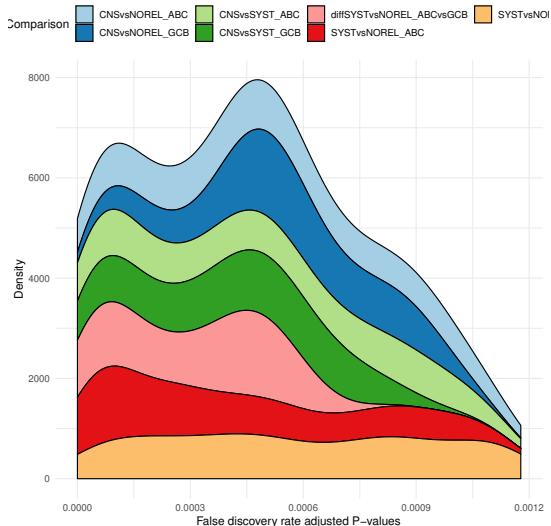
```
pvals %>%
  filter(Contrast == "systemicRelapseNodes") %>%
  ggplot(aes(x = Pval,
              fill = Comparison)) +
  geom_histogram(binwidth = 10^-5) +
  theme_minimal() +
  facet_wrap(~ Comparison,
             ncol = 2, scales = "free") +
  theme(legend.position = "none") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")
```



116
117 Distribution of p-values in pairwise comparisons between individuals recognized with cell-of-origin based
118 on ABC or GCB.

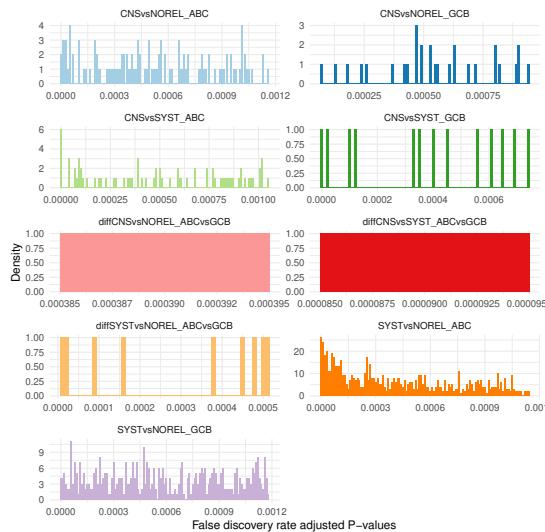
```
pvals %>%
  filter(Contrast == c("systemicRelapseCOOprediction")) %>%
  ggplot(aes(x = Pval,
              fill = Comparison)) +
  geom_density(position = "stack") +
  theme_minimal() +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Paired") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")
```

Warning: Groups with fewer than two data points have been dropped.
Warning: Groups with fewer than two data points have been dropped.
Warning: Removed 2 rows containing missing values (position_stack).



119
120 Distribution of p-values in pairwise comparisons between individuals recognized with cell-of-origin based
121 on ABC or GCB.

```
pvals %>%
  filter(Contrast == "systemicRelapseCOOprediction") %>%
  ggplot(aes(x = Pval,
             fill = Comparison)) +
  geom_histogram(binwidth = 10^-5) +
  theme_minimal() +
  facet_wrap(~ Comparison,
             ncol = 2, scales = "free") +
  theme(legend.position = "none") +
  scale_fill_brewer(palette = "Paired") +
  labs(x = "False discovery rate adjusted P-values",
       y = "Density")
```



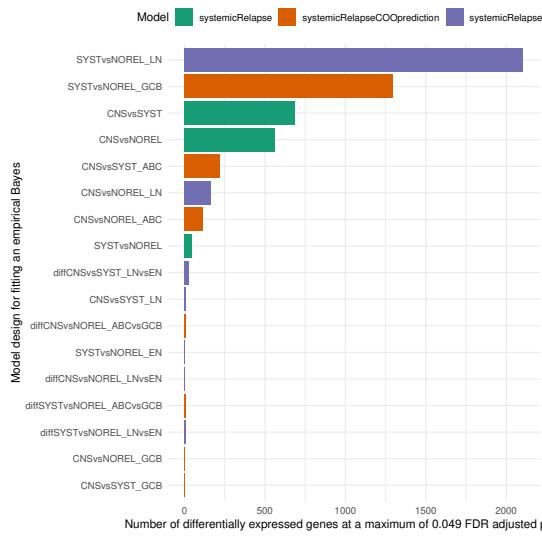
122
123 Across all contrasts class comparison, the number of genes significantly expressed at FDR multi test
124 corrections.

```
read.table("./data/summary.lmfit.bval.txt", header = TRUE, fill = TRUE) %>%
```

```

filter(Model == c("systemicRelapse", "systemicRelapseNodes", "systemicRelapseCOOprediction")) %>%
  ggplot(aes(x = reorder(Design, Transcripts),
             y = Transcripts,
             fill = Model)) +
  geom_bar(stat = "identity",
            position = "dodge") +
  coord_flip() +
  theme_minimal() +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Dark2") +
  labs(y = "Number of differentially expressed genes at a maximum of 0.049 FDR adjusted p-value",
       x = "Model design for fitting an empirical Bayes")

```



125

3 Clustering and network analyses

126

The number of clusters and modules per networks are assigned by designing first a similarity matrix between differentially expressed gene for any two conditions (eg., relapse vs no relapse individual cases). An adjacency matrix is then constructed by weighting the previously inferred measures. The data were transformed to increase the correlation coefficient therefore improving detection of strong correlated patterns. (Example of the strength of data transformation and correlation, visit the following [online page](#)).

127

The final methods selected were best according to our output. Convergence and reproducibility of each trend in gene number per module or scale of module per network helped narrow down our selection criteria for which method to incorporate in our clustering technique. The best configuration involve a Pearson correlation at stringent thresholds. Hellinger expression data standardization was better than other methods in helping to estimate similarity between genes. The correlation power was set to 6 at an overall threshold of recall of 0.5 and a complete clustering distribution to improve estimating adjacency distribution.

128

129

130

131

132

133

134

135

136

137

138

139

[†]Overfitting is a source of bias.

140

141

142

143

144

145

146

147

148

149

150

151

- **MaxEdgesPerGene**, maximum number of correlations per genes
- **NbNodes**, number of genes found for each edge connection bracket
- **Normalization**, method that focuses on creating complete clusters. We tested methods ranging from Complete clustering, Average, and Ward. [Each method is detailed here](#). Only Complete clustering was retained. All other methods overfitted the data.
- **Correlation**, finding ranges from linear to non-linear trends. We tested Pearson and Spearman correlation.
- **Standardization**, data transformation method. We tested transformation by Hellinger, Standardize, Range, and Logarithmic scaling. [Each method is detailed here](#).
- **MaxGenePerModule**, how many genes assigned by cluster (module)
- **SimilaritySize**, number of initial differentially expressed genes
- **EdgeThreshold**, parameter to limit the weight of the edges

- **CorrelationPower**, power transformation of the data

```
ns <- read.table("./data/networks.summary.437006.txt", header = T)
summary(ns)

MaxEdgesPerGene      NbNodes      Normalization   Correlation
Min.    : 1      Min.    : 0      complete:4140    pearson:4140
1st Qu.: 92     1st Qu.: 4
Median  :183      Median :234
Mean    :183      Mean   :194
3rd Qu.:274     3rd Qu.:338
Max.    :365      Max.   :366

Standardization MaxGenesPerModule SimilaritySize EdgeThreshold
hellinger       Min.    : 5.0      Min.    :366      Min.    :0.5
range          1st Qu.: 5.0      1st Qu.:366      1st Qu.:0.5
standardize:1380 Median  :20.0      Median  :366      Median  :0.5
                           Mean   :19.3      Mean   :366      Mean   :0.5
                           3rd Qu.:33.0      3rd Qu.:366      3rd Qu.:0.5
                           Max.   :33.0      Max.   :366      Max.   :0.5

CorrelationPower
Min.    :2
1st Qu.:3
Median  :4
Mean    :4
3rd Qu.:5
Max.    :6
```

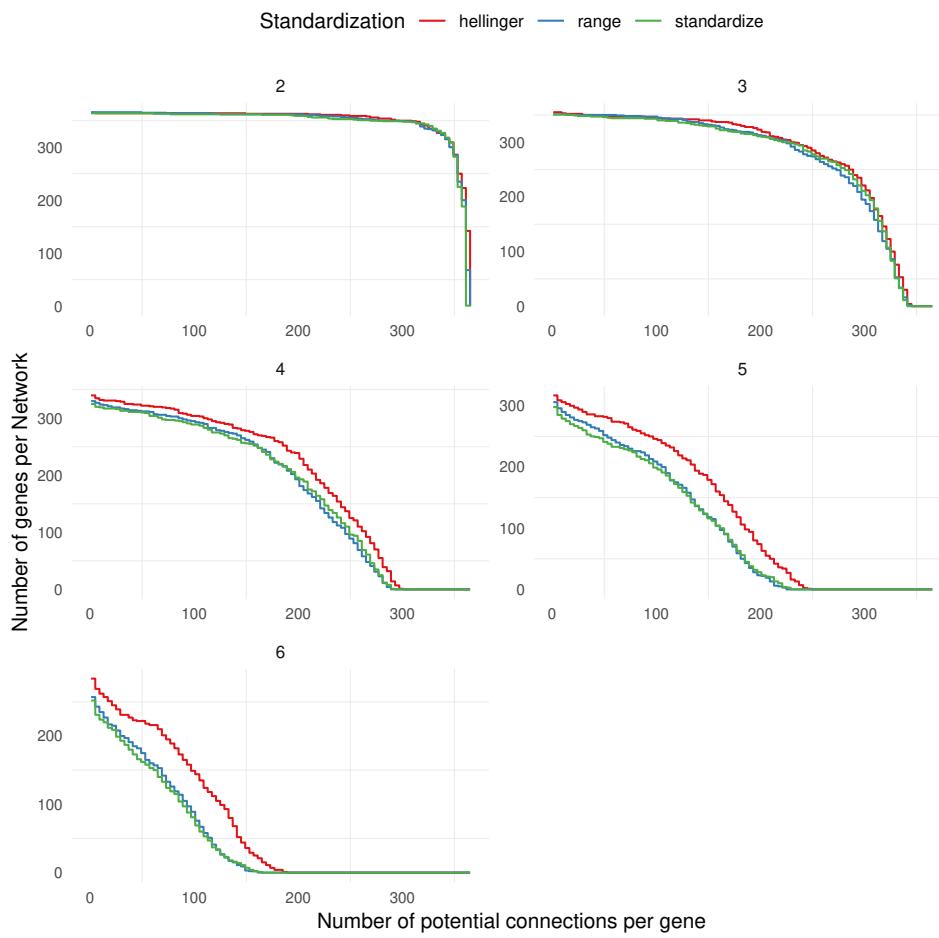
↳ Effect of correlation methods is seen on module content

153 Difference between methods used for network inference. Are we able to generate convergence of the
 154 output of all iterations across all methods? After many trials the below two graphs (in this section) rep-
 155 resent the number of networks, genes, clustered, and modules for the selected configuration (between
 156 normalization, clustering, and data transformation).

↳ Definitive graphs

```
networks.comparison <- function(data = ns) {
  data %>%
    ggplot(aes(
      x = MaxEdgesPerGene,
      y = NbNodes,
      fill = Standardization)) +
    theme_minimal() +
    geom_step(aes(color = Standardization),
              stat = "identity") +
    facet_wrap(~ CorrelationPower,
               ncol = 2,
               scales = "free") +
    scale_color_brewer(type = "qual", palette = 6) +
    labs(x = "Number of potential connections per gene",
         y = "Number of genes per Network") +
    theme(legend.position = "top",
          strip.background = element_rect(linetype = "blank",
                                         fill = "white"),
          panel.border = element_rect(linetype = "blank",
                                      fill = NA),
          panel.grid.major = element_line(linetype = "blank"),
          axis.text.x = element_text(size = 8),
          axis.text.y = element_text(size = 8))
}

networks.comparison(data = ns)
```



157
 158 Showing the number of modules per network and the number of genes per module. Each module contains
 159 differing number of nodes based on their correlation strength. Each cluster contains at least one module.
 160 Each network contains at least one cluster. One module can be assigned to nodes that belong to more
 161 than one cluster. The Lowess curves show if the trend in the data is linear or not. The wave around
 162 Lowess curves represents the level of confidence of the data points (the narrower the interval the better,
 163 less variability = more accuracy).

[†]Points=iterations. With less iterations comes high variability of the curve

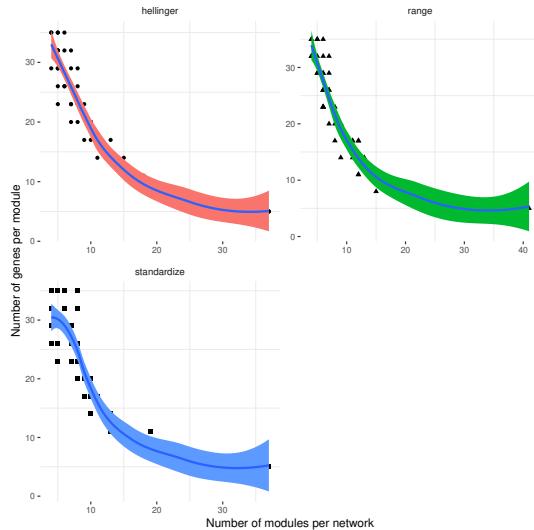
```
ms <- read.table("./data/modules.summary.437006.txt", header = TRUE)
```

```

modules.comparison <- function(data = ms) {
  data %>%
    ggplot(aes(
      x = NbModules,
      y = MaxGenesPerModule,
      fill = Standardization)) +
    theme_bw() +
    geom_point(aes(shape = Standardization)) +
    scale_color_brewer(type = "qual", palette = 6) +
    labs(x = "Number of modules per network",
         y = "Number of genes per module") +
    facet_wrap(~ Standardization,
              ncol = 2,
              scales = "free") +
    theme(legend.position = "none",
          strip.background = element_rect(linetype = "blank",
                                           fill = "white"),
          panel.border = element_rect(linetype = "blank",
                                       fill = NA),
          panel.grid.major = element_line(linetype = "blank"),
          axis.text.x = element_text(size = 8),
          axis.text.y = element_text(size = 8)) +
    geom_smooth(method = 'loess', alpha = 1)
}

modules.comparison(data = ms)

```



164

3.1 Network analysis for Spearman-related correlations (relaxed)

Thresholds based on the Empirical Bayes approach to rank genes and determine if a gene is significantly expressed by shrinking the sample variance [Smyth 2004](#). Limma implementation.

↑ Different iterations to test which normalization & clustering are best

- 168 • **Average Expression:** 5
- 169 • **Adjusted P-value:** equal or less than 0.045
- 170 • **Log Fold Change:** 1
- 171 • **B-statistics:** 1.5

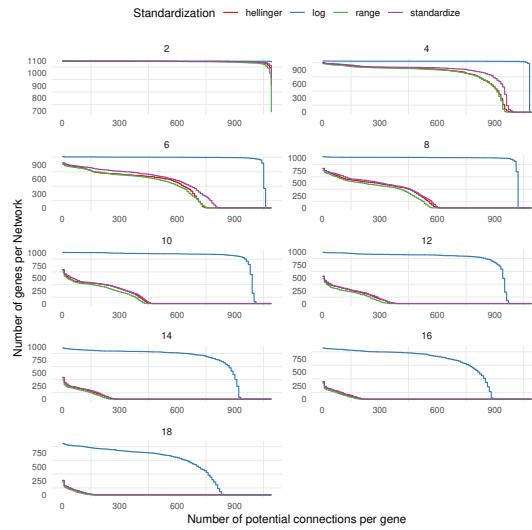
3.1.1 Nodal versus extra-nodal lymphoma

173 Genetic networks from differentially expressed genes selected by comparing sample cases with nodal
174 and extranodal lymphoma.

```

ns <- read.table("./data/networks.summary.104859.txt", header = TRUE)
networks.comparison(ns)

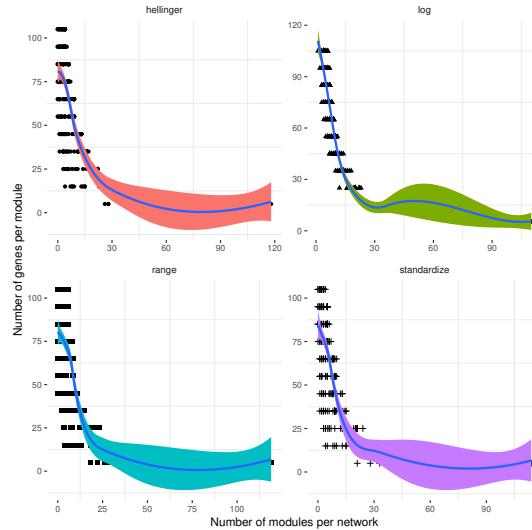
```



175

176 Showing the number of modules per network and the number of genes per module.

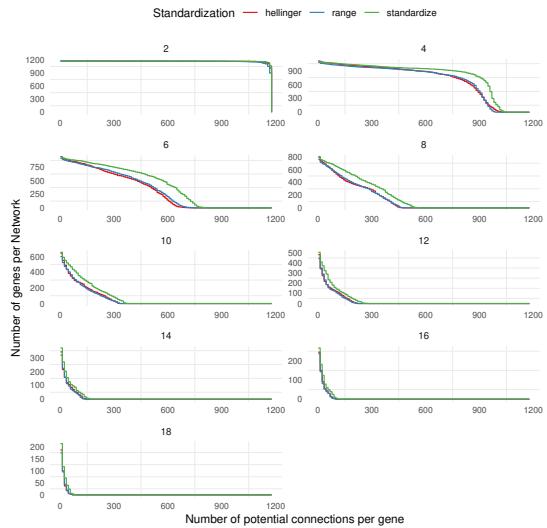
```
ms <- read.table("./data/modules.summary.104859.txt", header = TRUE)
modules.comparison(ms)
```



177

178 **3.1.2 Relapsed versus no CNS relapsed cases**179 Genetic networks from differentially expressed genes selected by comparing sample cases with systemic
180 or no CNS relapse lymphoma.

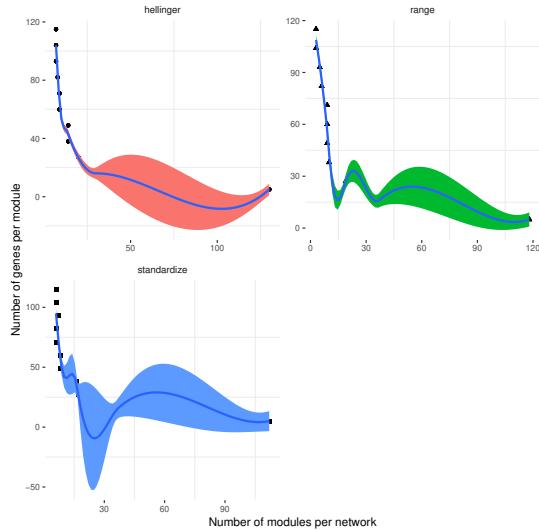
```
ns <- read.table("./data/networks.summary.114018.txt", header = TRUE)
networks.comparison(ns)
```



181
182

Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.114018.txt", header = TRUE)
modules.comparison(ms)
```

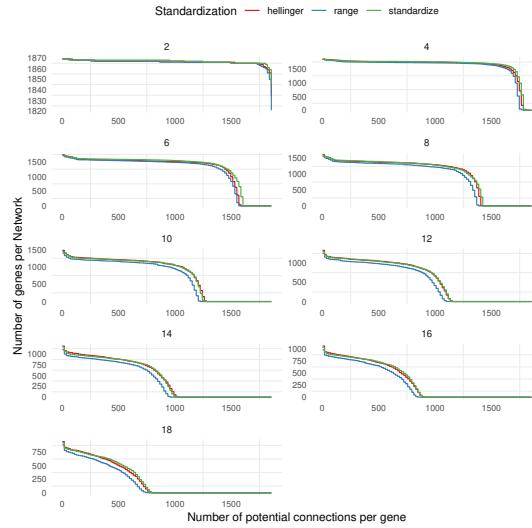


183
184

3.1.3 Lymphoma cases classified by Cell-of-origin subtypes

185 Genetic networks from differentially expressed genes selected by comparing sample cases with systemic
186 or no CNS relapse lymphoma.

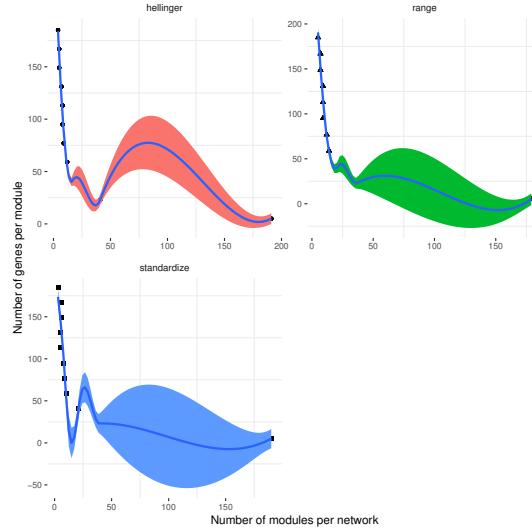
```
ns <- read.table("./data/networks.summary.114017.txt", header = TRUE)
networks.comparison(ns)
```



187
188

Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.114017.txt", header = TRUE)
modules.comparison(ms)
```



189

3.2 Network analysis for Pearson-related correlations (relaxed)

Thresholds based on the Empirical Bayes approach to rank genes and determine if a gene is significantly expressed. Limma implementation.

[†]With pearson, we can only raise the data to power 10. All are discarded after 10.

190

- **Average Expression:** 5

191

- **Adjusted P-value:** equal or less than 0.045

192

- **Log Fold Change:** 1

193

- **B-statistics:** 1.5

194

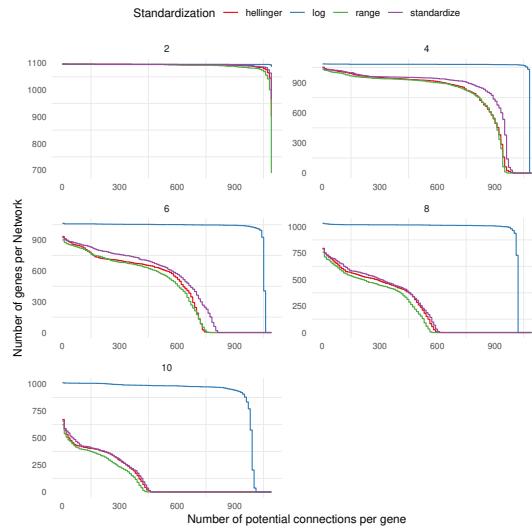
3.2.1 Nodal versus extra-nodal lymphoma

195

Genetic networks from differentially expressed genes selected by comparing sample cases with nodal and extranodal lymphoma.

196

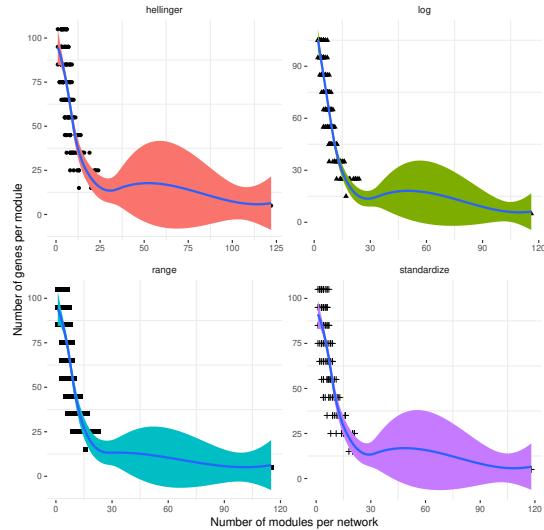
```
ns <- read.table("./data/networks.summary.104862.txt", header = TRUE)
networks.comparison(ns)
```



200
201 Showing the number of modules per network and the number of genes per module.

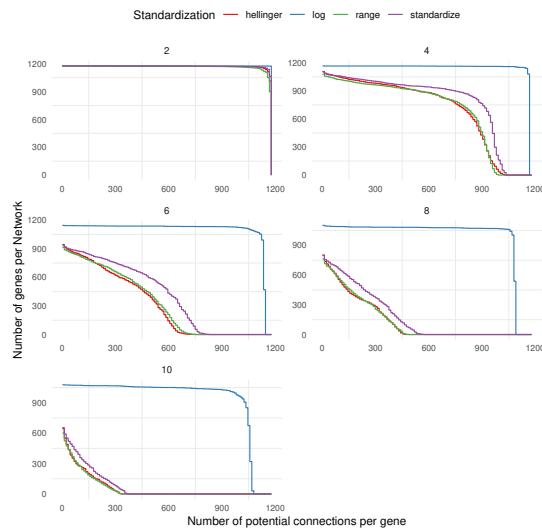
```
ms <- read.table("./data/modules.summary.104862.txt", header = TRUE)
modules.comparison(ms)
```

^aSince Lowess ranks by confidence, Log transformation seems the best, ie, low variability. For this, Log is removed from further tests.



202
203 **3.2.2 Relapsed versus no CNS relapsed cases**
204 Genetic networks from differentially expressed genes selected by comparing sample cases with systemic
205 or no CNS relapse lymphoma.

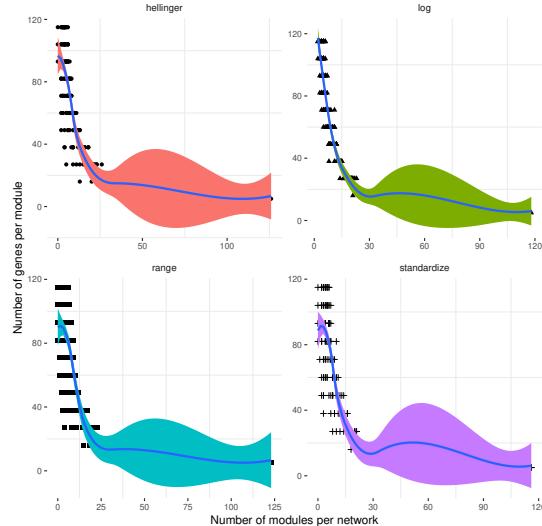
```
ns <- read.table("./data/networks.summary.104863.txt", header = TRUE)
networks.comparison(ns)
```



206

207 Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.104863.txt", header = TRUE)
modules.comparison(ms)
```

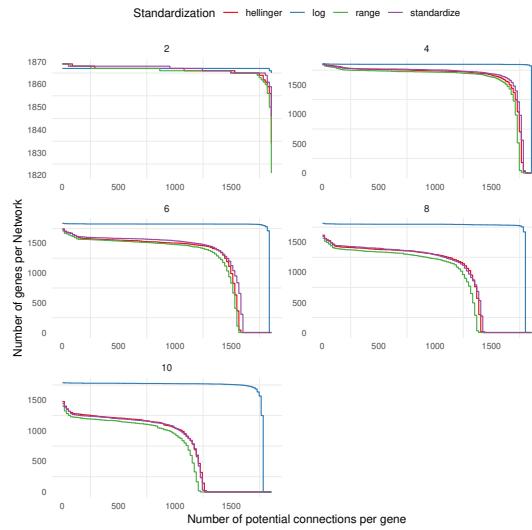


208

3.2.3 Lymphoma cases classified by Cell-of-origin subtypes

Genetic networks from differentially expressed genes selected by comparing sample cases with cell of origin classification based on ABC or GCB subtypes.

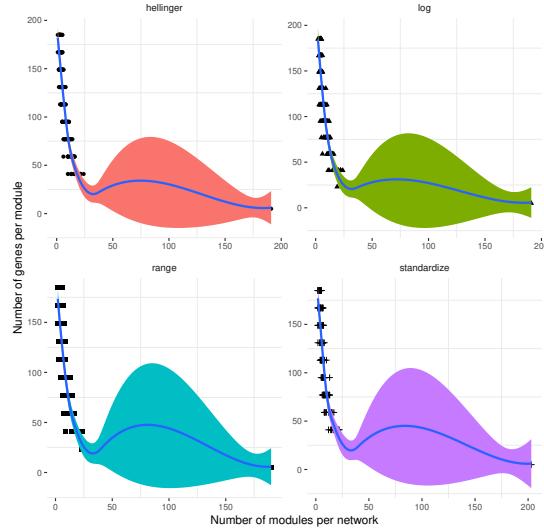
```
ns <- read.table("./data/networks.summary.104864.txt", header = TRUE)
networks.comparison(ns)
```



212
213

Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.104864.txt", header = TRUE)
modules.comparison(ms)
```



214
215
216
217

3.3 Network analysis for Spearman-related correlations (stringent)

Thresholds based on the Empirical Bayes approach to rank genes and determine if a gene is significantly expressed. Limma implementation.

[†]Same analysis with more stringent parameters

218
219
220
221

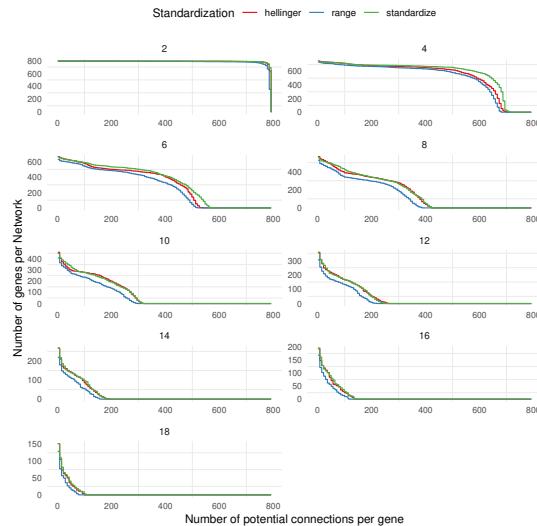
- **Average Expression:** 10
- **Adjusted P-value:** equal or less than 0.030
- **Log Fold Change:** 1
- **B-statistics:** 2

222
223
224

3.3.1 Nodal versus extra-nodal lymphoma

Genetic networks from differentially expressed genes selected by comparing sample cases with nodal and extranodal lymphoma.

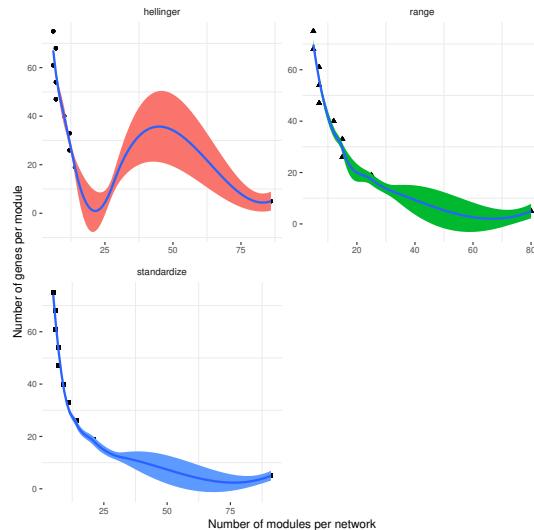
```
ns <- read.table("./data/networks.summary.119759.txt", header = TRUE)
networks.comparison(ns)
```



225

226 Showing the number of modules per network and the number of genes per module.

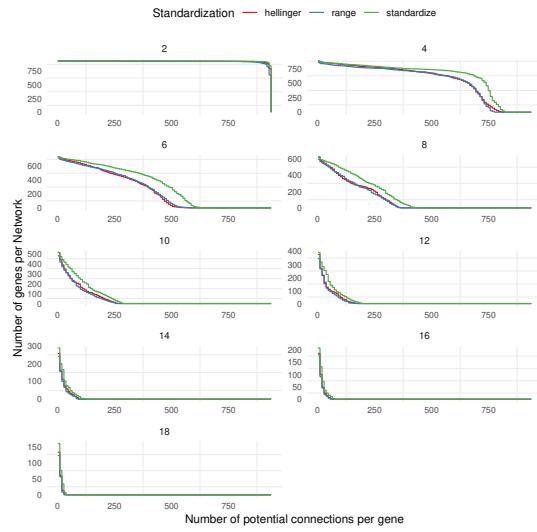
```
ms <- read.table("./data/modules.summary.119759.txt", header = TRUE)
modules.comparison(ms)
```



227

228 **3.3.2 Relapsed versus no CNS relapsed cases**229 Genetic networks from differentially expressed genes selected by comparing sample cases with systemic
230 or no CNS relapse lymphoma.

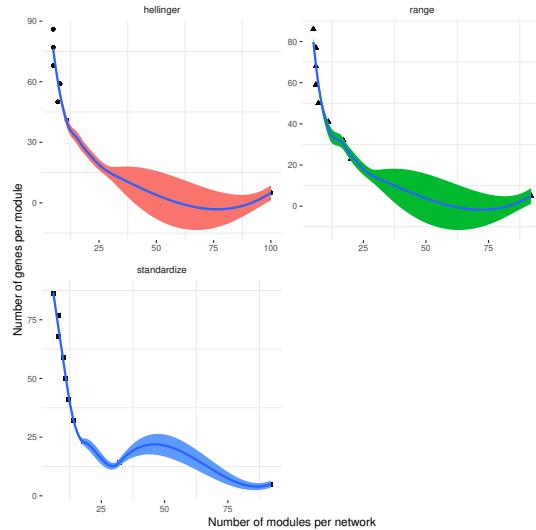
```
ns <- read.table("./data/networks.summary.119760.txt", header = TRUE)
networks.comparison(ns)
```



231

232 Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.119760.txt", header = TRUE)
modules.comparison(ms)
```

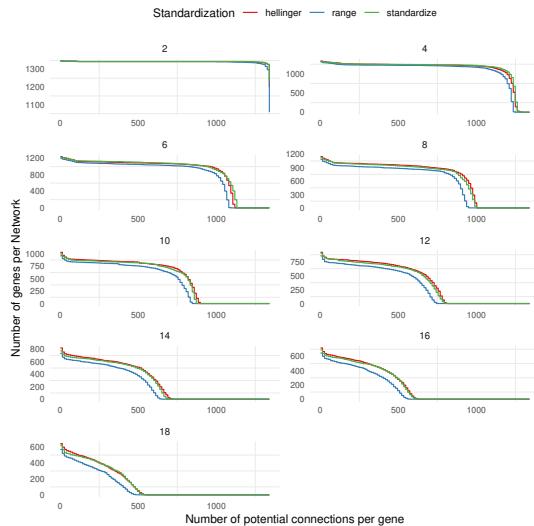


233

3.3.3 Lymphoma cases classified by Cell-of-origin subtypes

Genetic networks from differentially expressed genes selected by comparing sample cases with systemic or no CNS relapse lymphoma.

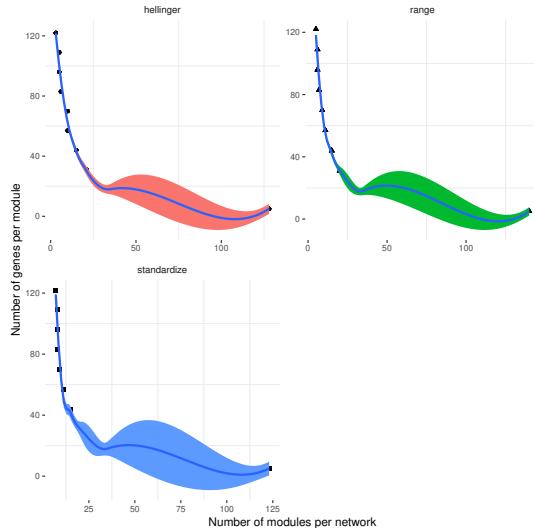
```
ns <- read.table("./data/networks.summary.119758.txt", header = TRUE)
networks.comparison(ns)
```



237

238 Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.119758.txt", header = TRUE)
modules.comparison(ms)
```



239

3.4 Network analysis for Pearson-related correlations (stringent)

Thresholds based on the Empirical Bayes approach to rank genes and determine if a gene is significantly expressed. Limma implementation.

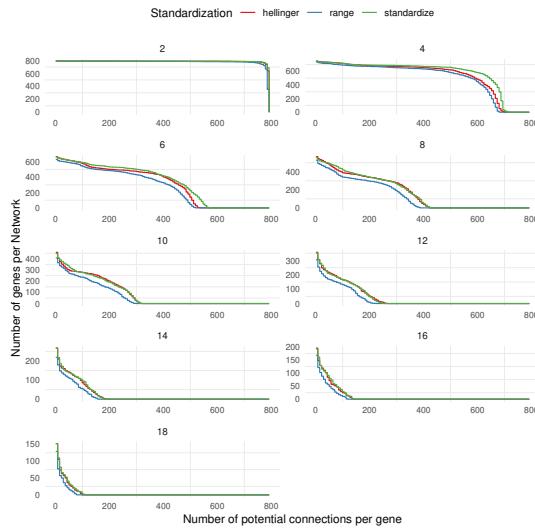
[†]Same analysis with more stringent parameters

240 • **Average Expression:** 10241 • **Adjusted P-value:** equal or less than 0.030242 • **Log Fold Change:** 1243 • **B-statistics:** 2

3.4.1 Nodal versus extra-nodal lymphoma

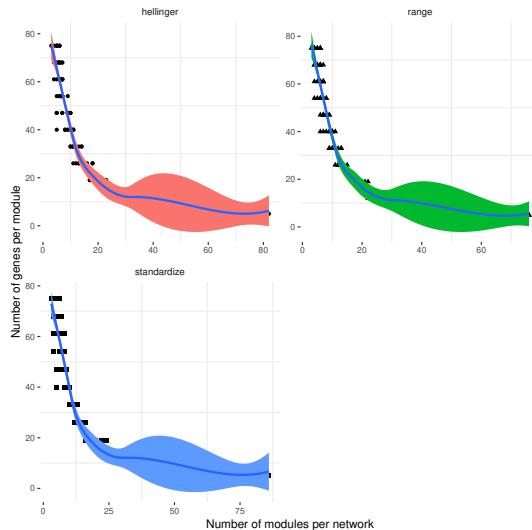
Genetic networks from differentially expressed genes selected by comparing sample cases with nodal and extranodal lymphoma.

```
ns <- read.table("./data/networks.summary.119755.txt", header = TRUE)
networks.comparison(ns)
```



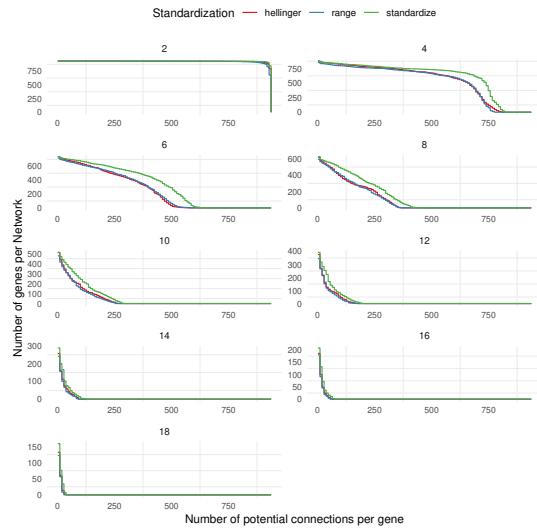
250
251 Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.119755.txt", header = TRUE)
modules.comparison(ms)
```



252
253 **3.4.2 Relapsed versus no CNS relapsed cases**
254 Genetic networks from differentially expressed genes selected by comparing sample cases with systemic
255 or no CNS relapse lymphoma.

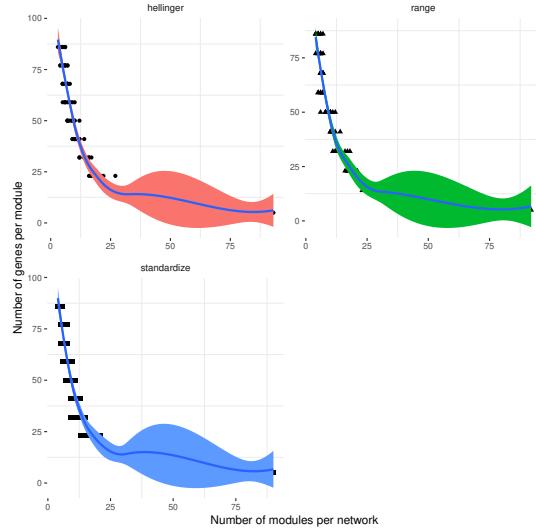
```
ns <- read.table("./data/networks.summary.119754.txt", header = TRUE)
networks.comparison(ns)
```



256
257

Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.119754.txt", header = TRUE)
modules.comparison(ms)
```

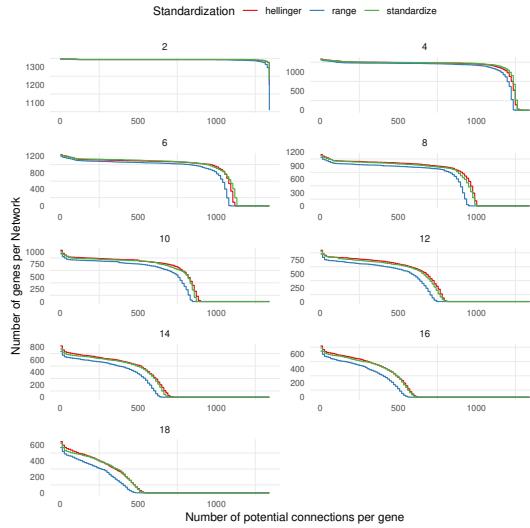


258
259
260
261

3.4.3 Lymphoma cases classified by Cell-of-origin subtypes

Genetic networks from differentially expressed genes selected by comparing sample cases with cell of origin classification based on ABC or GCB subtypes.

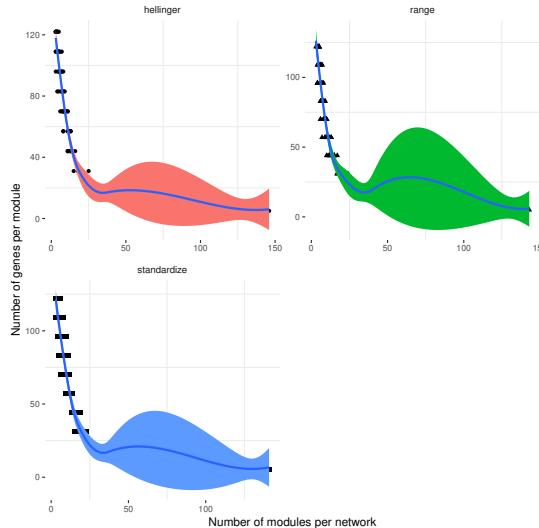
```
ns <- read.table("./data/networks.summary.119757.txt", header = TRUE)
networks.comparison(ns)
```



262

263 Showing the number of modules per network and the number of genes per module.

```
ms <- read.table("./data/modules.summary.119757.txt", header = TRUE)
modules.comparison(ms)
```



264

4 Machine Learning

Machine learning models were used for classification of cases into systemic relapse of DLBCL, CNS relapse or no relapse. Data are gene expression from Affymetrix arrays of 240 individuals with a form of DLBCL. Subsets of the whole number of microarray probes will be used for classification.

265

4.1 Regularization

266
267
268
269
270
271
272
273
274
275
276
277
278
279
280

Least absolute shrinkage and selection operator (LASSO) was used for dimension reduction. Gene expression profiles were extracted from networks with significant connectivity. Subset selection using lasso, penalizes genes based on coefficient estimates, to increase accuracy of classification. Briefly, cases are assigned to either diagnosis category, systemic relapse (SYST), CNS relapse (CNS), and no relapse (NOREL). During each iteration, a prediction is made to assign a category. Then a probability is calculated for having an accuracy performance for that iteration. A single iteration has a different random seed, which generates a different set of lambda coefficients for adjusting the lasso penalty. The best lambda across a grid of coefficients with the best accuracy classification is then selected based on accuracy. Adjusting the lambda score also adjusts the subset of genes used for the classification. For one best lambda there is one subset of significantly expressed genes and each gene has a different probability. For one best lambda there is one mean probability registered for that subset of genes.

281

4.1.1 Uncertainty estimation for selected genes from expression networks

282
283

The chart below shows how many iterations (dots) were executed for each sample group before selecting a subset of genes through regularization.

284
285
286

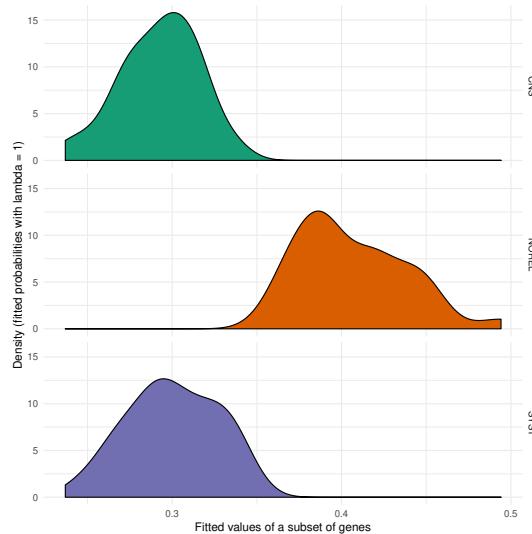
The probabilities are the fitted values of either a multinomial or binomial model at the best lambda (λ), shrinking parameter determined by tuning and cross-validation resampling. When predictions were made with lasso, the least squares were penalized. Lasso zero out coefficient estimates thus reducing the data.

If a subset has 50 genes, the reported probabilities are the mean of each gene individual probability to predict all individual cases

287 The fitted values are compared to the outcome to follow the proportion of variance "explained" by the
288 model and the proportion of variance "not explained".

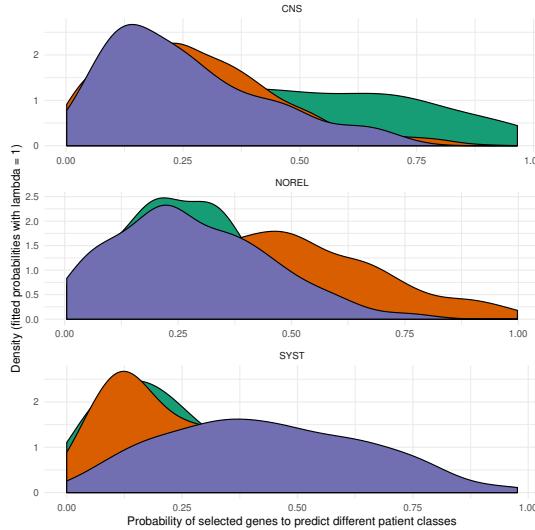
289
290 Peaks in density represents variance fitted at best λ between sample groups. Probabilities are compared
291 to the residuals of the data, the outcome is the fitted values. As long as the peaks differ between groups,
292 then the prediction is possible between samples. There is an overlay between CNS and SYST groups,
293 which indicates the presence of some bias in differentiating between them.

```
read.table("./data/logSummary.lambda.iterations30.multinomial.probabilities.txt",
           row.names = 1, header = T) %>%
  ggplot(aes(x = probabilityScore,
             fill = class)) +
  geom_density() +
  theme_minimal() +
  facet_grid(class ~ .) +
  theme(legend.position = "none") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "Fitted values of a subset of genes",
       y = "Density (fitted probabilities with lambda = 1)")
```



294
295 Density plot summarizing the distribution of how many times the CNS/NOREL/SYST sample classes were
296 correctly predicted against other classes using the list of genes that supposedly represent individuals with
297 each type of prognosis.

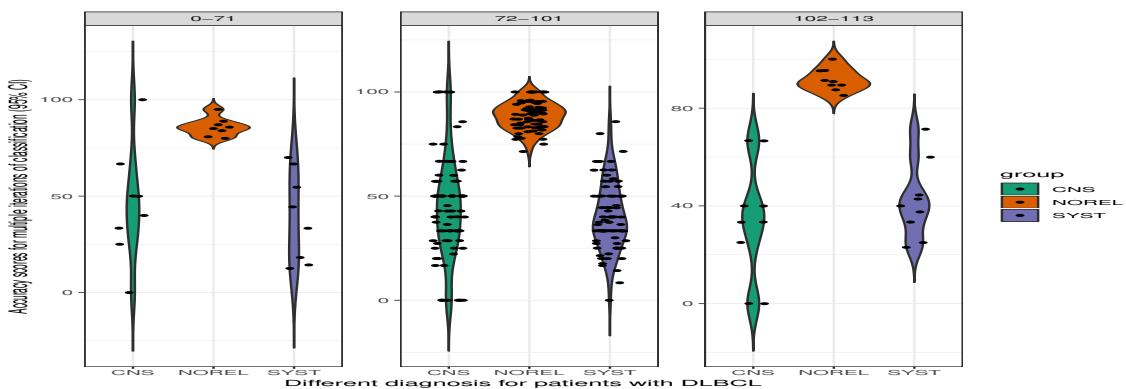
```
read.table("./data/logSummary.lambda.iterations30.multinomial.densities.txt",
           row.names = 1, header = T) %>%
  filter(classes == c("CNS", "SYST", "NOREL")) %>%
  ggplot(aes(x = probabilities,
             fill = classes)) +
  geom_density() +
  theme_minimal() +
  facet_wrap(~ genes,
             ncol = 1,
             scales = "free") +
  theme(legend.position = "none") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "Probability of selected genes to predict different patient classes",
       y = "Density (fitted probabilities with lambda = 1)")
```



298
299 Plot showing the accuracy of assigning an individual to its correct class (or diagnosis) based on lambda
300 calculation for lasso regularization. Each facet represents an accuracy for multiple iterations with a specific
301 number of genes.

```
df <- read.table("./data/logSummary.lambda.iterations30.multinomial.accuracies.txt",
  row.names = 1, header = T)
mir <- min(df$regNgenes)
mar <- max(df$regNgenes)
q1 <- floor((mir+mar)/2.5)
q2 <- floor((mir+mar)/1.75)
df$grouped <- cut(df$regNgenes, c(0, q1, q2, mar))
levels(df$grouped) <- c(paste0(0, "-", q1),
  paste0(q1+1, "-", q2),
  paste0(q2+1, "-", mar))

df %>%
  ggplot(aes(x = group,
  y = accuracy,
  fill = group)) +
  geom_violin(trim = FALSE) +
  geom_jitter(shape=16, position=position_jitter(0.2)) +
  scale_fill_brewer(palette = "Dark2") +
  theme_bw() +
  labs(x = "Different diagnosis for patients with DLBCL",
  y = "Accuracy scores for multiple iterations of classification (95% CI)") +
##  theme(legend.position = "top") +
  facet_wrap(~ grouped,
  ncol = 3,
  scales = "free")
```



302
303 **4.2 Selected machine and deep learning models**
304 Selection of learners was based on historical efficiency of such algorithms. Also, the training of classifiers
305 starts from simple logistic regression, naive bayes, nearest neighbors, going to more flexible models, such
306 as trees and deep neural nets that require more hyperparameter tuning. This strategy lets assess with

307 less bias the overfitting issues that may arise.

Table 1: Machine learning models

#	Classifiers trained	R package*	Parameters [†] tuned	Abbreviation
1	Naive bayes	naivebayes	laplace, usekernel, adjust	naive_bayes
2	Weighted k-Nearest Neighbors	kknn	kmax, distance, kernel	kknn
3	Penalized multinomial regression	nnet	decay	multinom
4	Random forest	randomForest	mtry	rf
5	Regularized random forest	RRF	mtry, coefReg, coefImp	RRF
6	Linear discriminant analysis (LDA)	MASS	dimen	lda2
7	Multilayer perceptron network by stochastic gradient descent	FCNN4R	size, l2reg, lambda, learn_rate, momentum, gamma, minibatchsz, repeats	mlpSGD
8	Flexible discriminant analysis (FDA)	mda	degree, nprune	fda
9	Bagged FDA	mda	degree, nprune	bagFDA
10	Bagged FDA using gCV pruning	earth	degree	bagFDAGCV
11	Penalized discriminant analysis	mda	lambda	pda
12	Partial least squares	pls	ncomp	kernelpls
13	Support vector machines (SVM) with linear kernel	kernlab	C	svmLinear
14	L2 regularized SVM (dual) with linear kernel	LiblineaR	cost, loss	svmLinear3
15	SVM with polynomial kernel	kernlab	degree, scale, C	svmPoly
16	SVM with radial basis function kernel	kernlab	sigma, C	svmRadialSigma
17	Neural network (NN)	nnet	size, decay	nnet
18	NN with feature extraction	nnet	size, decay	pcaNNet
19	Monotone multi-layer perceptron NN	monmlp	hidden1, n.ensemble	monmlp
20	Stacked autoencoder deep NN	deepnet	layer1, layer2, layer3, hid-den_dropout, visible_dropout	dnn
21	Stochastic gradient boosting	gbm	n.trees, interaction.depth, shrink-age, n.minobsinnode	gbm

* The version of each package is shared under section 4.10. Links are forwarded to the CRAN page (except those imported from [Tensorflow](#) and [H2O](#)) of each package for assessment of version, vignettes, advanced functionality, and description. [†]Parameters are crucial to optimize for accuracy (95% CI). Similar models have different parameters. Multi-layered neural networks are used for deep learning. In some instances, only the layer 1 is used. For such instances the classifier is considered a neural network.

308 4.3 Available and tuned hyperparameters for each model

309 4.4 Machine learning performance benchmarks

310 Please follow up on performance metrics for classification problem by reading [Sokolova 2009](#).

[↑ Link to metrics definitions](#)

- 311 • **Sensitivity**, is how many true cases are correctly classified to their expected class. Or **recall**, is the
312 fraction of events where we correctly declared i form all cases where the true state of the world
313 is i . $TP/(TP + FN)$
- 314 • **Specificity**, is how many wrong cases are correctly classified elsewhere. $TN/(TN + FP)$
- 315 • **Precision**, is the fraction of events where we correctly declared i out of all instances where the
316 algorithm declared i . $TP/(TP + FP)$
- 317 • **Accuracy**, is an overall measure that assesses the predictive model by comparing predicted classes
318 to observed expected classes. Scores can also be shown as the area under the receiver operator
319 curve (AUROC, 95% CI). $(TN + TP)/(TP + TN + FP + FN)$

320 4.4.1 Creating the baseline of models performance

321 Machine learning models were trained only without tuning for hyperparameter optimization. Metrics generated
322 show the raw performance of each model.

[↑ Precision and recall are best
for multi class learning](#)

323 For this type of nominal data, classification models (not regression) are used, see Section /refsub-
324 sec:models. The performance metrics for this type of models are an accuracy score and kappa, which
325 takes into account the possibility of the agreement occurring by chance (the kappa score however reflects
326 the adequate agreement). Standard error (**SE** in red) bars for the kappa significance per model repro-
327ducible across 10 cross-validation each repeated 5 times. Minimum and maximum accuracy thresholds
328 are held at 95% confidence intervals.

329 Load standard error and deviation equations.

[↑ Kappa is Cohen's
\(unweighted\) Kappa statistic
averaged across the resampling
results](#)

```
summary_SE <- function(data=NULL, measurevar, groupvars=NULL, na.rm=FALSE,
```

Table 2: Hyperparameters for each classifier.

Classifier trained	Hyperparameter	Scenario A*	Scenario B**	Scenario C***
Naive bayes	laplace usekernel adjust			
Weighted k-Nearest Neighbors	kmax distance kernel			
Penalized multinomial regression	decay			
Random forest	mtry			
Regularized random forest	mtry coefReg coefImp			
Linear discriminant analysis (LDA)	dimen			
Localized LDA	k			
Flexible discriminant analysis (FDA)	degree nprune			
Bagged FDA	degree nprune			
Bagged FDA using gCV pruning	degree			
Penalized discriminant analysis	lambda			
Partial least squares	ncomp			
Support vector machines (SVM) with linear kernel	C 0.1			
L2 regularized SVM (dual) with linear kernel	cost loss			
SVM with polynomial kernel	degree scale C			
SVM with radial basis function kernel	sigma C			
Neural network (NN)	size decay			
NN with feature extraction	size decay			
Monotone multi-layer perceptron NN	hidden1 n.ensemble			
Stacked autoencoder deep NN	layer1 layer2 layer3 hidden_dropout visible_dropout			
Boosted logistic regression	nIter			
Stochastic gradient boosting	n.trees interaction.depth shrinkage n.minobsinnode			
Multilayer perceptron network by stochastic gradient descent	size l2reg lambda learn_rate momentum gamma minibatchsz repeats			

*Comparing samples with individuals grouped by either having a central nervous system relapse (CNS), systemic relapse (SYST), no relapse (NOREL) or controls. **Comparing samples with individuals grouped by either having an activated B-cell (ABC) or germinal center B-cell (GCB) diagnosis. ***Comparing samples with individuals grouped by either having an tumor occurrence in lymph nodes (LN) in contrast to extra-nodal (EN) presence. The tuning parameters were iterated across a random selection grid for best tuning. Grid configurations are found [on Github](#).

```

conf.interval=.95, .drop=TRUE) {

length2 <- function (x, na.rm=FALSE) {
  if (na.rm) sum(!is.na(x))
  else length(x)
}
dataac <- ddply(data, groupvars, .drop=.drop,
  .fun= function(xx, col, na.rm) {
    c( N      = length2(xx[,col], na.rm=na.rm),
      mean   = mean     (xx[,col], na.rm=na.rm),
      sd     = sd       (xx[,col], na.rm=na.rm)
    )
  },
  measurevar,
  na.rm
)
dataac <- rename(dataac, c("mean"=measurevar))
# Calculate standard error of the mean
dataac$se <- dataac$sd / sqrt(dataac$N)
# Confidence interval multiplier for standard error
# Calculate t-statistic for confidence interval:
ciMult <- qt(conf.interval/2 + .5, dataac$N-1)
dataac$ci <- dataac$se * ciMult
return(dataac)
}

```

332 Metrics for classification performance without tuning for hyperparameter optimization. Quick comparison
 333 of statistical learning on the DLBCL data.

```
accuracy.pre <- read.table("./data/reports.437017/log.Accuracy.performance1.multiparameter.437259.4
```

[†] Accuracy, is the true prediction rate averaged over cross-validation iterations

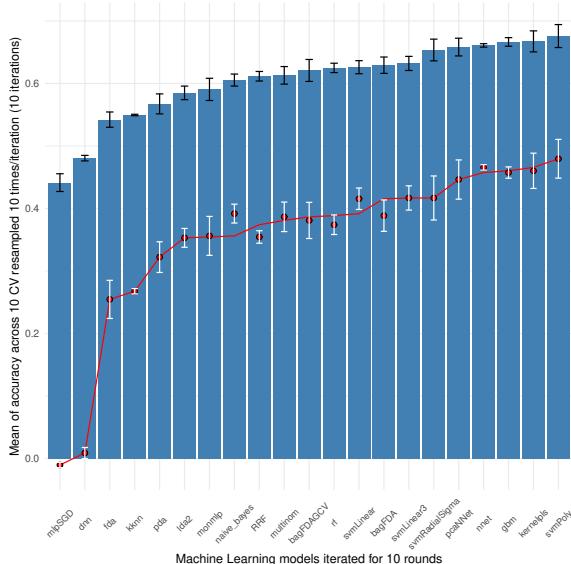
```

kappa <- read.table("./data/reports.437017/log.Kappa.performance1.multianalysis.seed5437259.437017.txt")

accuracy.se <- summary_SE(accuracy.pre, measurevar = "Mean", groupvars = "model")
kappa.se <- summary_SE(kappa, measurevar = "Mean", groupvars = "model") %>%
  arrange(Mean) %>%
  mutate(model = factor(model, unique(model)))

accuracy.se %>%
  arrange(Mean) %>%
  mutate(model = factor(model, unique(model))) %>%
  ggplot(aes(x = model,
    y = Mean)) +
  geom_bar(position="dodge",
    stat="identity",
    fill = "steelblue") +
  geom_errorbar(data = accuracy.se,
    aes(ymin=Mean-se,
      ymax=Mean+se),
    width=.3,
    position=position_dodge(.9)) +
  geom_line(data = kappa.se,
    aes(x = as.integer(model),
      y = Mean),
    color = "red") +
  geom_point(data = kappa.se,
    size=2, shape=21, fill="red") +
  geom_errorbar(data = kappa.se,
    aes(ymin=Mean-se,
      ymax=Mean+se),
    width=.25,
    position=position_dodge(.9),
    color = "white") +
  theme_minimal() +
  ylab("Mean of accuracy across 10 CV resampled 10 times/iteration (10 iterations)") +
  xlab("Machine Learning models iterated for 10 rounds") +
  theme(legend.position = "top") +
  guides(fill=guide_legend(title="Number of parameters per model")) +
  theme(axis.text.x = element_text(vjust = .5,
    angle = 45,
    size = 8))

```



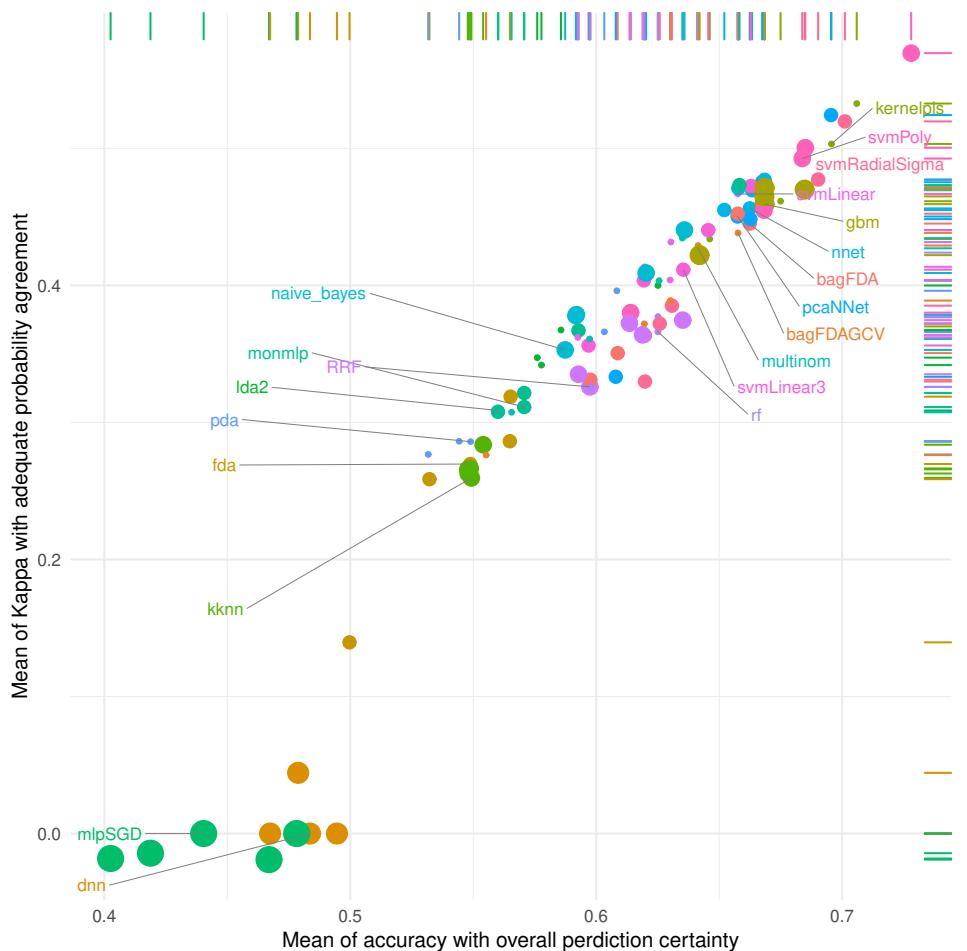
334

335 Kappa (vertical axis) and accuracy (horizontal axis) calculated from the performance tests of machine
 336 learning models. The higher kappa is the stronger agreement for a prediction and classification. Although,
 337 this chart shows little correlation between prediction accuracy and outcome of each model, its important

338 to highlight the number of parameters tune for each classifier. Because, based on a recent paper from
 339 Google in Nature, **Rajkomar 2018**, deep learning models with many layers and hyperparameters perform
 340 well enough as a regularized logistic regression model (Supplementary text). In the paper however, little
 341 was mentioned about this discrepancy in the published results. Lastly, it remains important to address the
 342 value of the parameters that are manipulated and the amount of time spent training the model based on
 343 the tuning process.

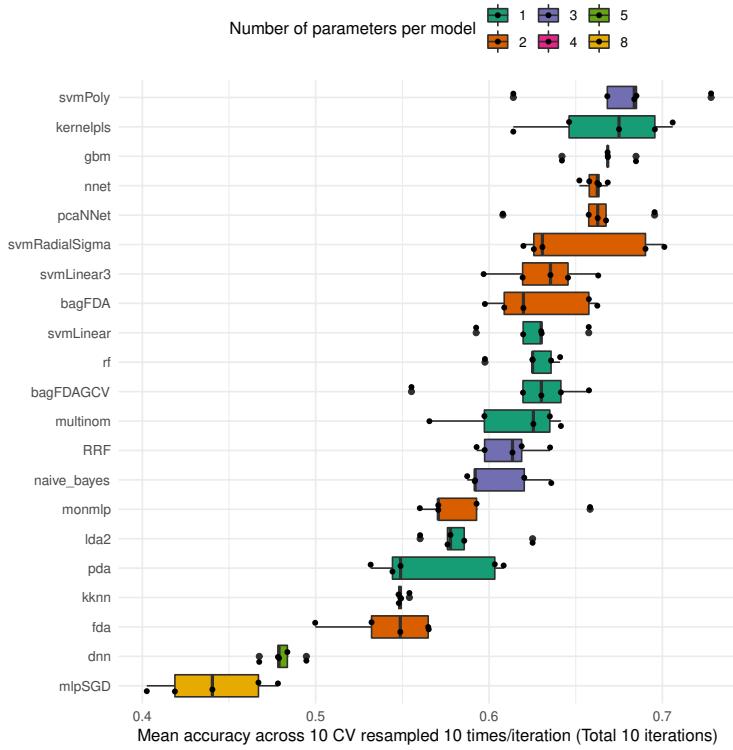
```
## get single instance label names
model.names <- data.frame(accuracy.pre, kappa) %>%
  select(model, iteration, Mean, Mean.1) %>%
  filter(iteration == 1)

data.frame(accuracy.pre, kappa) %>%
  ggplot(aes(x = Mean,
             y = Mean.1,
             color = model,
             label = model)) +
  geom_point(aes(size=parameters, fill = model), shape=21) +
  geom_rug(aes(color = as.character(model)),
            sides = "tr",
            show.legend = F) +
  geom_text_repel(data = subset(model.names, Mean >= .6),
                  nudge_x = .04,
                  nudge_y = -.03,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 3) +
  geom_text_repel(data = subset(model.names, Mean < .6),
                  nudge_x = -.1,
                  force = 8,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 3) +
  theme_minimal() +
  ylab("Mean of Kappa with adequate probability agreement") +
  xlab("Mean of accuracy with overall prediction certainty") +
  theme(legend.position = "none")
```



344
345 Mean accuracy (95% CI) registered for machine learning fitting for all classification groups. Iterations for
346 reproducibility were executed 10 times.

```
accuracy.pre %>%
  ggplot(aes(x = reorder(model, Mean),
             y = Mean,
             fill = as.character(parameters))) +
  geom_boxplot() +
  geom_jitter(shape=16, position=position_jitter(0.2), cex = 1.5) +
  coord_flip() +
  scale_fill_brewer(palette = "Dark2") +
  theme_minimal() +
  ylab("Mean accuracy across 10 CV resampled 10 times/iteration (Total 10 iterations)") +
  xlab("") +
  theme(legend.position = "top") +
  guides(fill=guide_legend(title="Number of parameters per model"))
```



347

348 4.4.2 Models performance with hyperparameter tuning

349
350
351

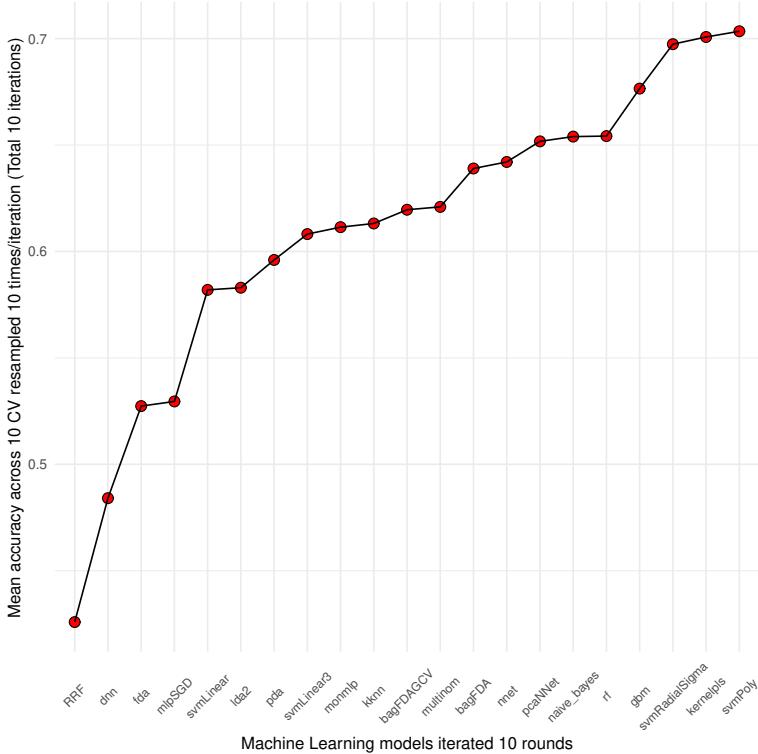
Compared to the baseline, the parameters available for each learner should increase its performance at predicting each expected outcome. Tuning these hyperparameters will leverage the results with increased accuracy.

352

353
354

Mean accuracy (95% CI) registered for machine learning fitting for all classification groups. Iterations for reproducibility were executed 10 times.

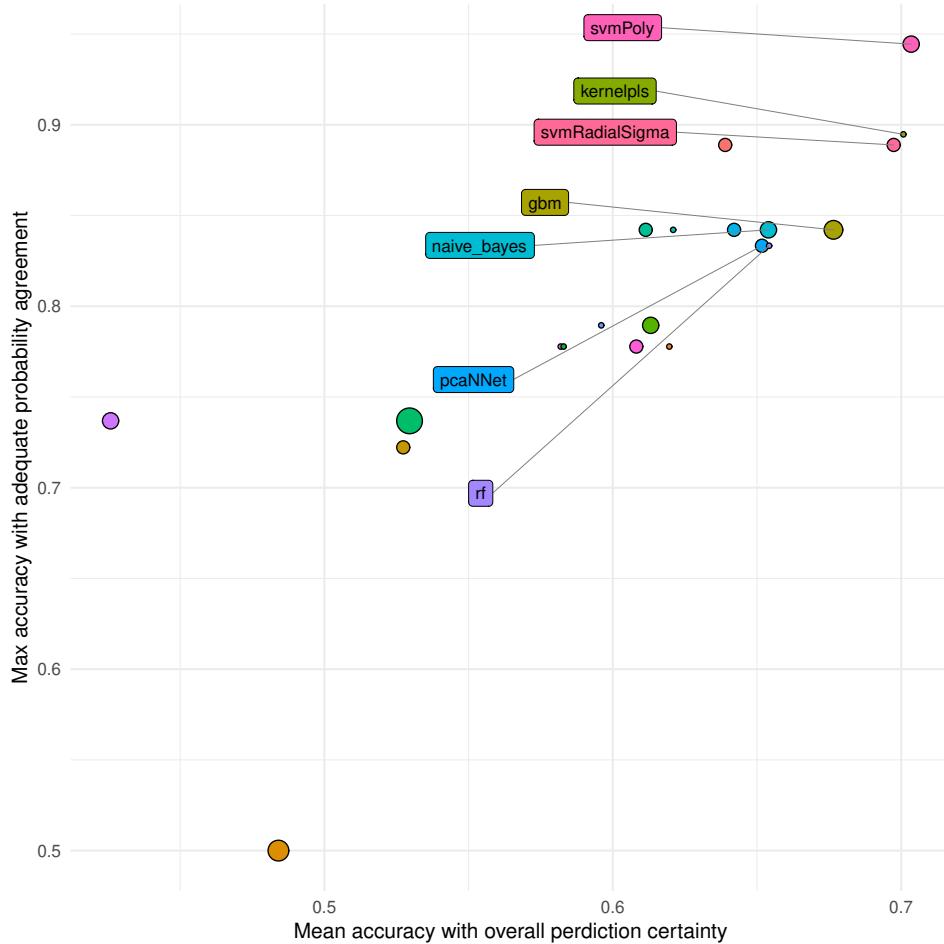
```
accuracy.pst <- read.table("./data/reports.437017/log.Accuracy.performance2.hyperTuning.seed5437259.437017")
accuracy.pst %>%
  arrange(Mean) %>%
  mutate(model = factor(model, unique(model))) %>%
  ggplot(aes(x = model,
             y = Mean)) +
  geom_point(size = 3, shape = 21, fill = "red") +
  geom_line(aes(x = as.integer(model),
                y = Mean)) +
  scale_fill_brewer(palette = "Dark2") +
  theme_minimal() +
  ylab("Mean accuracy across 10 CV resampled 10 times/iteration (Total 10 iterations)") +
  xlab("Machine Learning models iterated 10 rounds") +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 8))
```



355

356 Showing the mean accuracy of each model across its maximum predictive performance.

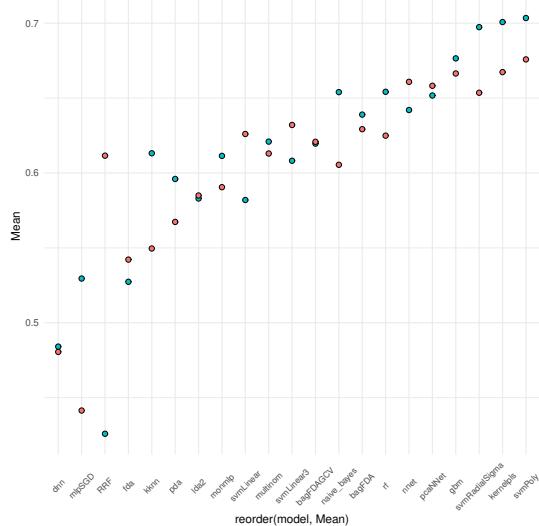
```
accuracy.pst %>%
  ggplot(aes(x = Mean,
             y = Max.,
             fill = model,
             label = as.factor(model))) +
  geom_point(aes(size=parameters), shape=21) +
  geom_label_repel(data = subset(accuracy.pst, Mean >= .65),
                   nudge_x = -.1,
                   force = 10,
                   segment.color = "grey50",
                   direction = "y",
                   segment.size = 0.1,
                   size = 3) +
  theme_minimal() +
  ylab("Max accuracy with adequate probability agreement") +
  xlab("Mean accuracy with overall prediction certainty") +
  theme(legend.position = "none")
```



357
358 Comparing the performance of each model during baseline testing versus optimization, both during the
359 training process.

```
pre.tune <- accuracy.se %>%
  select(model, Mean)
post.tune <- accuracy.pst %>%
  select(model, Mean)

full_join(pre.tune, post.tune, by = "model") %>%
  gather("condition", "Mean", 2:3) %>%
  arrange(Mean) %>%
  mutate(model = factor(model, post.tune$model)) %>%
  ggplot(aes(x = reorder(model, Mean),
              y = Mean,
              fill = condition)) +
  geom_point(shape = 21, size = 2) +
  theme_minimal() +
  theme(axis.text.x = element_text(vjust = .5,
                                    angle = 45,
                                    size = 8),
        legend.position = "none")
```



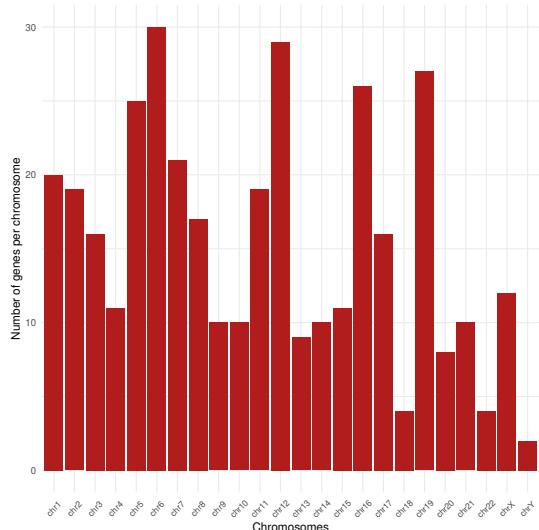
360

361 4.4.3 Genomic representation of selected genes

362 Distribution of all the genes that were selected after differential expression and network analysis.

```
read.table("./data/log.gene.names_allnetworks.txt", header = T) %>%
  mutate(order = gsub("chr", "", chromosome)) %>%
  mutate(order = gsub("X", "24", order)) %>%
  mutate(order = gsub("Y", "25", order)) %>%
  arrange(as.numeric(order)) %>%
  ggplot(aes(x = reorder(chromosome, as.numeric(order)))) +
  geom_histogram(stat = "count", fill = "firebrick") +
  theme_minimal() +
  labs(x = "Chromosomes",
       y = "Number of genes per chromosome") +
  theme(axis.text.x = element_text(vjust = .5,
                                    angle = 45,
                                    size = 8))
```

Warning: Ignoring unknown parameters: binwidth, bins, pad



363

364 Distribution of the genes with preferential transcriptional characteristics to predict individuals who are at
365 risk or not of CNS relapse after diagnosis and DLBCL treatment.

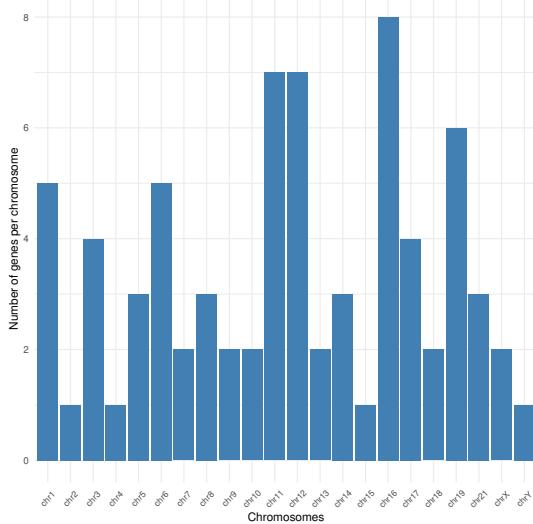
```
read.table("./data/log.gene.names_bestLambda.txt", header = T) %>%
```

```

  mutate(order = gsub("chr", "", chromosome)) %>%
  mutate(order = gsub("X", "24", order)) %>%
  mutate(order = gsub("Y", "25", order)) %>%
  arrange(as.numeric(order)) %>%
  ggplot(aes(x = reorder(chromosome, as.numeric(order)))) +
  geom_histogram(stat = "count", fill = "steelblue") +
  theme_minimal() +
  labs(x = "Chromosomes",
       y = "Number of genes per chromosome") +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 8))

```

Warning: Ignoring unknown parameters: binwidth, bins, pad



366

4.4.4 Prediction accuracy of each classifier at optimal parameters

367
368
369

Final report of the actual accuracy (95% CI) for each machine learning model from comparing predicted values and expected outcomes.

370
371
372

How long (seconds) a statistical learner requires to optimize the hyperparameters and gets the highest significant accuracy on expected data.

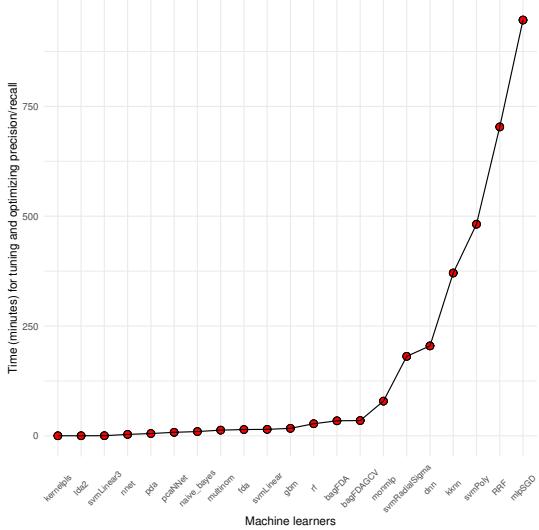
↑ Data are retrieved from Confusion Matrix

```

df <- read.table("./data/reports.437017/log.performance3.full.hyperTuning.seed5437259.437017.txt", header=TRUE)
df$group <- gsub("[0-9]", "", df$group)

df %>%
  arrange(durationMinutes) %>%
  mutate(model = factor(model, unique(model))) %>%
  ggplot(aes(x = model,
             y = durationMinutes)) +
  geom_point(stat = "identity",
             size=3, shape=21,
             fill = "red") +
  geom_line(aes(x = as.integer(model))) +
  theme_minimal() +
  xlab("Machine learners") +
  ylab("Time (minutes) for tuning and optimizing precision/recall") +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 8))

```



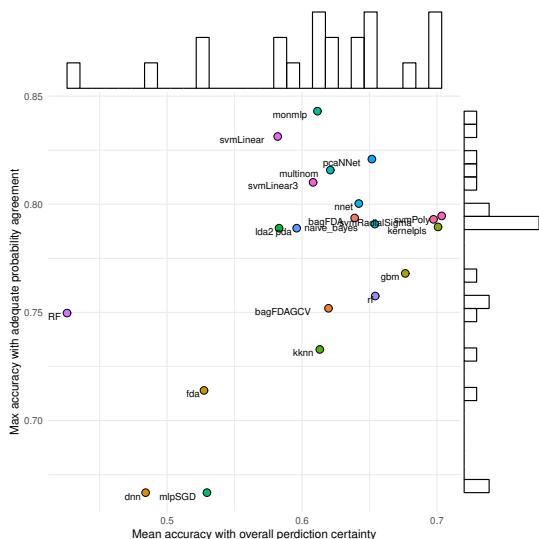
373
374 Showing the mean accuracy of each model in contrast to their specificity for predicting individuals classes.
375

```
specificity.mean <- df %>%
  select(model, Specificity) %>%
  summary_SE(measurevar = "Specificity", groupvars = "model") %>%
  select(model, Specificity)

accuracy.mean <- accuracy.pst %>%
  select(model, Mean)

p <- full_join(specificity.mean, accuracy.mean, by = "model") %>%
  ggplot(aes(x = Mean,
             y = Specificity,
             fill = as.factor(model))) +
  geom_point(shape=21, size = 3) +
  geom_text(aes(x = Mean,
                y = Specificity,
                label = model,
                hjust = 1.3,
                vjust = 1), size = 3) +
  theme_minimal() +
  ylab("Max accuracy with adequate probability agreement") +
  xlab("Mean accuracy with overall prediction certainty") +
  theme(legend.position = "none")

ggMarginal(p, type = "histogram", fill="transparent")
```



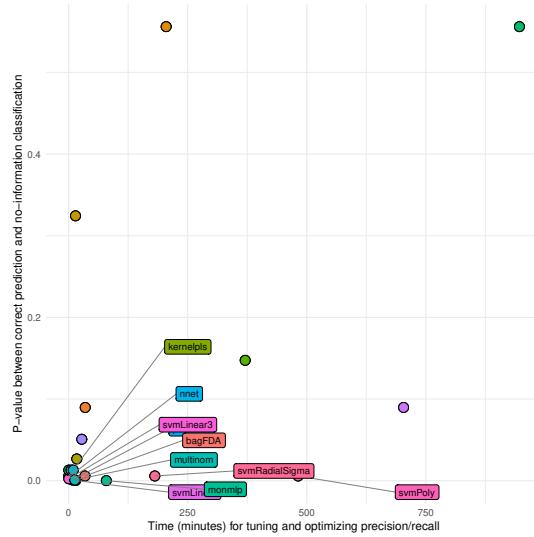
376

377 How is time training a model deliver on the significance of its accuracy? The p-value evaluates whether
378 the overall accuracy rate is greater than the rate of the largest class. Proportions between classes (if one
379 group of samples is larger than an other) is also considered in the hypothesis testing.

[↑ Link documentation Section 17.2](#)

```
unique.acc <- df %>%
  select(model, accuracyPval, durationMinutes) %>%
  unique()

df %>%
  ggplot(aes(y = accuracyPval,
             x = durationMinutes,
             fill = as.character(model),
             label = as.factor(model))) +
  geom_point(size = 4,
             shape = 21) +
  geom_label_repel(data = subset(unique.acc, accuracyPval <= .01),
                  nudge_y = -1,
                  nudge_x = 250,
                  force = 5,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 3) +
  theme_minimal() +
  xlab("Time (minutes) for tuning and optimizing precision/recall") +
  ylab("P-value between correct prediction and no-information classification") +
  theme(legend.position = "none")
```



380

381 Precision versus recall across all sample groups for a multi-class classification.

[↑ True/False Positives/Negatives](#)

```
prec.rec <- df %>%
```

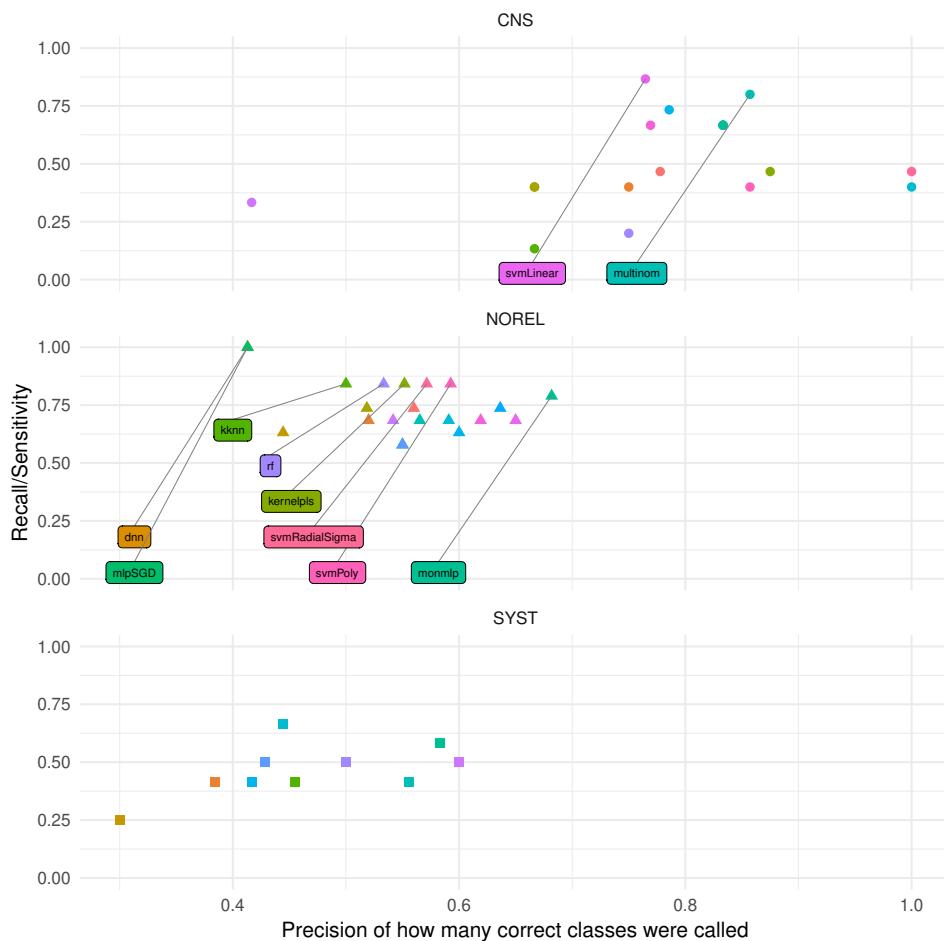
```

select(group, model, Precision, Recall)

df %>%
  ggplot(aes(x = Precision,
             y = Recall,
             fill = model,
             label = as.factor(model),
             na.rm = T)) +
  geom_point(aes(shape = group, color = model),
             size = 2) +
  geom_label_repel(data = subset(prec.rec, Recall >= .75),
                   nudge_y = -1,
                   nudge_x = -.1,
                   force = 1,
                   segment.color = "grey50",
                   direction = "y",
                   segment.size = 0.1,
                   size = 2) +
  facet_wrap(~ group, ncol = 1) +
  theme_minimal() +
  xlab("Precision of how many correct classes were called") +
  ylab("Recall/Sensitivity") +
  theme(legend.position = "none")

```

Warning: Removed 4 rows containing missing values (geom_point).



382

383 Specificity and sensitivity across all sample groups.

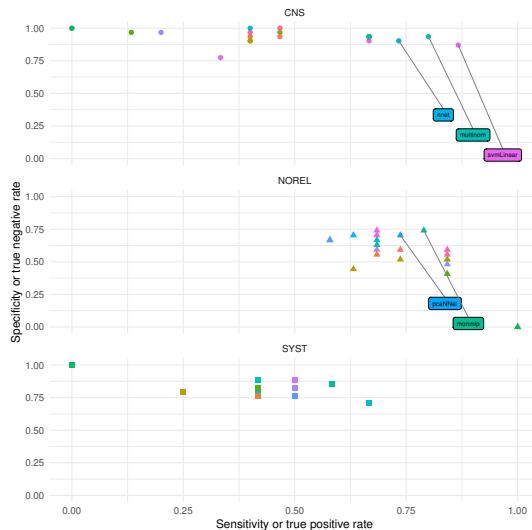
```
spec.sens <- df %>%
```

```

select(group, model, Sensitivity, Specificity)

df %>%
  ggplot(aes(x = Sensitivity,
             y = Specificity,
             fill = model,
             label = as.factor(model))) +
  geom_point(aes(shape = group,
                 color = model),
              size = 2) +
  geom_label_repel(data = subset(spec.sens, Specificity >= .7 & Sensitivity >= .7),
                   nudge_y = -1,
                   nudge_x = .1,
                   force = 1,
                   segment.color = "grey50",
                   direction = "y",
                   segment.size = 0.1,
                   size = 2) +
  facet_wrap(~ group, ncol = 1) +
  theme_minimal() +
  xlab("Sensitivity or true positive rate") +
  ylab("Specificity or true negative rate") +
  theme(legend.position = "none")

```



384

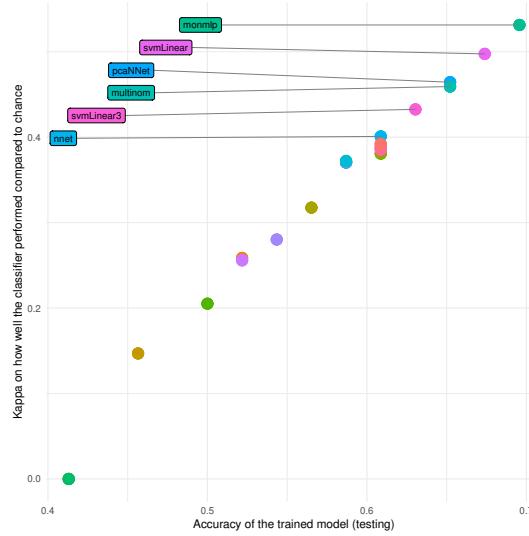
385 Accuracy and Kappa of each of the optimized model across all sample groups after grid tuning search. [` Optimized model](#)

```
pak <- df %>%
```

```

  select(model, accuracy, kappa) %>%
  unique
df %>%
  ggplot(aes(x = accuracy,
             y = kappa,
             fill = model,
             label = as.factor(model))) +
  geom_point(aes(size = 4,
                  color = model)) +
  geom_label_repel(data = subset(pak, accuracy >= .6 & kappa >= .4),
                    nudge_x = -.2,
                    force = 1,
                    segment.color = "grey50",
                    direction = "y",
                    segment.size = 0.1,
                    size = 3) +
  theme_minimal() +
  xlab("Accuracy of the trained model (testing)") +
  ylab("Kappa on how well the classifier performed compared to chance") +
  theme(legend.position = "none")

```



386
387 Accuracy versus the p-value of each classification. The p-value is a hypothesis test between predicting
388 expected samples and the probability that the classification is biased by disproportionate class sizes (one
389 group of samples is larger than an other).

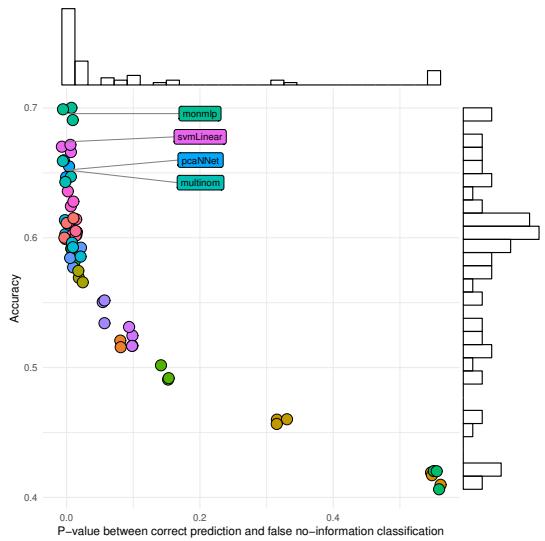
```
paap <- df %>%
```

```

  select(model, accuracy, accuracyPval) %>%
  unique
p <- df %>%
  ggplot(aes(x = accuracyPval,
             y = accuracy,
             label = as.factor(model),
             fill = model)) +
  geom_point(aes(size = 3,
                 fill = model),
              shape = 21,
              position=position_jitter(width=.01,height=.01)) +
  geom_label_repel(data = subset(paap, accuracy >= .65 & accuracyPval <= .01),
                   nudge_x = .2,
                   force = 1,
                   segment.color = "grey50",
                   direction = "y",
                   segment.size = 0.1,
                   size = 3) +
##    scale_x_reverse(limits = rev(levels(df$accuracyPval))) +
theme_minimal() +
ylab("Accuracy") +
xlab("P-value between correct prediction and false no-information classification") +
theme(legend.position = "none")

ggMarginal(p, type = "histogram", fill="transparent")

```



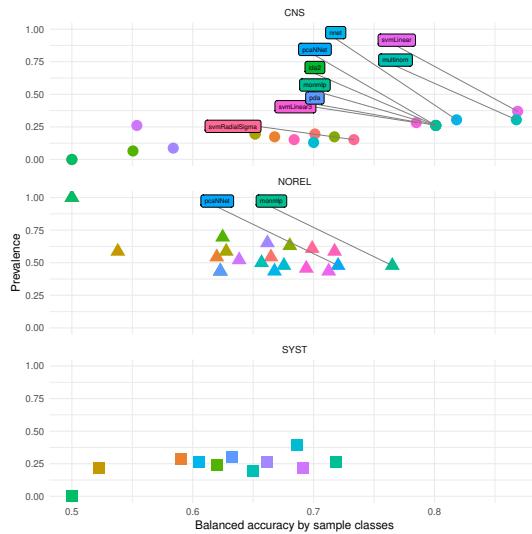
390
391 Prevalence of cases for each classifier. Were the classes perfectly balanced? A positive predictive score
392 is similar to precision while accounting for disproportionality of the classes.

pbd <- df %>%

```

  select(group, model, Balanced.Accuracy, Detection.Prevalence)
df %>%
  ggplot(aes(x = Balanced.Accuracy,
             y = Detection.Prevalence,
             label = as.factor(model),
             fill = model)) +
  geom_point(aes(size = 2,
                 color = model,
                 shape = as.factor(group))) +
  geom_label_repel(data = subset(pbd, Balanced.Accuracy >= .72),
                   nudge_x = -.1,
                   nudge_y = .5,
                   force = 15,
                   segment.color = "grey50",
                   direction = "y",
                   segment.size = 0.1,
                   size = 2) +
  facet_wrap(~ group, ncol = 1) +
  theme_minimal() +
  xlab("Balanced accuracy by sample classes") +
  ylab("Prevalence") +
  theme(legend.position = "none")

```



393

4.4.5 Probability to classify samples with the best model

Using the best model, support vector machines with a polynomial kernel, gets the cleanest predictions of individuals with CNS and systemic relapse.

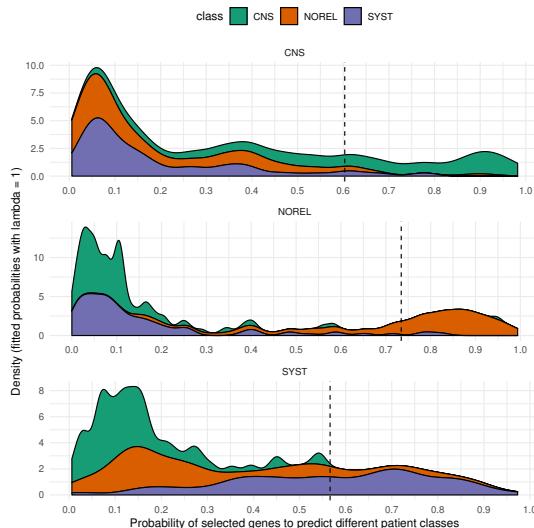
```
dfp <- read.table("./data/reports.437017/log.performance6.class_probabilities.seed5437259.437017.txt",
```

```

dfm = NULL
classes.s <- c("CNS", "NOREL", "SYST")
for (cp in 1:3) {
  dfmx <- dfp %>%
    select(one_of("original", classes.s[[cp]])) %>%
    filter(original == classes.s[[cp]])
  dfm <- rbind(dfm, data.frame(original = classes.s[[cp]],
                                grp.mean = mean(dfmx[, 2])))
}

dfp %>%
  gather("class", "prob", 3:5) %>%
  ggplot(aes(x = prob,
              fill = class)) +
  geom_density(alpha = 0.3) +
  geom_density(position = "stack") +
##  geom_area(aes(fill = class), stat = "bin", alpha=0.6) +
  geom_vline(data = dfm, mapping = aes(xintercept=grp.mean), linetype = "dashed") +
  scale_x_continuous(breaks = pretty(dfp$CNS, n = 10)) +
  theme_minimal() +
  facet_wrap(~ original,
             ncol = 1,
             scales = "free") +
  theme(legend.position = "top") +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "Probability of selected genes to predict different patient classes",
       y = "Density (fitted probabilities with lambda = 1)")

```



397

398 Similar to the above but non stacked densities.

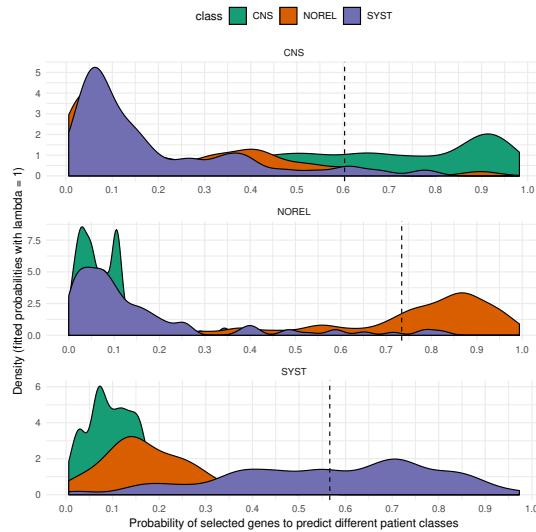
[†] similar & optional representation

```
dfp %>%
```

```

gather("class", "prob", 3:5) %>%
ggplot(aes(x = prob,
           fill = class)) +
geom_density() +
geom_vline(data = dfm, mapping = aes(xintercept=grp.mean), linetype = "dashed") +
scale_x_continuous(breaks = pretty(dfpm$CNS, n = 10)) +
theme_minimal() +
facet_wrap(~ original,
           ncol = 1,
           scales = "free") +
theme(legend.position = "top") +
scale_fill_brewer(palette = "Dark2") +
labs(x = "Probability of selected genes to predict different patient classes",
     y = "Density (fitted probabilities with lambda = 1)")

```



399

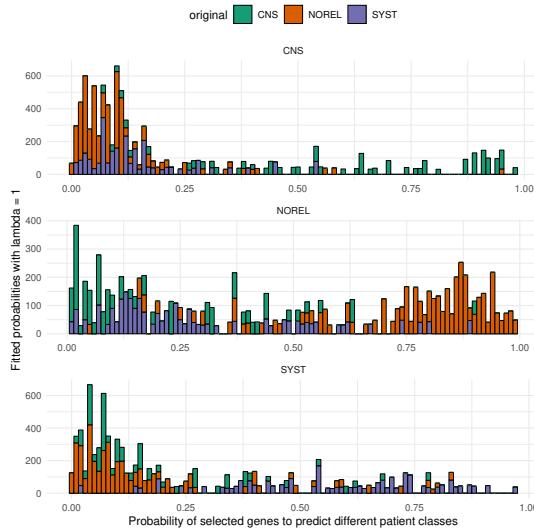
400 Bars showing concentration of good predictions by individual for each outcome class.

[↑ similar & optional representation](#)

```

dfp %>%
gather("class", "prob", 3:5) %>%
ggplot(aes(x = prob,
           fill = original)) +
geom_histogram(binwidth = 0.01, col = "black", size = .1) +
theme_minimal() +
facet_wrap(~ class,
           ncol = 1,
           scales = "free") +
theme(legend.position = "top") +
scale_fill_brewer(palette = "Dark2") +
labs(x = "Probability of selected genes to predict different patient classes",
     y = "Fitted probabilities with lambda = 1")

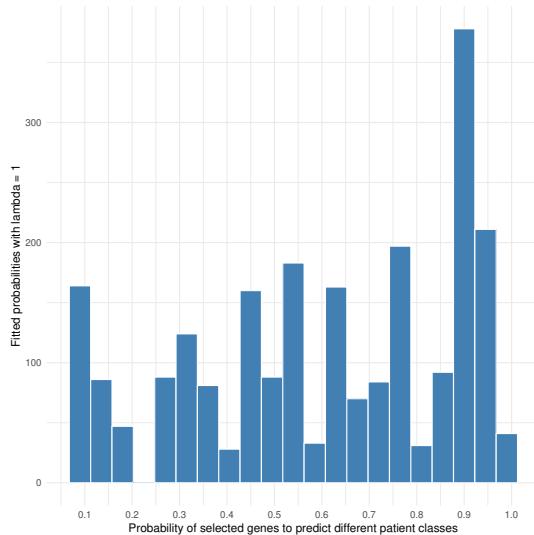
```



401
402

Distribution of probabilities for individuals identified with risk of central nervous system relapse.

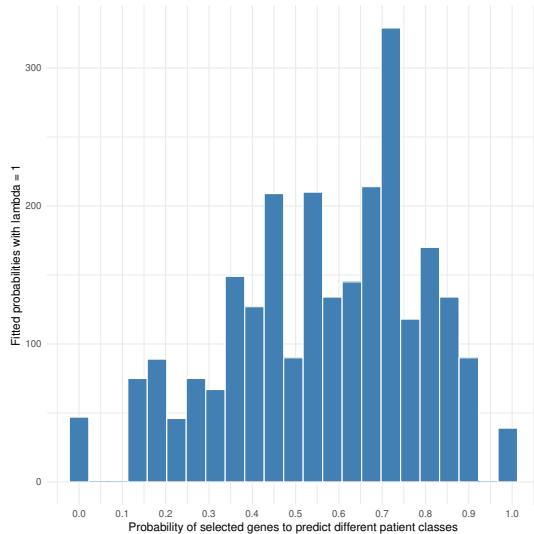
```
dfp %>%
  select(original, CNS) %>%
  filter(original == "CNS") %>%
  ggplot(aes(x=CNS)) +
  geom_histogram(binwidth = 0.045, color = "white", fill = "steelblue") +
  theme_minimal() +
  scale_x_continuous(breaks = pretty(dfps$CNS, n = 10)) +
  labs(x = "Probability of selected genes to predict different patient classes",
       y = "Fitted probabilities with lambda = 1")
```



403
404

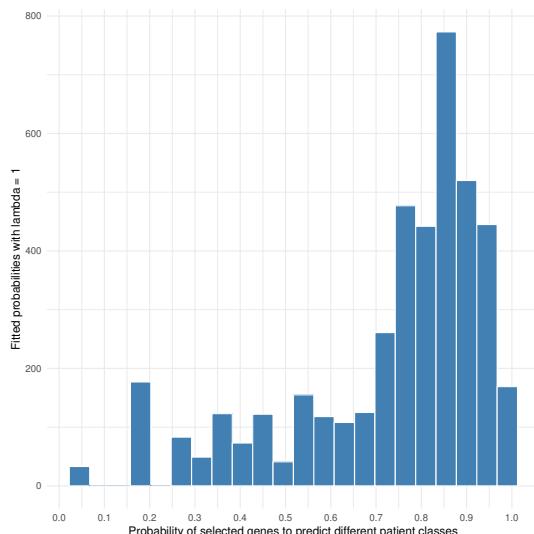
Distribution of probabilities for individuals identified with risk of a systemic relapse.

```
dfp %>%
  select(original, SYST) %>%
  filter(original == "SYST") %>%
  ggplot(aes(x=SYST)) +
  geom_histogram(binwidth = 0.045, color = "white", fill = "steelblue") +
  theme_minimal() +
  scale_x_continuous(breaks = pretty(dfps$SYST, n = 10)) +
  labs(x = "Probability of selected genes to predict different patient classes",
       y = "Fitted probabilities with lambda = 1")
```



405
406 Distribution of probabilities for individuals identified with a no risk of relapse after treatment, at the time of
407 diagnosis.

```
dfp %>%
  select(original, NOREL) %>%
  filter(original == "NOREL") %>%
  ggplot(aes(x=NOREL)) +
  geom_histogram(binwidth = 0.045, color = "white", fill = "steelblue") +
  theme_minimal() +
  scale_x_continuous(breaks = pretty(dfp$NOREL, n = 10)) +
  labs(x = "Probability of selected genes to predict different patient classes",
       y = "Fitted probabilities with lambda = 1")
```



408
409 **4.5 Importance scope of gene contribution**
410 Importance scores of each gene is calculated based on that variable significance to the model perfor-
411 mance. The score is thus model-based therefore taking into account the estimated correlation between
412 predictors. For each class, a gene is attributed an importance. All counts are scaled to 100. A trapezoid
413 rule is used to approximate an integral with linear guidelines. This helps estimate an area under the ROC
414 curve. The area is used to measure the variable importance.

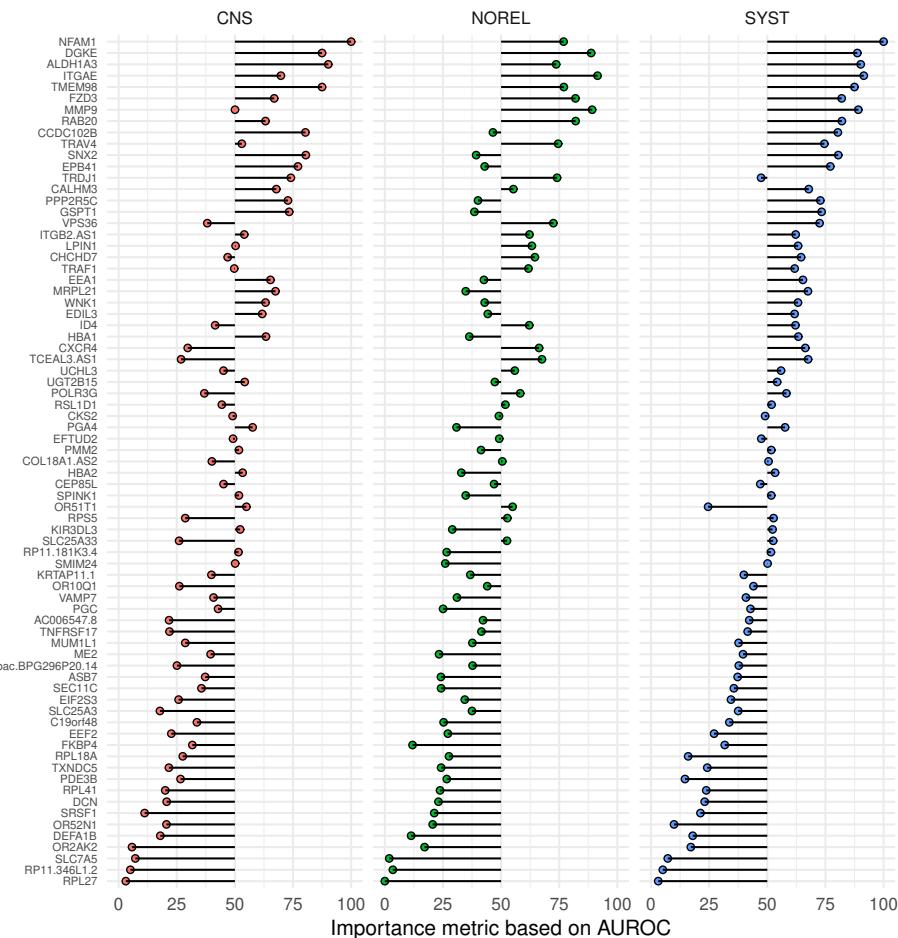
⁴Multiclass pairwise
decomposition AUROC

```
imp.model <- read.table("./data/reports.437017/log.performance5.importance.seed5437259.437017.txt", head=
```

```

imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  filter(model == "svmPoly") %>%
  gather("groups", "importance", 3:5) %>%
  ggplot(aes(x = reorder(genes, importance),
              y = importance,
              fill = as.character(groups))) +
  geom_point(shape=21) +
  geom_segment(aes(y = 50,
                   x = genes,
                   yend = importance,
                   xend = genes),
               color = "black") +
  facet_wrap(~ groups, ncol = 3) +
  theme_minimal() +
  coord_flip() +
  ylab("Importance metric based on AUROC") +
  xlab("") +
  theme(axis.text.y = element_text(size = 6)) +
  theme(legend.position = "none")

```



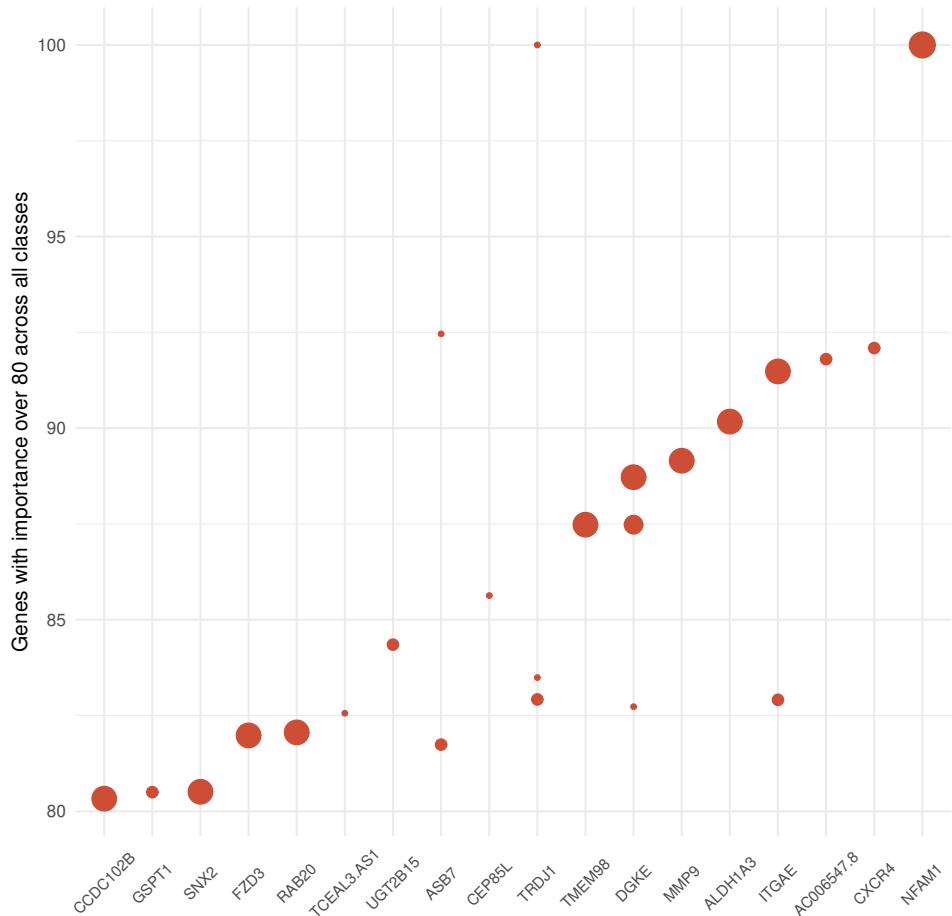
415
416 Genes ranked by importance across all classes. The bigger the bubble, the more models were to rely on
417 that one gene.

```
imp.model %>%
```

```

  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  gather("groups", "importance", 3:5) %>%
  filter(importance >= 80) %>%
  ggplot(aes(x = reorder(genes, importance),
             y = importance,
             color = groups)) +
  geom_count(col="tomato3", show.legend=F) +
  theme_minimal() +
  ylab("Genes with importance over 80 across all classes") +
  xlab("") +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 8)) +
  theme(legend.position = "top")

```



418
 419 Comparing the importance of genes across all model trained and optimized, annotated by genes, only
 420 for the SVM model with a polynomial kernel. Model comparison between individuals diagnosed with CNS
 421 relapse or no relapse.

```
model.names <- imp.model %>%
```

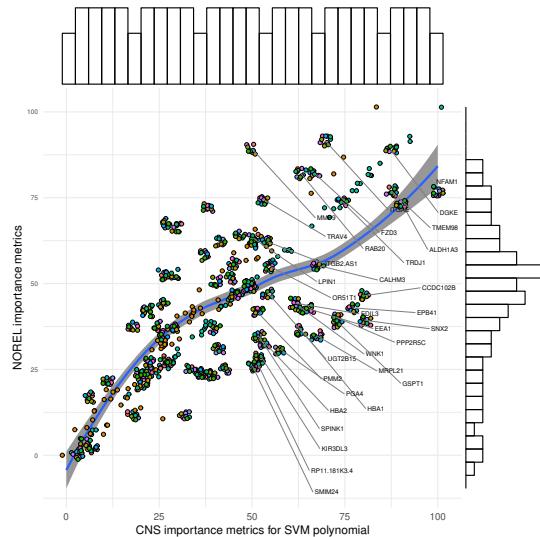
```

  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  select(genes, CNS, NOREL, model) %>%
  filter(model == "svmPoly")

p <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  ggplot(aes(x = CNS,
             y = NOREL,
             label = genes)) +
  geom_smooth(method = 'loess', alpha=1) +
  geom_point(aes(fill = model), shape=21,
             position=position_jitter(width=1.5, height=1.5)) +
  theme_minimal() +
  geom_text_repel(data = subset(model.names, CNS >= 50),
                  nudge_x = 20,
                  nudge_y = -20,
                  force = 5,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 2) +
  xlab("CNS importance metrics for SVM polynomial") +
  ylab("NOREL importance metrics") +
  theme(axis.text.y = element_text(size = 6)) +
  theme(legend.position = "none")

ggMarginal(p, type = "histogram", fill="transparent")

```



422
 423 Comparing the importance of genes across all model trained and optimized, annotated by genes, only
 424 for the SVM model with a polynomial kernel. Model comparison between individuals diagnosed with CNS
 425 relapse or systemic.

```
model.names <- imp.model %>%
```

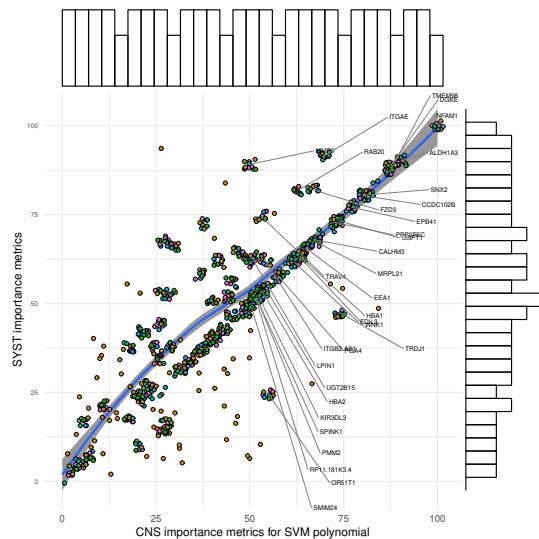
```

  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  select(genes, CNS, SYST, model) %>%
  filter(model == "svmPoly")

p <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  ggplot(aes(x = CNS,
             y = SYST,
             label = genes)) +
  geom_smooth(method = 'loess', alpha=1) +
  geom_point(aes(fill = model), shape=21,
             position=position_jitter(width=1.5, height=1.5)) +
  theme_minimal() +
  geom_text_repel(data = subset(model.names, CNS >= 50),
                  nudge_x = 20,
                  nudge_y = -20,
                  force = 5,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 2) +
  xlab("CNS importance metrics for SVM polynomial") +
  ylab("SYST importance metrics") +
  theme(axis.text.y = element_text(size = 6)) +
  theme(legend.position = "none")

ggMarginal(p, type = "histogram", fill="transparent")

```



426
427 Comparing the importance of genes across all model trained and optimized, annotated by genes, only for
428 the SVM model with a polynomial kernel. Model comparison between individuals diagnosed with systemic
429 or no relapse.

```
model.names <- imp.model %>%
```

```


mutate(genes = gsub(".TC.*$", "", genes)) %>%
  select(genes, SYST, NOREL, model) %>%
  filter(model == "svmPoly")

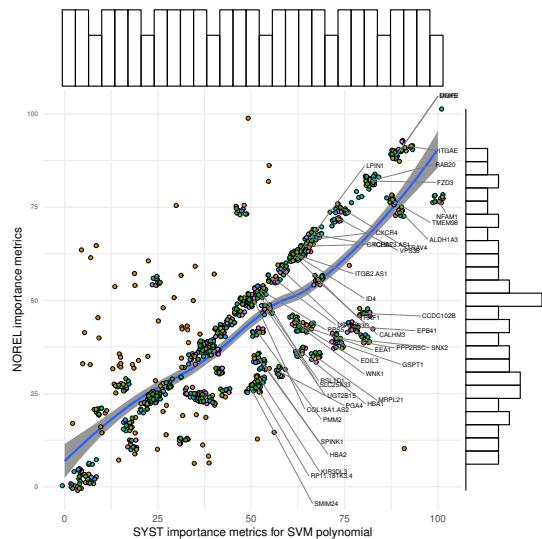


p <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  ggplot(aes(x = SYST,
             y = NOREL,
             label = genes)) +
  geom_smooth(method = 'loess', alpha=1) +
  geom_point(aes(fill = model), shape=21,
             position=position_jitter(width=1.5, height=1.5)) +
  theme_minimal() +
  geom_text_repel(data = subset(model.names, SYST >= 50),
                  nudge_x = 20,
                  nudge_y = -20,
                  force = 5,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 2) +
  xlab("SYST importance metrics for SVM polynomial") +
  ylab("NOREL importance metrics") +
  theme(axis.text.y = element_text(size = 6)) +
  theme(legend.position = "none")



ggMarginal(p, type = "histogram", fill="transparent")


```



430

4.6 Accuracy measured across pipelines

432
433
434
435

Different strategies for gene filtering can be used, either at different thresholds for some methods and implementing or not a method can produce a different accuracy. The performance was measured for SVM model with a polynomial kernel. Metrics were registered manually after the execution of the whole pipeline (from gene expression, to networks, and finally classification).

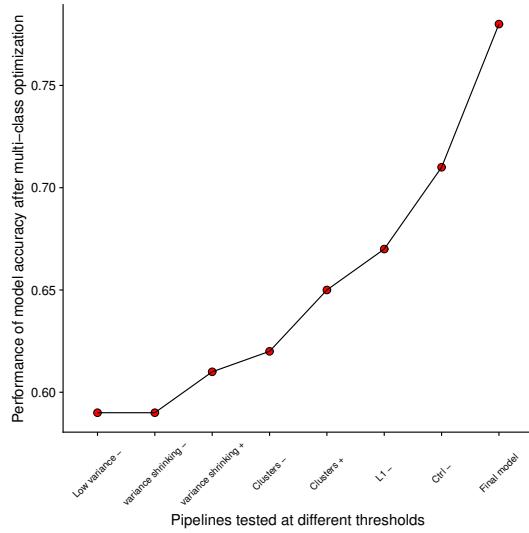
↳ Final model is w/o variance shrinking, w/o ctrl, w nets, w L1

```
pipelines <- c("variance shrinking +",
```

```

    "Low variance -",
    "variance shrinking -",
    "Clusters +",
    "Clusters -",
    "L1 -",
    "Ctrl -",
    "Final model")
perf.estimated <- c(.61, .59, .59, .65, .62, .67, .71, .78)
dfpe <- data.frame(pipelines = as.factor(pipelines), accuracy = perf.estimated)
dfpe %>%
  arrange(accuracy) %>%
  mutate(pipelines = factor(pipelines, pipelines)) %>%
  ggplot(aes(x = reorder(pipelines, accuracy),
              y = accuracy)) +
  geom_point(stat = "identity",
             size=3, shape=21,
             fill = "red") +
  geom_line(aes(x = as.integer(pipelines))) +
  xlab("Pipelines tested at different thresholds") +
  ylab("Performance of model accuracy after multi-class optimization") +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 9))

```



436

4.7 Expression of important genes summarized by RMA scores

The expression is based on RMA normalized likelihoods for 230 individuals. Genes with importance score above 90.

```

rma.genes <- read.table("./data/expressions.epoch50", header = TRUE)
sel.genes <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  gather("groups", "importance", 3:5) %>%
  filter(importance >= 90) %>%
  select(genes) %>%
  unique

sel.genes

      genes
1      CXCR4
2      NFAM1
3 AC006547.8
4      ALDH1A3
29     ITGAE
31     TRDJ1
50     ASB7

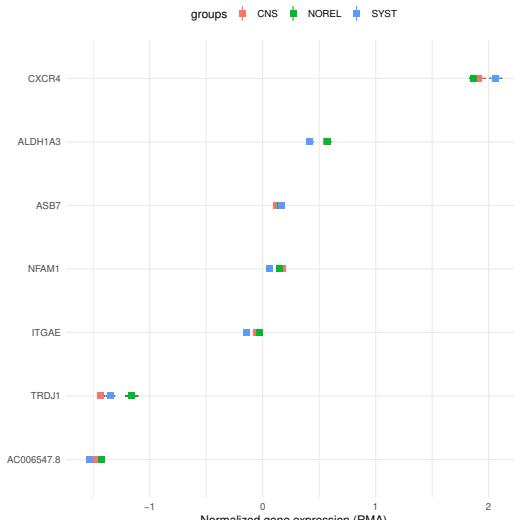
```

440 Create a forest plot of all important genes

```
gene.by.forest <- function(datx = rma.genes, y = meta.selected$Groups) {  
  dat <- data.frame(y, t(datx))  
  selx <- NULL  
  for(i in 1:nrow(sel.genes) ) {  
    sels <- dat %>%  
    select(contains(sel.genes$genes[i])) %>%  
    mutate(gene = sel.genes$genes[i]) %>%  
    mutate(groups = y)  
    colnames(sels) <- c("expression", "gene", "groups")  
  
    selx <- rbind(selx, sels)  
  }  
  sel.se <- summary_SE(selx, measurevar = "expression", groupvars = c("gene", "groups"))  
  
  sel.se %>%  
    ggplot(aes(x = reorder(gene, expression),  
                 y = expression,  
                 ymin = c(expression-se),  
                 ymax=c(expression+se)) +  
    xlab("") +  
    ylab("Normalized gene expression (RMA)") +  
    geom_errorbar(aes(ymin=expression-se,  
                         ymax=expression+se),  
                  width=0,  
                  position=position_dodge(.9)) +  
    geom_pointrange(aes(col=groups),  
                      size = 0.5, shape = 15) +  
    coord_flip() +  
    theme_minimal() +  
    theme(legend.position = "top")  
}  
}
```

441 Comparing important genes by patient relapse prognosis.

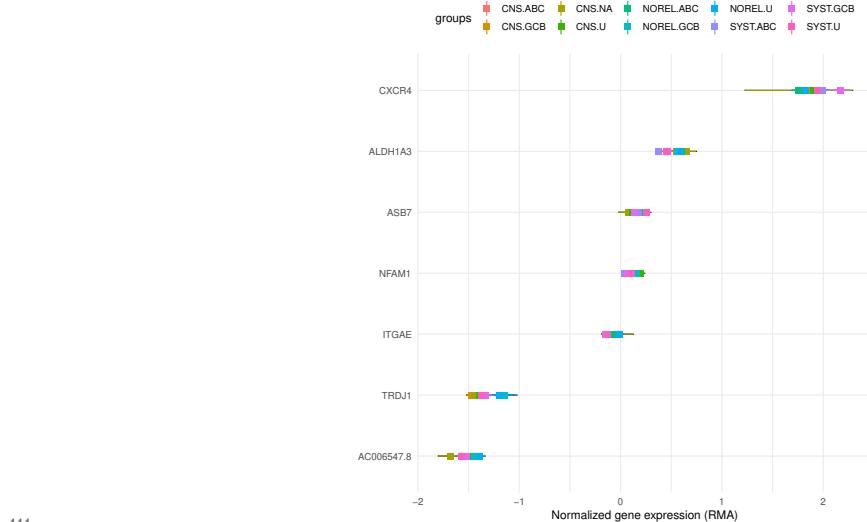
```
gene.by.forest(datx = rma.genes, y = meta.selected$Groups)
```



442

443 Comparing important genes by patient cell-of-origin prognosis.

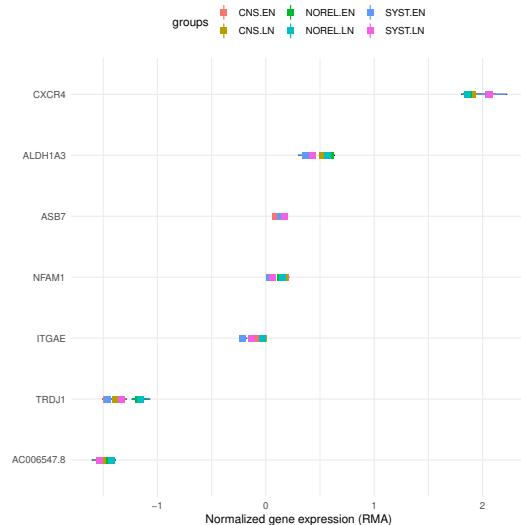
```
gene.by.forest(datx = rma.genes, y = meta.selected$Contrast1)
```



444

445 Comparing important genes by patient nodal involvement prognosis.

```
gene.by.forest(datx = rma.genes, y = meta.selected$Contrast2)
```



446

447 Define function that combines samples, RMA-normalized log2 fold changes, and prognosis classes. Gen-
448 erates plots.

```
gene.by.class <- function(data = rma.genes, y = meta.selected$Groups, geneSel = 1, implev = 90) {
```

```

y.samples <- y
dat <- data.frame(y.samples, t(data))

## get top genes
sel.genes <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  gather("groups", "importance", 3:5) %>%
  filter(importance >= implev) %>%
  select(genes) %>%
  unique

## get expression of each gene for each class
sel.box <- dat %>%
  select(contains(sel.genes$genes[geneSel])) %>%
  mutate(groups = y.samples)
colnames(sel.box) <- c("gene", "groups")

sel.box %>%
  arrange(gene) %>%
  ggplot(aes(x = reorder(groups, gene),
             y = gene,
             color = groups)) +
  geom_boxplot(outlier.colour = NA, lwd = .4) +
  geom_jitter(aes(color = factor(groups)),
              shape=16,
              position=position_jitterdodge(dodge.width=.8),
              cex = 2) +
  scale_color_brewer(palette="Paired") +
  theme_minimal() +
  theme(legend.position = "none") +
  xlab("") + ylab(paste0(sel.genes$genes[geneSel])) +
  theme(axis.text.x = element_text(vjust = .5,
                                    angle = 45,
                                    size = 10))
}

}

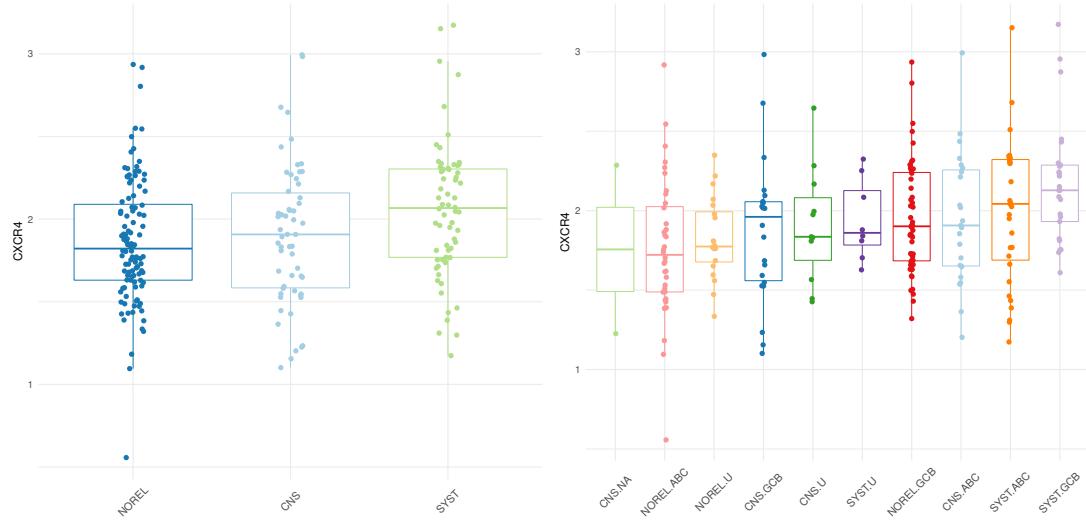
```

449 We choose the genes with the top important score, above than 90. CXRC4

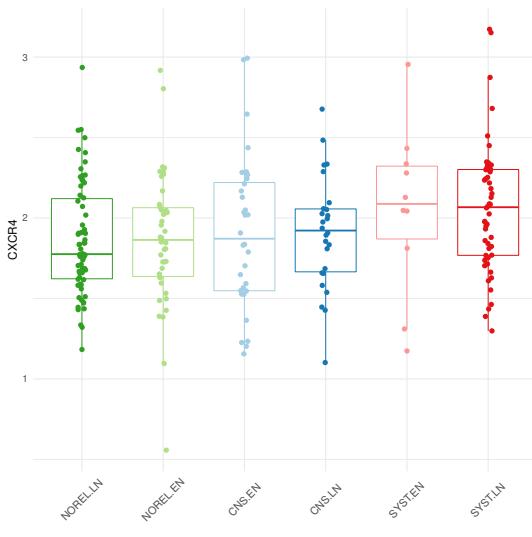
```

gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 1, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 1, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 1, implev = 90)

```



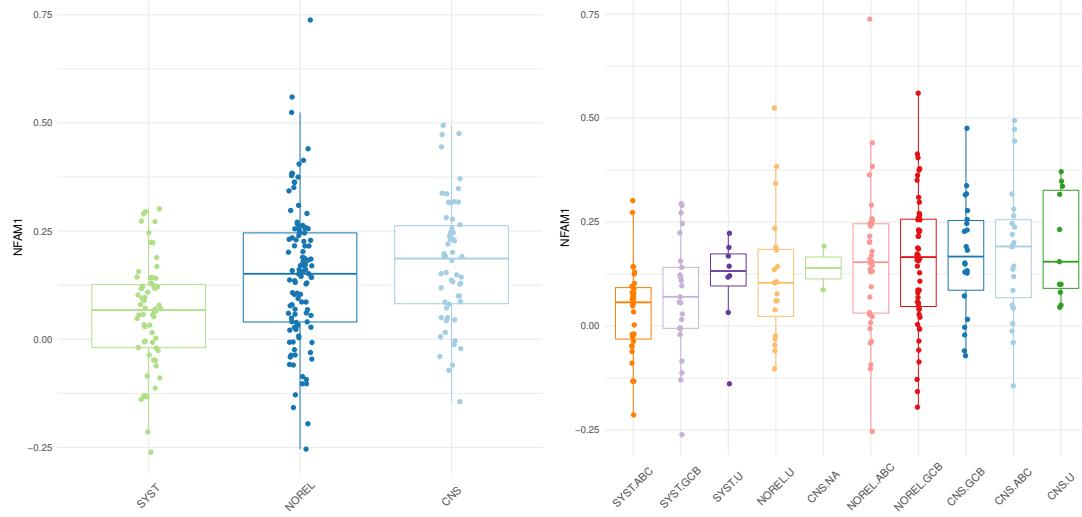
450



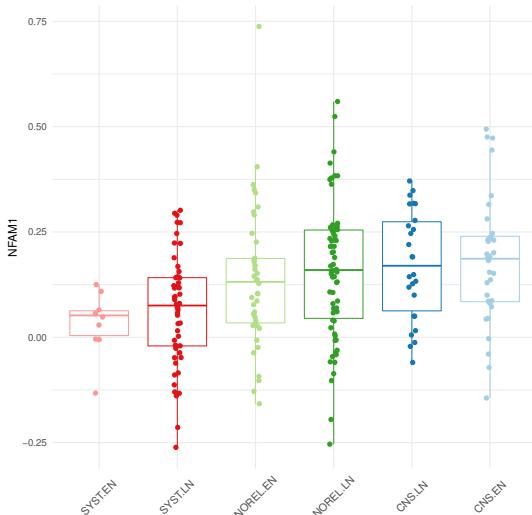
451
452

NFMA1

```
gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 2, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 2, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 2, implev = 90)
```



453



454
455

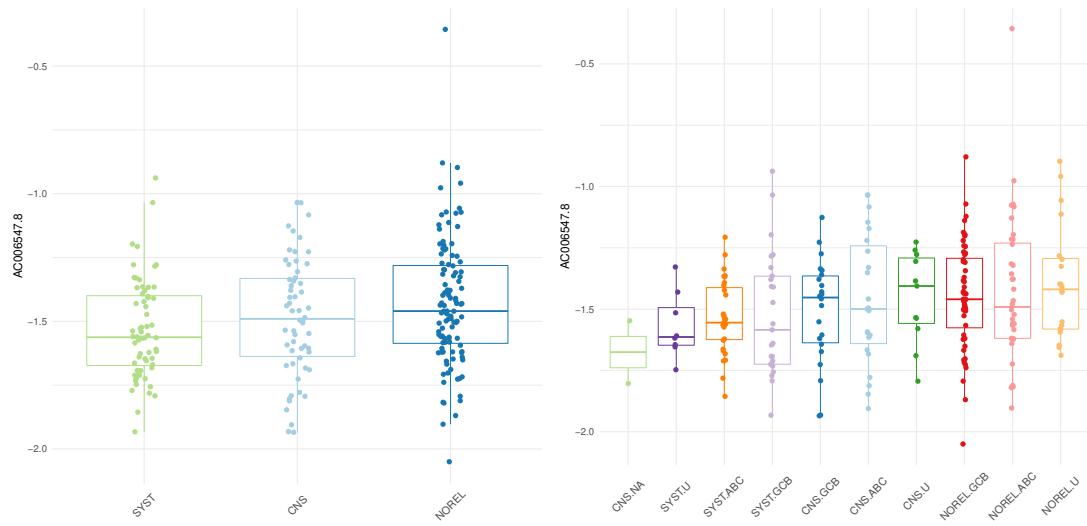
AC006547.8

```

gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 3, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 3, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 3, implev = 90)

```

456



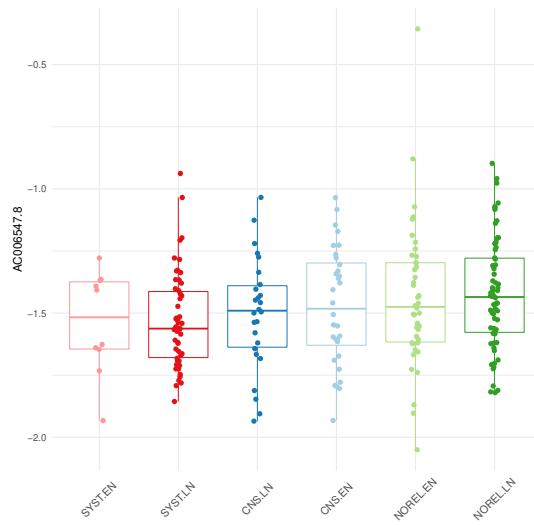
457

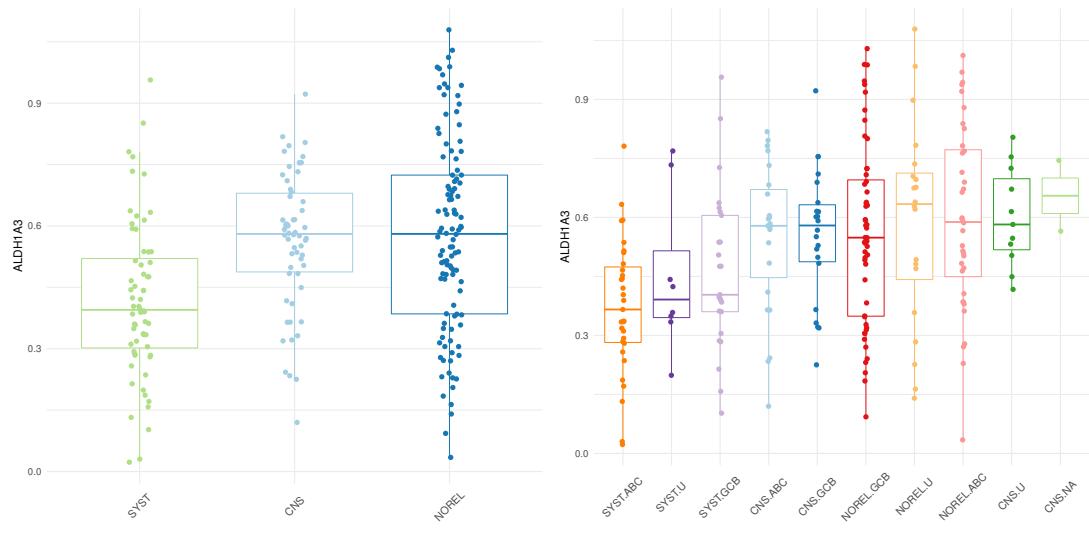
ALDH1A3

```

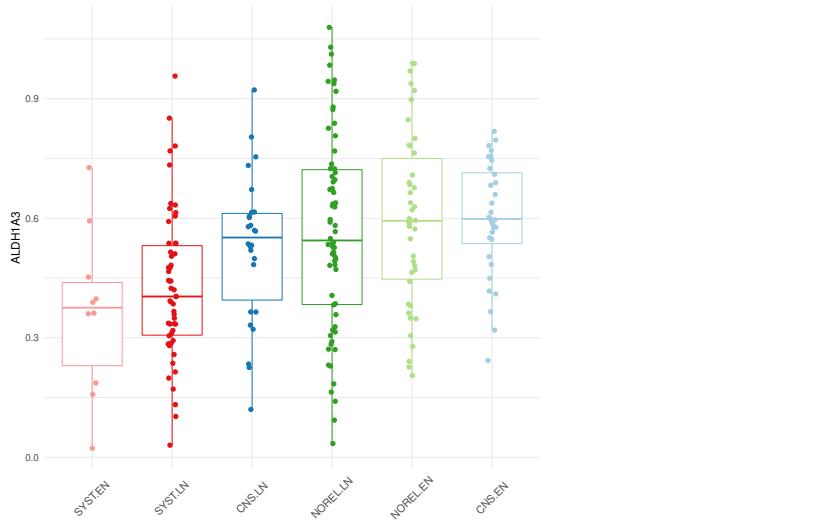
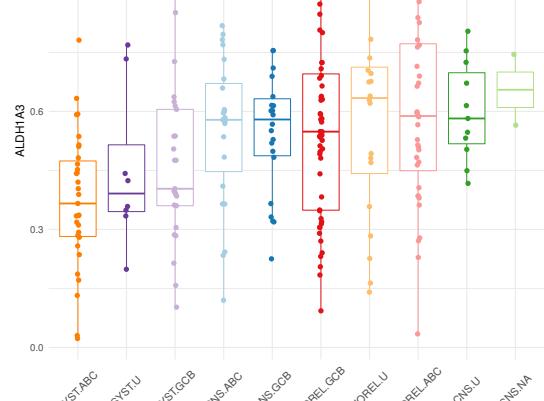
gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 4, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 4, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 4, implev = 90)

```





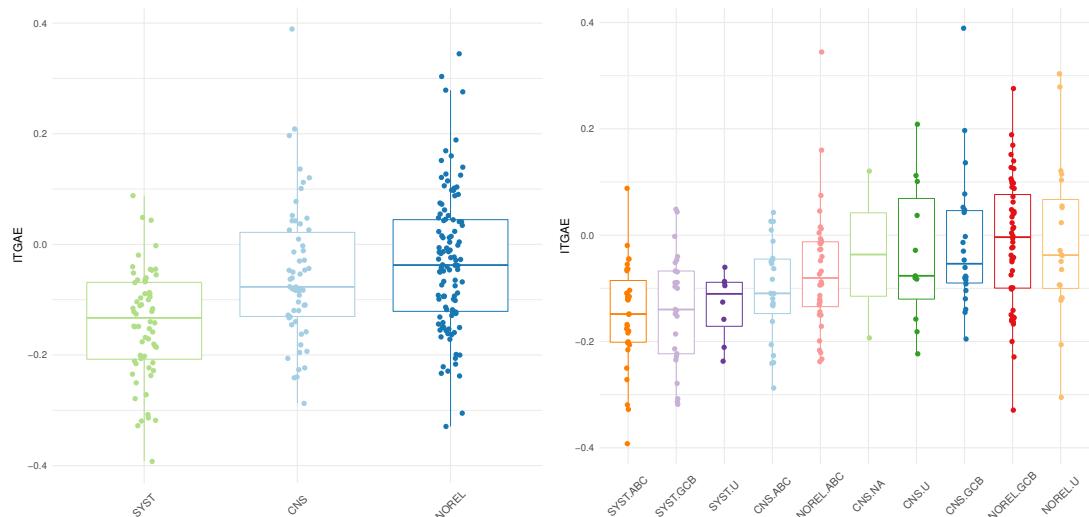
459



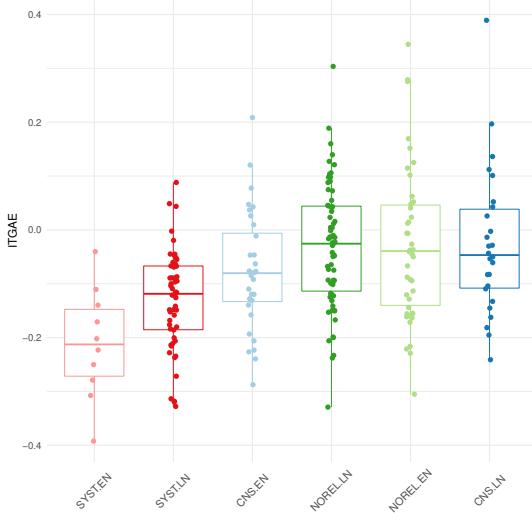
460

ITGAE

```
gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 5, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 5, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 5, implev = 90)
```



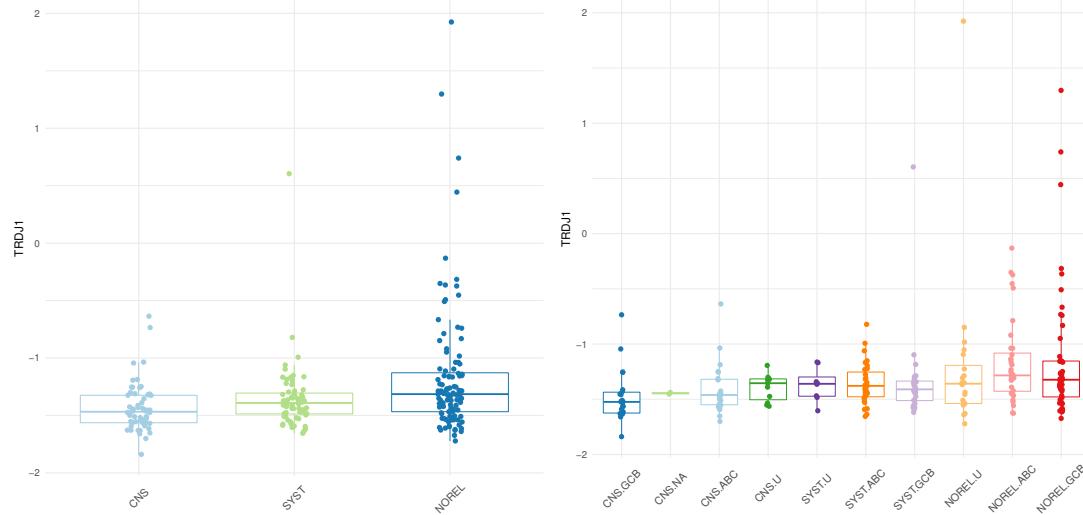
462



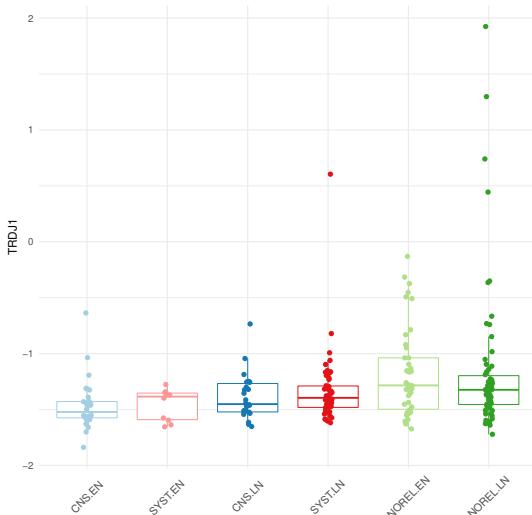
463
464

TRDJ1

```
gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 6, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 6, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 6, implev = 90)
```



465



466
467

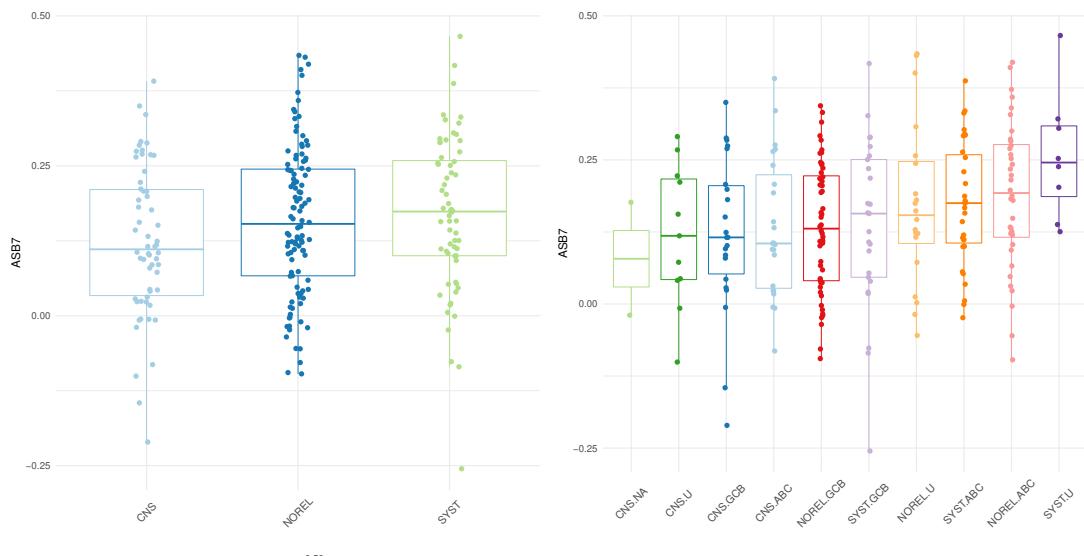
ASB7

```

gene.by.class(data = rma.genes, y = meta.selected$Groups, geneSel = 7, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast1, geneSel = 7, implev = 90)
gene.by.class(data = rma.genes, y = meta.selected$Contrast2, geneSel = 7, implev = 90)

```

468



469

4.8 Expression of important genes summarized by limma scores

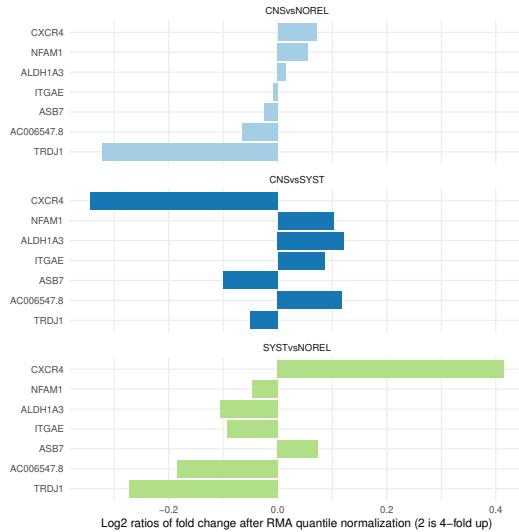
470 Expression based on log2 fold change for individuals compared based on a relapse contrast.

```

pvals <- read.table("./data/pvals.lmfit", header = TRUE)

pvals %>%
  filter(Contrast == "systemicRelapse") %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  ggplot(aes(x = reorder(genes, LogFC),
             y = LogFC,
             fill = factor(Comparison))) +
  geom_bar(stat = "identity",
            position = "dodge") +
  coord_flip() +
  scale_fill_brewer(palette="Paired") +
  facet_wrap(~ Comparison, ncol = 1) +
  theme_minimal() +
  theme(legend.position = "none") +
  xlab("") +
  ylab("Log2 ratios of fold change after RMA quantile normalization (2 is 4-fold up)")

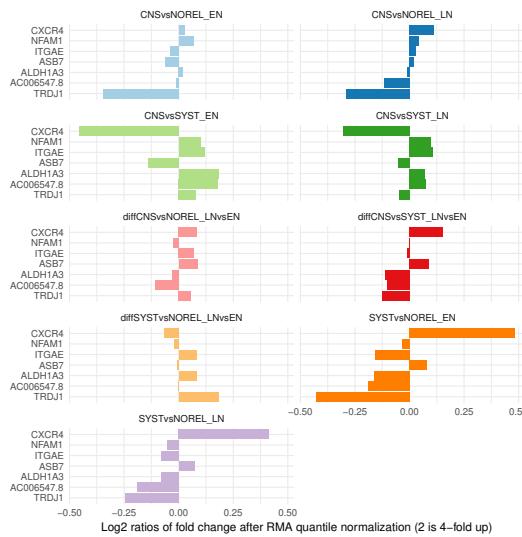
```



472
473
474

Expression based on log2 fold change for individuals compared based on involvement of lymphnodes or extra nodal sites.

```
pvals %>%
  filter(Contrast == "systemicRelapseNodes") %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  ggplot(aes(x = reorder(genes, LogFC),
             y = LogFC,
             fill = factor(Comparison))) +
  geom_bar(stat = "identity",
            position = "dodge") +
  coord_flip() +
  scale_fill_brewer(palette="Paired") +
  facet_wrap(~ Comparison, ncol = 2) +
  theme_minimal() +
  theme(legend.position = "none") +
  xlab("") +
  ylab("Log2 ratios of fold change after RMA quantile normalization (2 is 4-fold up)")
```



475
476

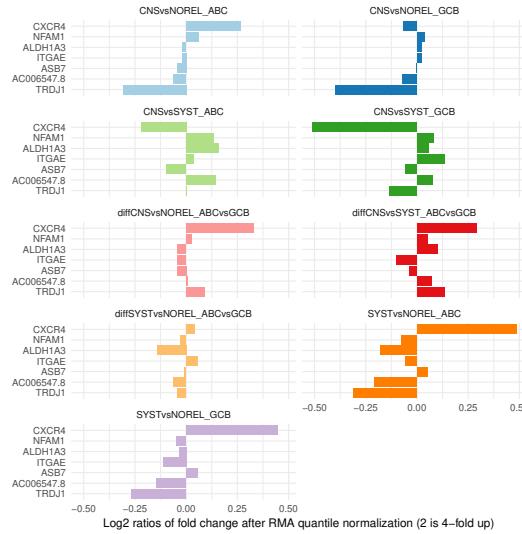
Expression based on log2 fold change for individuals compared based on involvement of cell-of-origin.

```
pvals %>%
```

```

filter(Contrast == "systemicRelapseCOOprediction") %>%
mutate(genes = gsub(".TC.*$", "", genes)) %>%
ggplot(aes(x = reorder(genes, LogFC),
           y = LogFC,
           fill = factor(Comparison))) +
geom_bar(stat = "identity",
         position = "dodge") +
coord_flip() +
scale_fill_brewer(palette="Paired") +
facet_wrap(~ Comparison, ncol = 2) +
theme_minimal() +
theme(legend.position = "none") +
xlab("") +
ylab("Log2 ratios of fold change after RMA quantile normalization (2 is 4-fold up)")

```



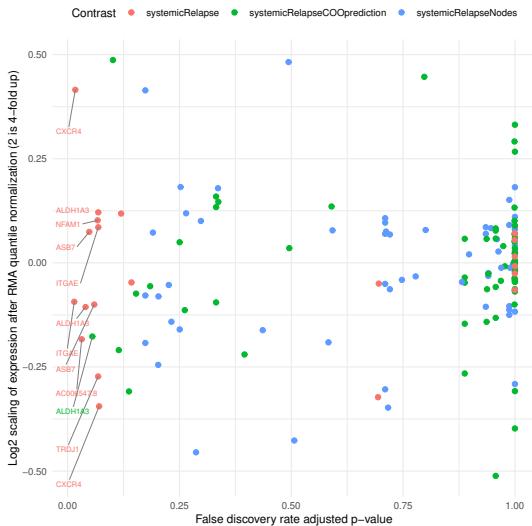
477
478 Expression based on log2fold and adjusted p-values. Labeled genes have a minimum of 0.1 adjusted p-
479 value. Duplication in gene names is possible, because each contrast uses different comparisons between
480 the groups of individuals.

```

model.names <- pvals %>%
  filter(FDRadjPval <= .1)

pvals %>%
  ggplot(aes(x = FDRadjPval,
             y = LogFC,
             color = Contrast,
             label = Symbol)) +
  geom_point(aes(color = Contrast),
             shape = as.factor(Contrast),
             size = 2) +
  geom_text_repel(data = subset(model.names, FDRadjPval <= .1),
                 nudge_x = -.5,
                 nudge_y = -.1,
                 force = 5,
                 segment.color = "grey50",
                 direction = "y",
                 segment.size = 0.1,
                 size = 2.5) +
  scale_fill_brewer(palette="Paired") +
  theme_minimal() +
  theme(legend.position = "top") +
  xlab("False discovery rate adjusted p-value") +
  ylab("Log2 scaling of expression after RMA quantile normalization (2 is 4-fold up)")

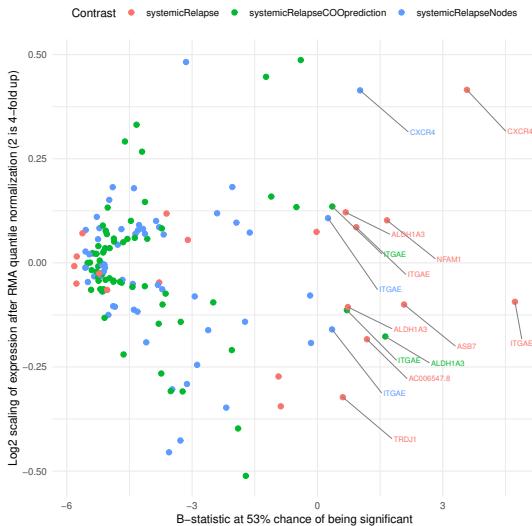
```



481
482 Expression based on log2fold and B-statistics. Labeled genes have a minimum of 0 B-statistic score (ie.,
483 53% probability of being significant). Duplication in gene names is possible, because each contrast uses
484 different comparisons between the groups of individuals.

```
model.names <- pvals %>%
  filter(B >= 0)

pvals %>%
  ggplot(aes(x = B,
             y = LogFC,
             color = Contrast,
             label = Symbol)) +
  geom_point(aes(color = Contrast),
#              shape = as.factor(Contrast)),
              size = 2) +
  geom_text_repel(data = subset(model.names, B >= 0),
                  nudge_x = 1.5,
                  nudge_y = -.1,
                  force = 5,
                  segment.color = "grey50",
                  direction = "y",
                  segment.size = 0.1,
                  size = 2.5) +
  scale_fill_brewer(palette="Dark2") +
  theme_minimal() +
  theme(legend.position = "top") +
  xlab("B-statistic at 53% chance of being significant") +
  ylab("Log2 scaling of expression after RMA quantile normalization (2 is 4-fold up)")
```



485

4.9 All classifying genes visualized by prognosis classes

Without clustering, visualizing the expression of all selected genes used for classification. Log2 FC was scaled to 0-1 for better interpretability.

489

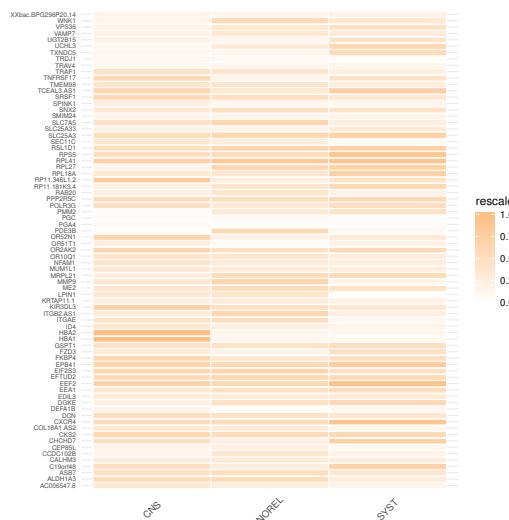
4.9.1 Top classifier genes

491 Gene log₂ ratios for individuals when grouped according to CNS or systemic relapse, or no relapse.

```

data.frame(y=meta.selected$Groups, t(rma.genes)) %>%
  melt() %>%
  dplyr::ddply( .(variable), transform, rescale = rescale(value)) %>%
  mutate(genes = gsub(".TC.*$", "", variable)) %>%
  ggplot(aes(y, genes)) +
  geom_tile(aes(fill = rescale), colour = "white") +
  scale_fill_gradient(low = "white", high = "#fdc086") +
  xlab("") + ylab("") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6)) +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 9))

```

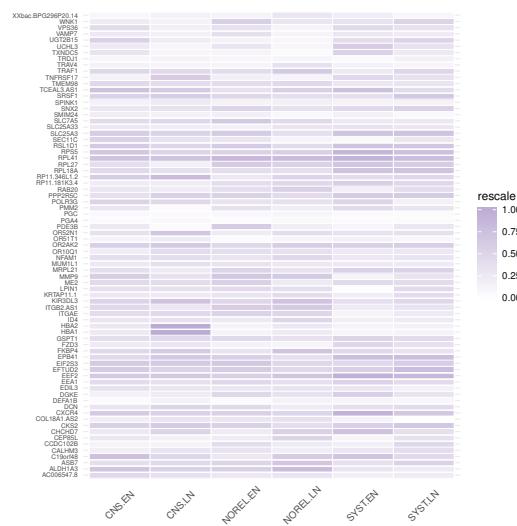


193

493 Gene log₂ ratios for individuals when grouped according to lymphnodes or extranodal involvement.

```
data.frame(v=meta.selected$Contrast2, t(rma.genes))
```

```
melt() %>%
ddply( .(variable), transform, rescale = rescale(value)) %>%
mutate(genes = gsub(".TC.*$", "", variable)) %>%
ggplot(aes(y, genes)) +
geom_tile(aes(fill = rescale), colour = "white") +
scale_fill_gradient(low = "white", high = "#beaed4") +
xlab("") + ylab("") +
theme_minimal() +
theme(axis.text.y = element_text(size = 6)) +
theme(axis.text.x = element_text(vjust = .5,
                                 angle = 45,
                                 size = 9))
```



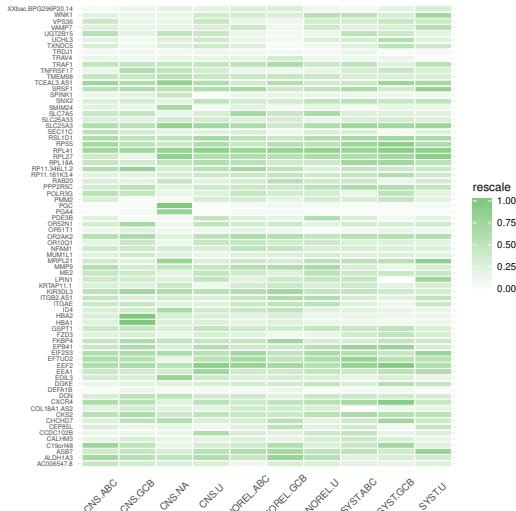
494

495 Gene log2 ratios for individuals when grouped according to cell-of-origin.

```

data.frame(y=meta.selected$Contrast1, t(rma.genes)) %>%
  melt() %>%
  ddpyle(.variable, transform, rescale = rescale(value)) %>%
  mutate(genes = gsub(".TC.*$", "", variable)) %>%
  ggplot(aes(y, genes)) +
  geom_tile(aes(fill = rescale), colour = "white") +
  scale_fill_gradient(low = "white", high = "#7fc97f") +
  xlab("") + ylab("") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6)) +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 9))

```



496

4.9.2 Top importance genes

498

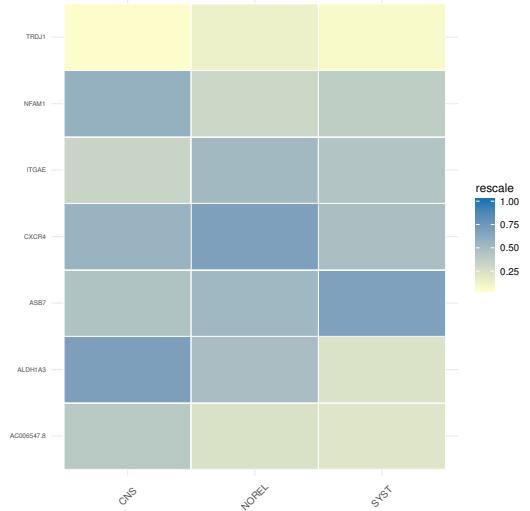
The selected genes with highest importance score across models, showing log2 ratios for individuals when grouped according to relapse classes.

```

sel.genes <- imp.model %>%
  mutate(genes = gsub(".TC.*$", "", genes)) %>%
  gather("groups", "importance", 3:5) %>%
  filter(importance >= 90) %>%
  select(genes) %>%
  unique

data.frame(y=meta.selected$Groups, t(rma.genes)) %>%
  melt() %>%
  ddply( .(variable), transform, rescale = rescale(value)) %>%
  mutate(genes = gsub(".TC.*$", "", variable)) %>%
  filter(genes == sel.genes$genes) %>%
  ggplot(aes(y, genes)) +
  geom_tile(aes(fill = rescale), colour = "white") +
  scale_fill_gradient(low = "#fffffc", high = "#1f78b4") +
  xlab("") + ylab("") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6)) +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 9))

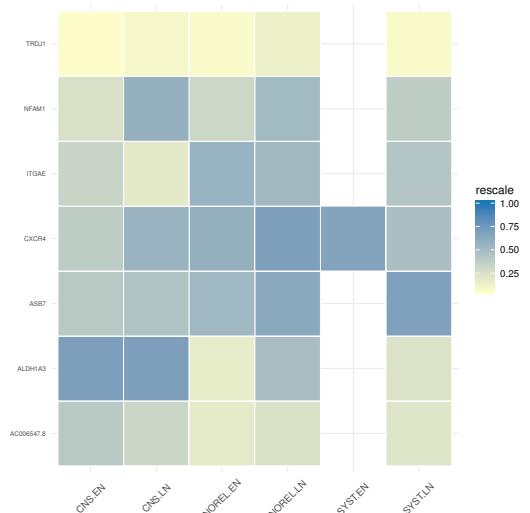
```



500
501 The selected genes with highest importance score across models, showing log2 ratios for individuals
502 when grouped according to nodal classes.

```
data.frame(y=meta.selected$Contrast2, t(rma.genes)) %>%
  melt() %>%
  ddply( .(variable), transform, rescale = rescale(value)) %>%
  mutate(genes = gsub(".TC.*$", "", variable)) %>%
  filter(genes == sel.genes$genes) %>%
  ggplot(aes(y, genes)) +
  geom_tile(aes(fill = rescale), colour = "white") +
  scale_fill_gradient(low = "#fffffc", high = "#1f78b4") +
  xlab("") + ylab("") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6)) +
  theme(axis.text.x = element_text(vjust = .5,
                                   angle = 45,
                                   size = 9))
```

Warning in genes == sel.genes\$genes: longer object length is not a multiple of shorter object length



503
504 The selected genes with highest importance score across models, showing log2 ratios for individuals
505 when grouped according to cell-of-origin.

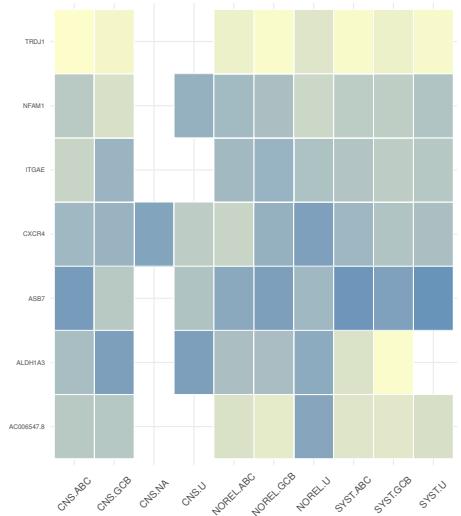
```
data.frame(y=meta.selected$Contrast1, t(rma.genes)) %>%
```

```

melt() %>%
ddply( .(variable), transform, rescale = rescale(value)) %>%
mutate(genes = gsub(".TC.*$", "", variable)) %>%
filter(genes == sel.genes$genes) %>%
ggplot(aes(y, genes)) +
geom_tile(aes(fill = rescale), colour = "white") +
scale_fill_gradient(low = "#ffffcc", high = "#1f78b4") +
xlab("") + ylab("") +
theme_minimal() +
theme(axis.text.y = element_text(size = 6)) +
theme(axis.text.x = element_text(vjust = .5,
                                 angle = 45,
                                 size = 9))

```

Warning in genes == sel.genes\$genes: longer object length is not a multiple of shorter object length



506
507

4.10 Version of machine learning models on high performance clusters

[†] Version of R packages used in the machine learning pipeline

```

## R version 3.5.0 (2018-04-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.4 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblas-r0.2.18.so
##
## locale:
## [1] LC_CTYPE=en_CA.UTF-8          LC_NUMERIC=C
## [3] LC_TIME=en_CA.UTF-8          LC_COLLATE=en_CA.UTF-8
## [5] LC_MONETARY=en_CA.UTF-8       LC_MESSAGES=en_CA.UTF-8
## [7] LC_PAPER=en_CA.UTF-8          LC_NAME=C
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8   LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics    grDevices  utils      datasets   methods    base
##
## other attached packages:
## [1] pls_2.6-0            bindrcpp_0.2.2    plot3D_1.1.1      plyr_1.8.4
## [5] tidyr_0.8.1          reshape2_1.4.3    vegan_2.5-2        permute_0.9-4
## [9] earth_4.6.3          plotmo_3.4.0     TeachingDemos_2.10  plotrix_3.7-2
## [13] ROCR_1.0-7           doSNOW_1.0.16     snow_0.4-2         iterators_1.0.9
## [17] caret_6.0-80          ggplot2_3.0.0     lattice_0.20-35    glmnet_2.0-16
## [21] foreach_1.4.4         Matrix_1.2-14     dplyr_0.7.6        gplots_3.0.1
## [25] pvclust_2.0-0          RColorBrewer_1.1-2 #####

```

```

#####
## [1] minqa_1.2.4          colorspace_1.3-2    class_7.3-14      ##
## [4] DRR_0.0.3              svUnit_0.7-12       prodlim_2018.04.18 ##
## [7] lubridate_1.7.4        codetools_0.2-15    splines_3.5.0      ##
## [10] mnormt_1.5-5          robustbase_0.93-0   RcppRoll_0.2.2      ##
## [13] mda_0.4-10             broom_0.4.4         ddalpha_1.3.3      ##
## [16] cluster_2.0.7-1       kernlab_0.9-26     sfsmisc_1.1-2      ##
## [19] compiler_3.5.0         assertthat_0.2.0    lazyeval_0.2.1      ##
## [22] FCNN4R_0.6.2          Rcgmin_2013-2.21   optextras_2016-8.8 ##
## [25] tools_3.5.0            igraph_1.2.1        misc3d_0.8-4       ##
## [28] gtable_0.2.0            glue_1.2.0          LiblineaR_2.10-8    ##
## [31] naivebayes_0.9.2       Rcpp_0.12.18       gdata_2.18.0       ##
## [34] nlme_3.1-137           setRNG_2013.9-1    psych_1.8.4        ##
## [37] timeDate_3043.102     gower_0.1.2        stringr_1.3.1      ##
## [40] kknn_1.3.1              gtools_3.5.0        DEoptimR_1.0-8      ##
## [43] Rvmmin_2018-4.17       MASS_7.3-50        scales_1.0.0       ##
## [46] ipred_0.9-6             parallel_3.5.0    monmlp_1.1.5       ##
## [49] rpart_4.1-13            stringi_1.2.2     ucminf_1.1-4       ##
## [52] randomForest_4.6-14    deepnet_0.2        e1071_1.6-8       ##
## [55] BB_2014.10-1            caTools_1.17.1    optimx_2013.8.7    ##
## [58] lava_1.6.1              geometry_0.3-6    rlang_0.2.1        ##
## [61] pkgconfig_2.0.1          bitops_1.0-6       purrr_0.2.5        ##
## [64] bindr_0.1.1              recipes_0.1.2     labeling_0.3       ##
## [67] CVST_0.2-2              tidyselect_0.2.4   gbm_2.1.3         ##
## [70] magrittr_1.5              R6_2.2.2          dimRed_0.1.0       ##
## [73] pillar_1.2.3             foreign_0.8-70    withr_2.1.2       ##
## [76] mgcv_1.8-23              survival_2.42-3   abind_1.4-5       ##
## [79] nnet_7.3-12              tibble_1.4.2      KernSmooth_2.23-15 ##
## [82] RRF_1.7                  grid_3.5.0        ModelMetrics_1.1.0  ##
## [85] digest_0.6.15            dfoptim_2018.2-1  numDeriv_2016.8-1  ##
## [88] stats4_3.5.0              munsell_0.5.0    magic_1.5-8       ##
## [91] quadprog_1.5-5          #####

```

5 Clonal evolution

The allele frequency can be indicative if the mutation called, whether it is of germline or somatic origin. Worth investigating if the call is either 100% or 50%. That is all reads show the variant (100%), it's very likely a germline call. Unless the sample is 100% tumor cells and all tumor cells have the mutation. If it is a somatic mutation and the samples are of clinical nature, then a normal cell infiltration is inevitable. The allele frequency for these mutations will be at 30-40% ([Zainal 2012](#)).

- 514 • Flag known somatic mutations using the COSMIC database
- 515 • Flag known germline mutations using the dbSNP database
- 516 • Filtering by allele frequency
- 517 • Distribution of read mismatches (using bamtools XM:O and XT:U, as in unique)
- 518 • [Koboldt 2012](#) authors of Varscan2 recommended passed somatic mutations
- 519 • Use the [1000 Genomes project](#)

5.1 Distribution of variants by estimated allele frequency

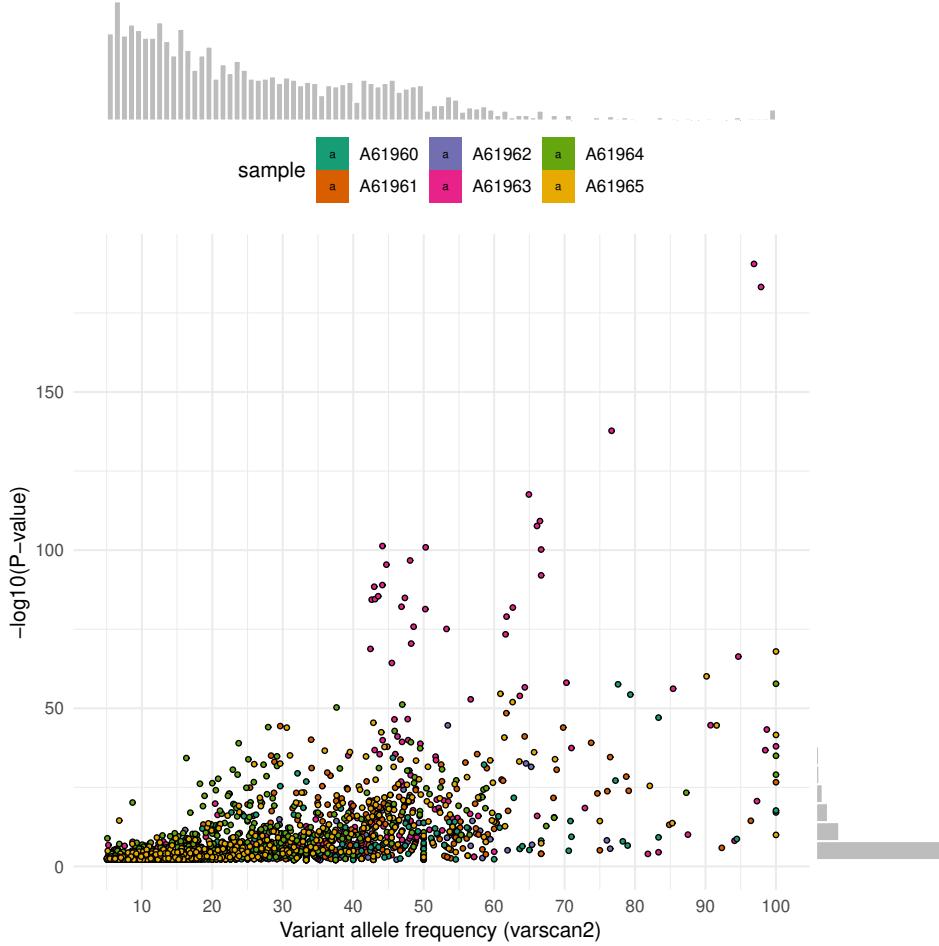
Exome sequencing of tumor samples generated several libraries of reads. Allele frequency distribution was measured by calling variants on the reads files. These variants were thresholded with P-value at 10e-25, depth of coverage by point mutation at 50, variant allele frequency at 1% (discard below 1%), and pyclone normal-minor-major copy numbers at 2-0-2 with major-copy-number sets. Also, some variants were removed based on their known locations from VarScan annotations.

```
snv.post.filter <- read.table("./data/pyclone/output.10_bash.summary.snv.postFilter_VAFpval.varscan2.txt"
                                header = T, fill = TRUE)

model.names <- snv.post.filter %>%
  mutate(pval = -log10(pval)) %>%
  filter(vaf >= 35) %>%
  filter(vaf <= 85)

p <- snv.post.filter %>%
  mutate(pval = -log10(pval)) %>%
  ggplot(aes(x = vaf,
              y = pval,
              label = snv,
              fill = sample)) +
  geom_point(size = 1, shape = 21) +
  geom_label_repel(data = subset(model.names, pval <= 0),
                     nudge_x = 1.5,
                     nudge_y = 200,
                     force = 10,
                     segment.color = "grey50",
                     direction = "y",
                     segment.size = 0.1,
                     size = 2) +
  scale_fill_brewer(palette="Dark2") +
  theme_minimal() +
  scale_x_continuous(breaks = pretty(snv.post.filter$vaf, n = 10)) +
  xlab("Variant allele frequency (varsan2)") +
  ylab("-log10(P-value)") +
  theme(legend.position = "top")

ggMarginal(p, type = "histogram", fill="transparent",
            color = "white",
            xparams = list(binwidth = 1, fill = "grey"),
            yparams = list(binwidth = 6, fill = "grey"))
```



526

5.2 Allele frequencies and variants using different callers

5.2.1 Individual 1

Varscan was used to call for variants against the hg19 GRCh37 human genome. The amount of variants was reduced based on read depth per site called. Stringency was relaxed and the depth was set between 40 and 400 for all individuals.

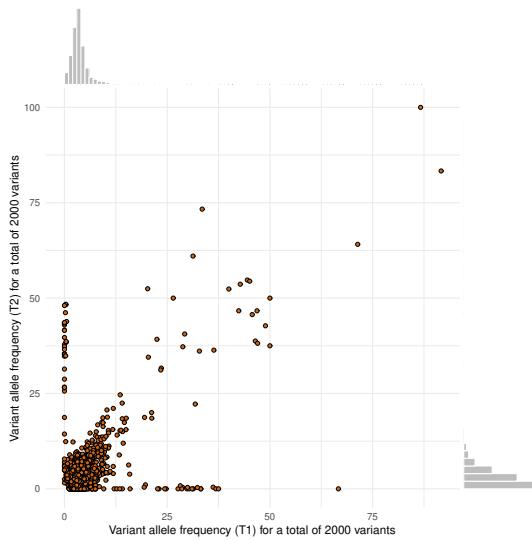
```

varscan_af <- read.table("./data/pyclone/reduced_variants/output.4_mined.varscan2.snpeff_annot_genome.c

plot.af <- function(df, pval) {
  p <- df %>%
    filter(s1_pval <= pval) %>%
    filter(s2_pval <= pval) %>%
    ggplot(aes(x = s1_freq,
               y = s2_freq)) +
    geom_point(size = 1.5, fill = "chocolate", shape = 21) +
    xlab("Variant allele frequency (T1) for a total of 2000 variants") +
    ylab("Variant allele frequency (T2) for a total of 2000 variants") +
    theme(legend.position = "top") +
    theme_minimal()
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = 1, fill = "grey"),
             yparams = list(binwidth = 2, fill = "grey"))
}

plot.af(df = varscan_af, pval = 1)

```



532

533 Same as above and with logarithmic scaling.

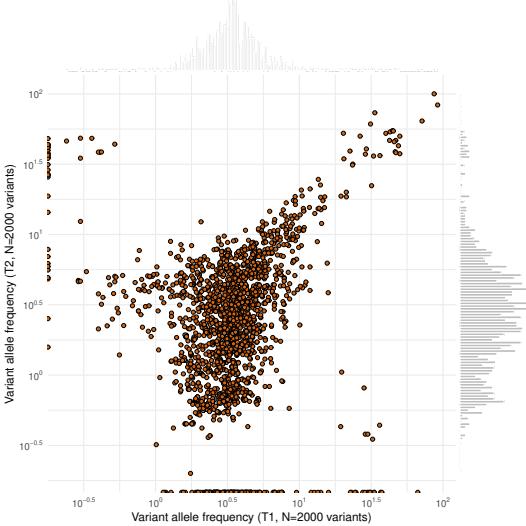
```

plot.af.logged <- function(df, pval) {
  p <- df %>%
    filter(s1_pval <= pval) %>%
    filter(s2_pval <= pval) %>%
    ggplot(aes(x = s1_freq,
                  y = s2_freq)) +
    geom_point(size = 1.5, fill = "chocolate", shape = 21) +
    xlab("Variant allele frequency (T1, N=2000 variants)") +
    ylab("Variant allele frequency (T2, N=2000 variants)") +
    scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
                    labels = trans_format("log10", math_format(10^.x))) +
    scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                    labels = trans_format("log10", math_format(10^.x))) +
    theme(legend.position = "top") +
    theme_minimal()
  ggMarginal(p, type = "histogram", fill="transparent",
              color = "white",
              xparams = list(binwidth = .01, fill = "grey"),
              yparams = list(binwidth = .02, fill = "grey"))
}

plot.af.logged(df = varscan_af, pval = 1)

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 34 rows containing non-finite values (stat_bin).
Warning: Removed 157 rows containing non-finite values (stat_bin).

```

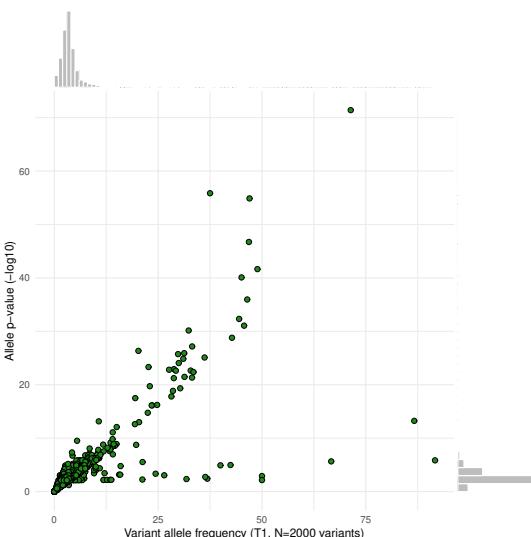


534
535

Allele frequency versus p-value of called variants. For each timepoint.

```
plot.pval.t1 <- function(df) {
  p <- varsanc_af %>%
    mutate(s1_pval = -log10(s1_pval)) %>%
    ggplot(aes(x = s1_freq,
               y = s1_pval)) +
    geom_point(size = 2, fill = "forestgreen", shape = 21) +
    xlab("Variant allele frequency (T1, N=2000 variants)") +
    ylab("Allele p-value (-log10)") +
    theme(legend.position = "top") +
    theme_minimal()
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = 1, fill = "grey"),
             yparams = list(binwidth = 1.5, fill = "grey"))
}

plot.pval.t1(df = varsanc_af)
```



536
537

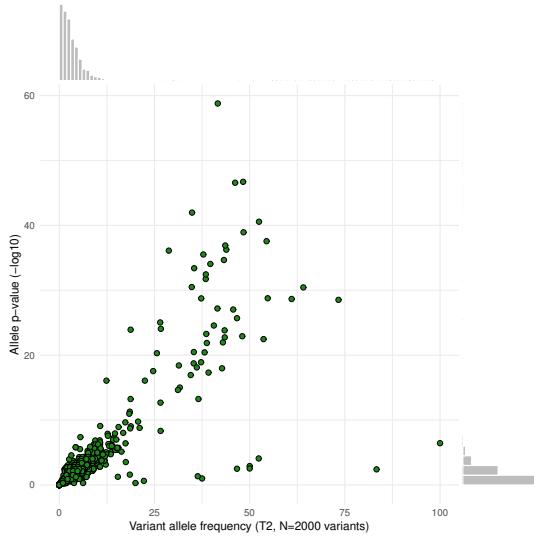
Allele frequency versus p-value of called variants. For each timepoint.

```
plot.pval.t2 <- function(df) {
```

```

p <- varscan_af %>%
  mutate(s2_pval = -log10(s2_pval)) %>%
  ggplot(aes(x = s2_freq,
             y = s2_pval)) +
  geom_point(size = 2, fill = "forestgreen", shape = 21) +
  xlab("Variant allele frequency (T2, N=2000 variants)") +
  ylab("Allele p-value (-log10)") +
  theme(legend.position = "top") +
  theme_minimal()
ggMarginal(p, type = "histogram", fill="transparent",
           color = "white",
           xparams = list(binwidth = 1, fill = "grey"),
           yparams = list(binwidth = 1.5, fill = "grey"))
}
plot.pval.t2(df = varscan_af)

```

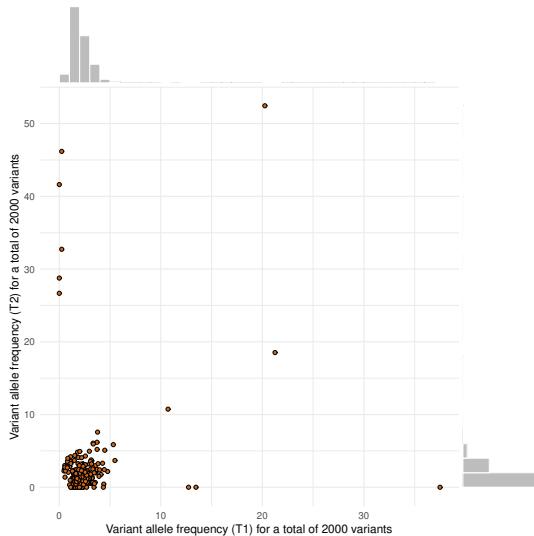


538

539 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
varscan_af <- read.table("./data/pyclone/stringent/output.4_mined.varscan2.snpeff_annot_genome.clinvar.
```

```
plot.af(df = varscan_af, pval = 1)
```



540

541 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

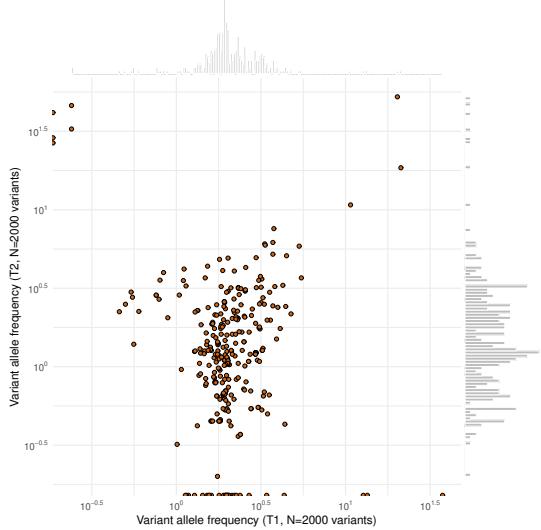
542

```
plot.af.logged(df = varscan_af, pval = 1)
```

```

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 3 rows containing non-finite values (stat_bin).
Warning: Removed 29 rows containing non-finite values (stat_bin).

```



543

544 Bcftools was also used to call for variants against the hg19 GRCh37 human genome.

[†] Varscan and bcftools calls for a total of 2000 variants

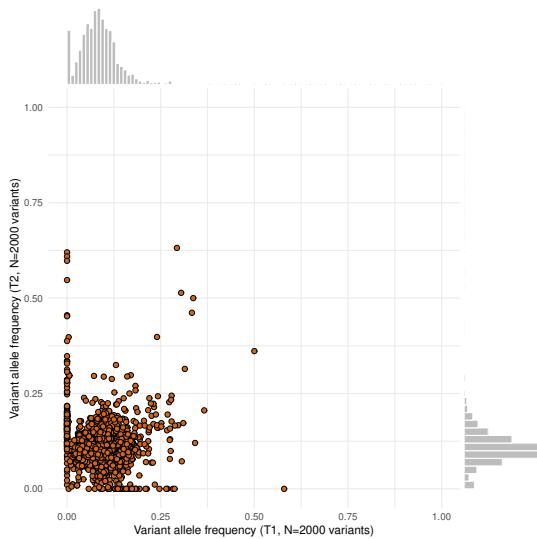
```

bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcftools.snpeff_annot_genome.c

plot.baf <- function(df) {
  p <- bcftools_af %>%
    mutate(af1 = c(s1_ad/dp)) %>%
    mutate(af2 = c(s2_ad/dp)) %>%
    ggplot(aes(x = af1,
               y = af2)) +
    geom_point(size = 2, fill = "chocolate", shape = 21) +
    scale_y_continuous(limits = c(0,1)) +
    scale_x_continuous(limits = c(0,1)) +
    xlab("Variant allele frequency (T1, N=2000 variants)") +
    ylab("Variant allele frequency (T2, N=2000 variants)") +
    theme(legend.position = "top") +
    theme_minimal()
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = .01, fill = "grey"),
             yparams = list(binwidth = .02, fill = "grey"))
}

plot.baf(df = bcftools_af)

```



545

546 Same as above with logarithmic scaling.

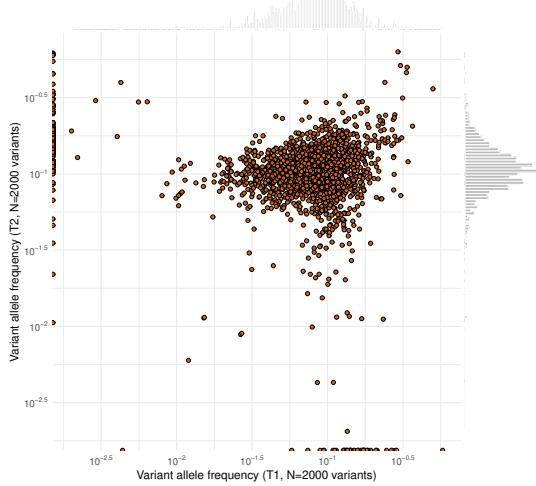
```

plot.baf.logged <- function(df) {
  p <- bcftools_af %>%
    mutate(af1 = c(s1_ad/dp)) %>%
    mutate(af2 = c(s2_ad/dp)) %>%
    ggplot(aes(x = af1,
               y = af2)) +
    geom_point(size = 1.5, fill = "chocolate", shape = 21) +
    scale_x_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x))) +
    scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                  labels = trans_format("log10", math_format(10^.x))) +
    xlab("Variant allele frequency (T1, N=2000 variants)") +
    ylab("Variant allele frequency (T2, N=2000 variants)") +
    theme(legend.position = "top") +
    theme_minimal()
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = .01, fill = "grey"),
             yparams = list(binwidth = .02, fill = "grey"))
}

plot.baf.logged(df = bcftools_af)

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 129 rows containing non-finite values (stat_bin).
Warning: Removed 45 rows containing non-finite values (stat_bin).

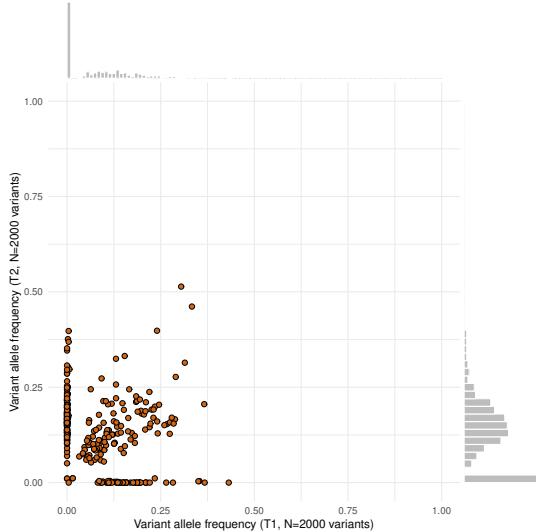
```



547

Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
bcftools_af <- read.table("./data/pyclone/stringent/output.4_mined.bcftools.snpeff_annot_genome.clinvar")
plot.baf(df = bcftools_af)
```



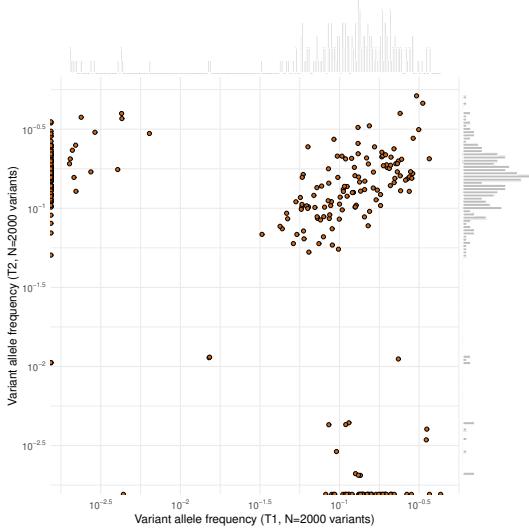
549

(Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

551

```
plot.baf.logged(df = bcftools_af)

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 114 rows containing non-finite values (stat_bin).
Warning: Removed 45 rows containing non-finite values (stat_bin).
```

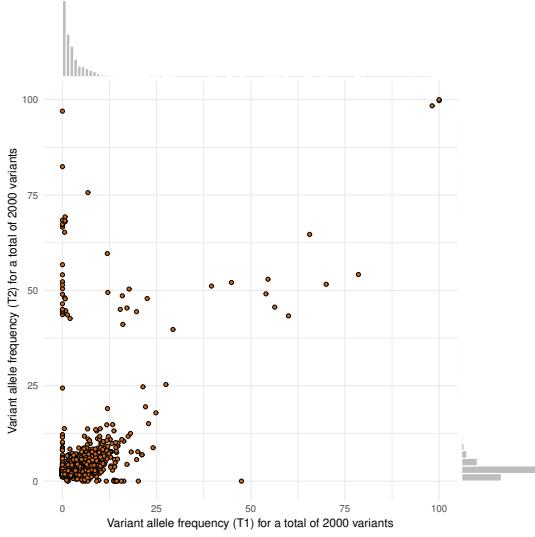


552

5.2.2 Individual 2

Varscan was used to call for variants against the hg19 GRCh37 human genome.

```
varsan_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.varscan2.snpEff_annot_genome.cl")
plot.af(df = varsan_af, pval = 1)
```

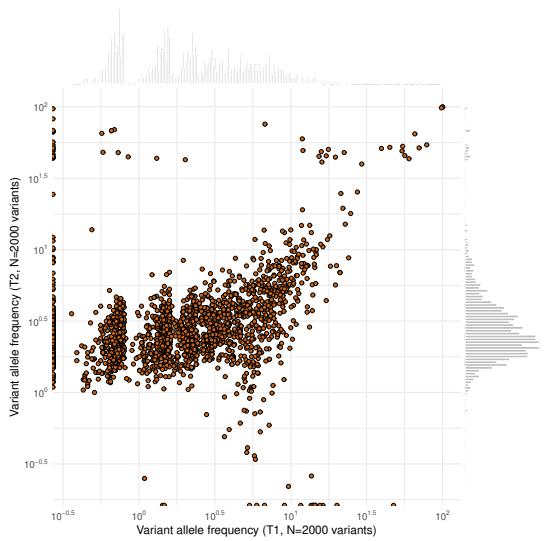


555

Logarithmic

```
plot.af.logged(df = varsan_af, pval = 1)
```

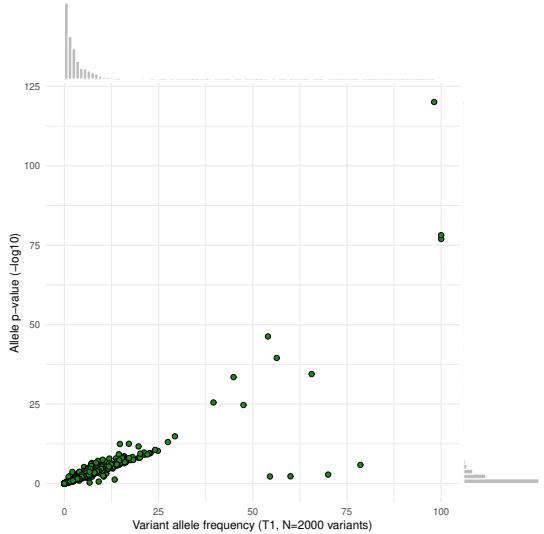
```
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 307 rows containing non-finite values (stat_bin).
Warning: Removed 17 rows containing non-finite values (stat_bin).
```



557

558 Allele frequency versus p-value of called variants. For each timepoint.

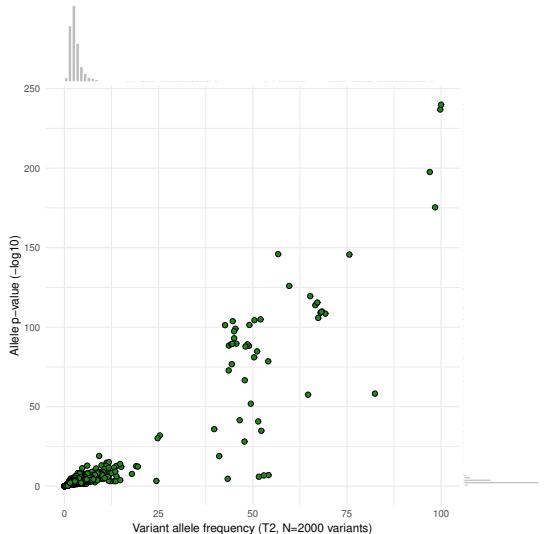
```
plot.pval.t1(df = varscan_af)
```



559

560 Allele frequency versus p-value of called variants. For each timepoint.

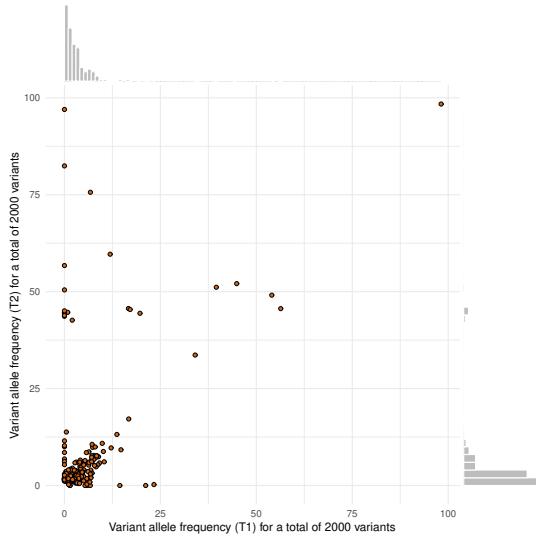
```
plot.pval.t2(df = varscan_af)
```



561

562 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

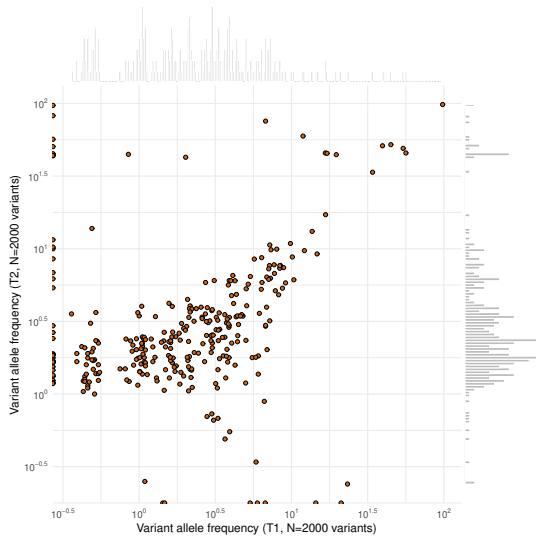
```
varscan_af <- read.table("./data/pyclone/stringent/output.4_mined.varscan2.snpeff_annot_genome.clinvar.  
plot.af(df = varscan_af, pval = 1)
```



563
564 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
565

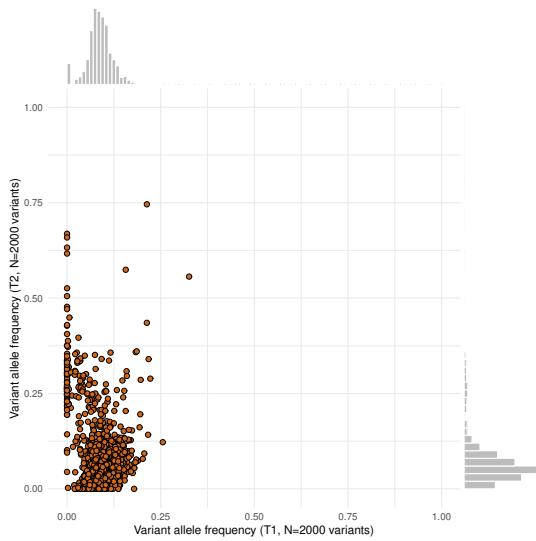
```
plot.af.logged(df = varscan_af, pval = 1)
```

```
Warning: Transformation introduced infinite values in continuous x-axis  
Warning: Transformation introduced infinite values in continuous y-axis  
Warning: Transformation introduced infinite values in continuous x-axis  
Warning: Transformation introduced infinite values in continuous y-axis  
Warning: Removed 37 rows containing non-finite values (stat_bin).  
Warning: Removed 7 rows containing non-finite values (stat_bin).
```



566
567 Bcftools was used to call for variants against the hg19 GRCh37 human genome.

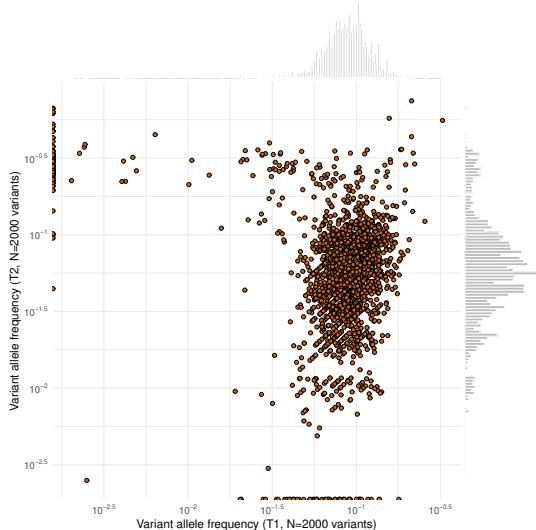
```
bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcftools.snpeff_annot_genome.c  
plot.baf(df = bcftools_af)
```



568
569 Bcftools was used to call for variants against the hg19 GRCh37 human genome.

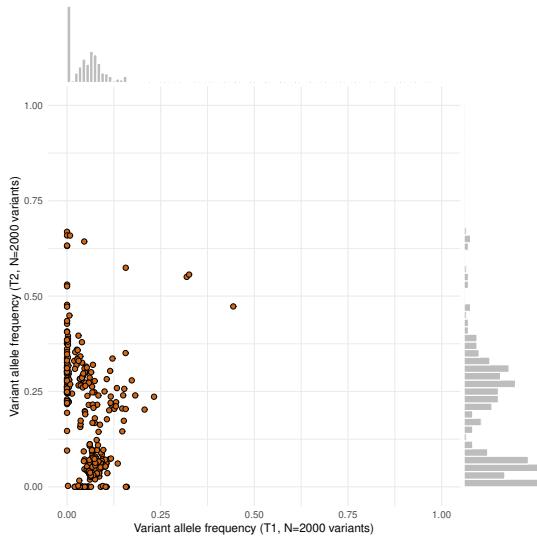
```
bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcfTools.snpeff_annot_genome.c
plot.baf.logged(df = bcftools_af)

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 72 rows containing non-finite values (stat_bin).
Warning: Removed 55 rows containing non-finite values (stat_bin).
```



570
571 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

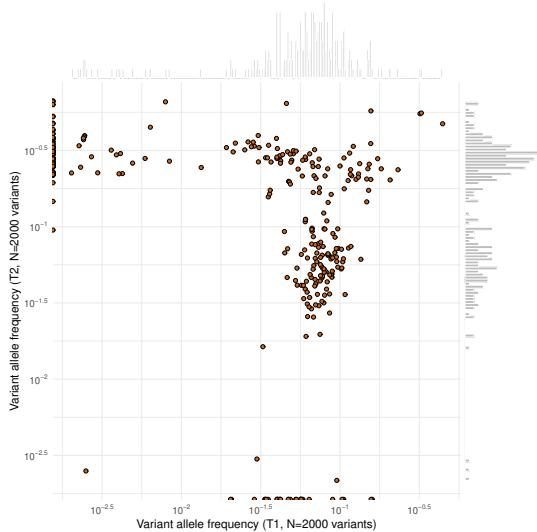
```
bcftools_af <- read.table("./data/pyclone/stringent/output.4_mined.bcfTools.snpeff_annot_genome.clinvar
plot.baf(df = bcftools_af)
```



572
573 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
574

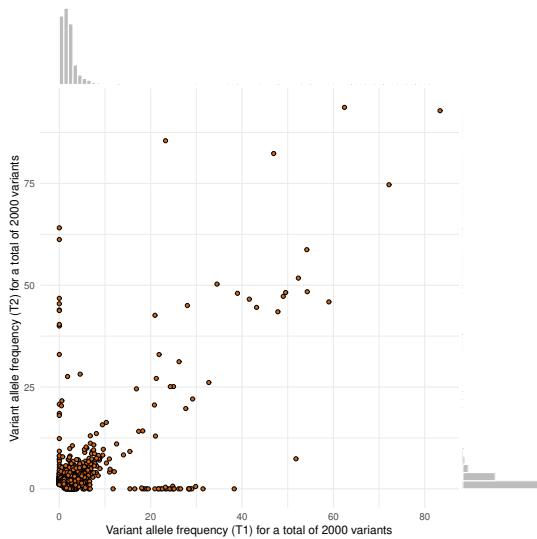
```
plot.baf.logged(df = bcftools_af)
```

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 68 rows containing non-finite values (stat_bin).
Warning: Removed 29 rows containing non-finite values (stat_bin).



575
576 **5.2.3 Individual 3**
577 Varscan was used to call for variants against the hg19 GRCh37 human genome.

```
varscan_af<- read.table("./data/pyclone/reduced_variants/output.4_miner.varscan2.snpEff_annot_genome.cl")
plot.af(df = varscan_af, pval = 1)
```

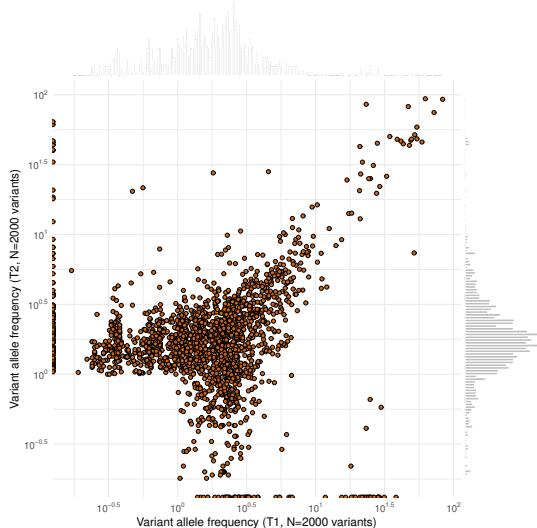


578
579

Logarithmic

```
plot.af.logged(df = varscan_af, pval = 1)
```

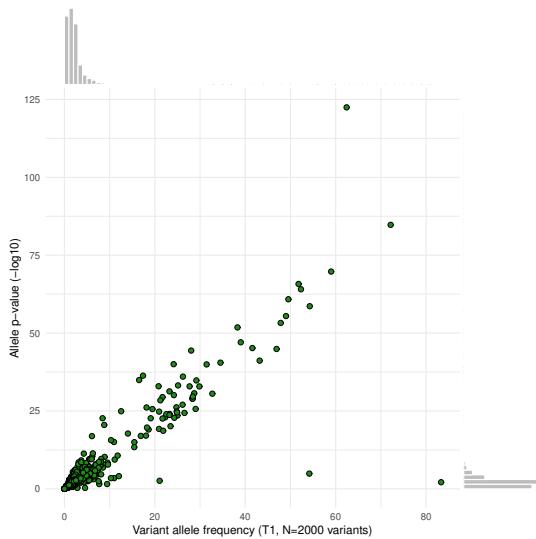
Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 114 rows containing non-finite values (stat_bin).
 Warning: Removed 71 rows containing non-finite values (stat_bin).



580
581

Allele frequency versus p-value of called variants. For each timepoint.

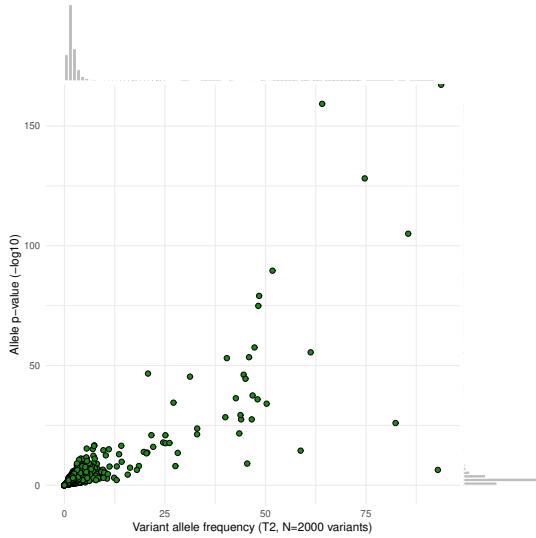
```
plot.pval.t1(df = varscan_af)
```



582
583 Allele frequency versus p-value of called variants. For each timepoint.

```
plot.pval.t2(df = varscan_af)

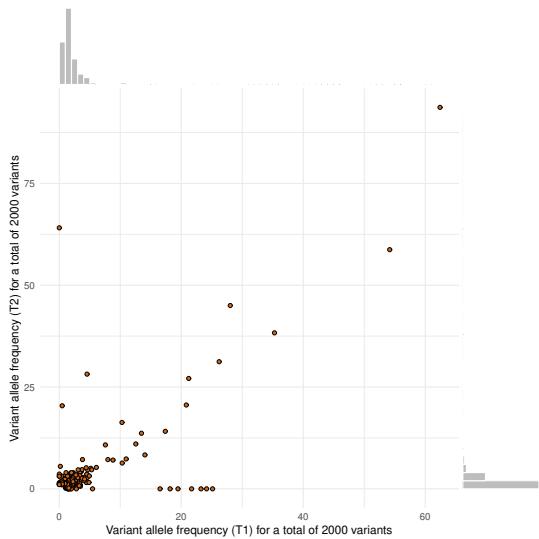
Warning: Removed 1 rows containing non-finite values (stat_bin).
```



584
585 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
varscan_af <- read.table("./data/pyclone/stringent/output.4_mined.varscan2.snpeff_annot_genome.clinvar.

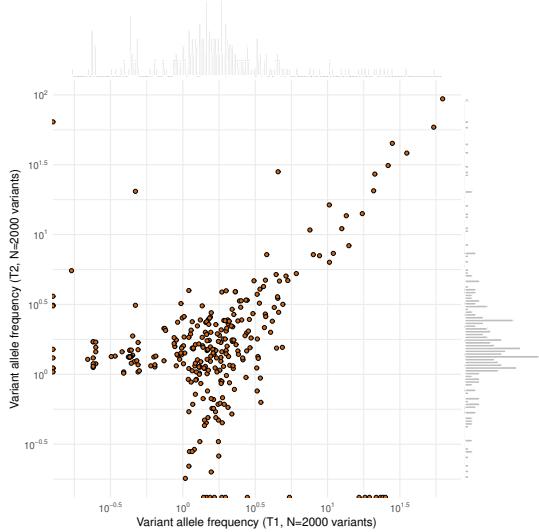
plot.af(df = varscan_af, pval = 1)
```



586
587 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
588

```
plot.af.logged(df = varscan_af, pval = 1)
```

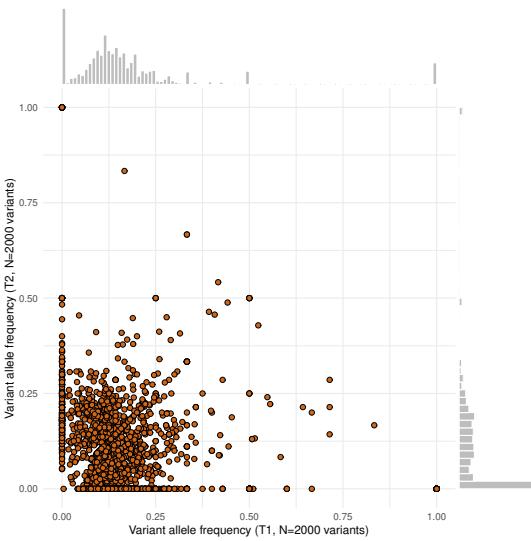
Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 9 rows containing non-finite values (stat_bin).
 Warning: Removed 15 rows containing non-finite values (stat_bin).



589
590 Bcftools was used to call for variants against the hg19 GRCh37 human genome.

```
bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcftools.snpeff_annot_genome.c  

plot.baf(df = bcftools_af)
```

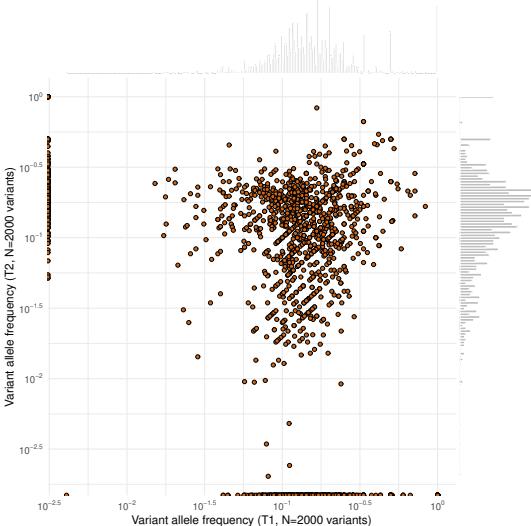


591

Logarithmic

```
plot.baf.logged(df = bcftools_af)
```

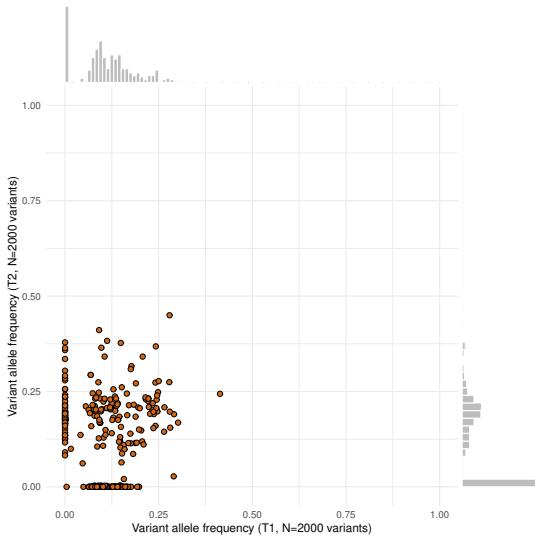
Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 199 rows containing non-finite values (stat_bin).
 Warning: Removed 514 rows containing non-finite values (stat_bin).



593

594 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

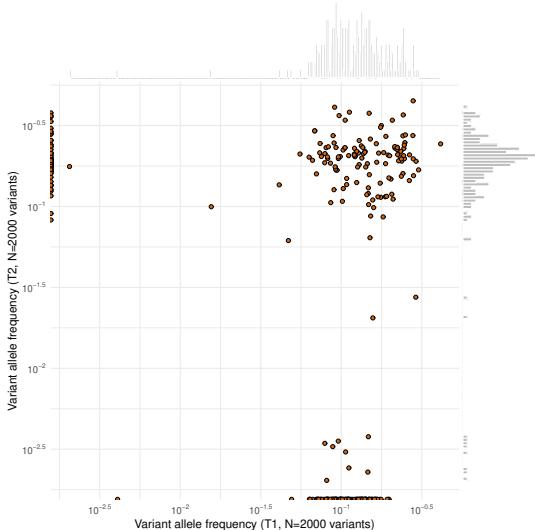
```
bcftools_af <- read.table("./data/pyclone/stringent/output.4_mined.bcftools_snpeff_annot_genome.clinvar")
plot.baf(df = bcftools_af)
```



595
596 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
597

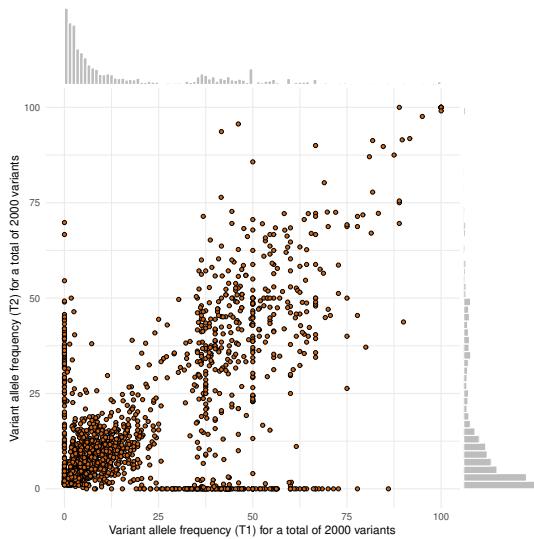
```
plot.baf.logged(df = bcftools_af)
```

Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 53 rows containing non-finite values (stat_bin).
 Warning: Removed 112 rows containing non-finite values (stat_bin).



598
599 **5.2.4 Individual 4**
600 Varscan was used to call for variants against the hg19 GRCh37 human genome.

```
varscan_af <- read.table("./data/pyclone/reduced_variants/output.4_mined.varscan2.snpeff_annot_genome.c")
plot.af(df = varscan_af, pval = 1)
```

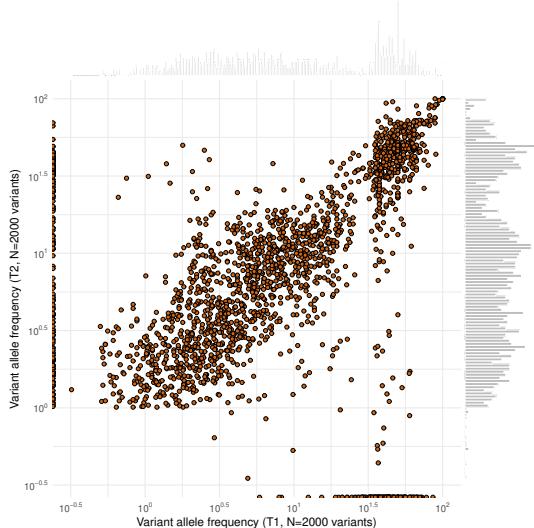


601
602

Logarithmic

```
plot.af.logged(df = varscan_af, pval = 1)
```

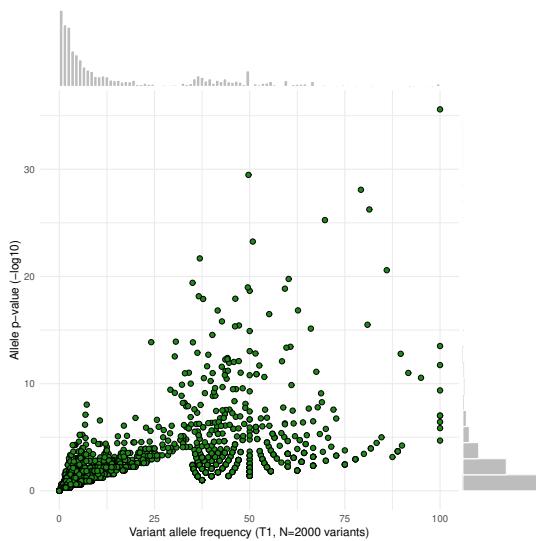
Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 182 rows containing non-finite values (stat_bin).
 Warning: Removed 146 rows containing non-finite values (stat_bin).



603
604

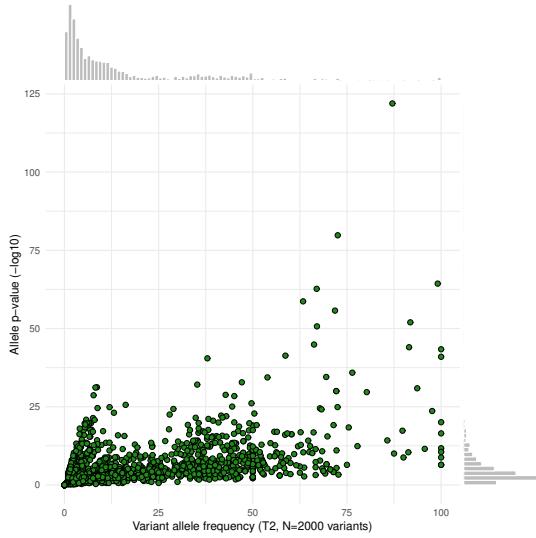
Allele frequency versus p-value of called variants. For each timepoint.

```
plot.pval.t1(df = varscan_af)
```



605
606 Allele frequency versus p-value of called variants. For each timepoint.

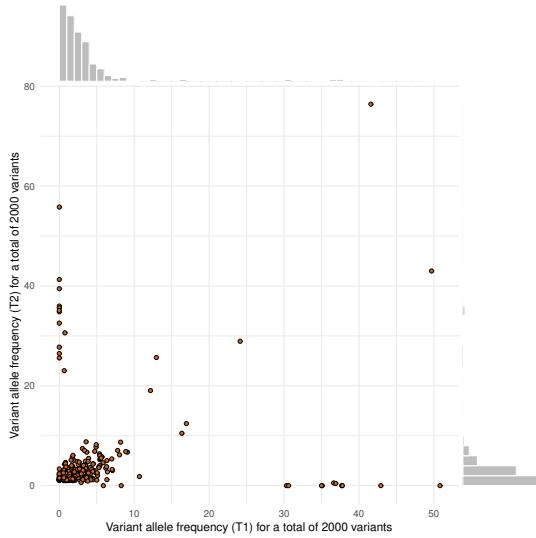
```
plot.pval.t2(df = varscan_af)
```



607
608 Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
varscan_af <- read.table("./data/pyclone/stringent/output.4_mined.varscan2.snpeff_annot_genome.clinvar.
```

```
plot.af(df = varscan_af, pval = 1)
```



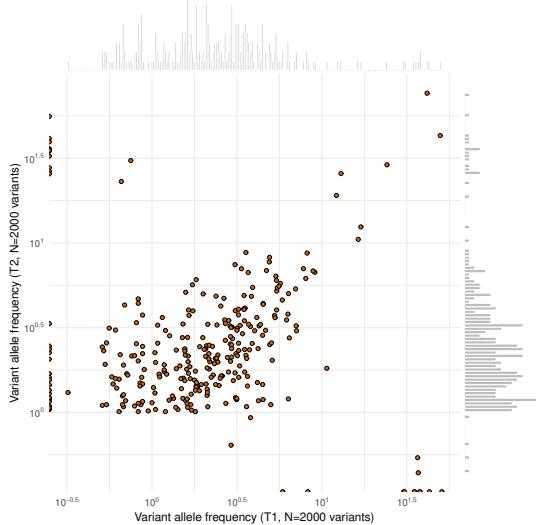
609

610 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

611

```
plot.af.logged(df = varscan_af, pval = 1)
```

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 30 rows containing non-finite values (stat_bin).
Warning: Removed 10 rows containing non-finite values (stat_bin).

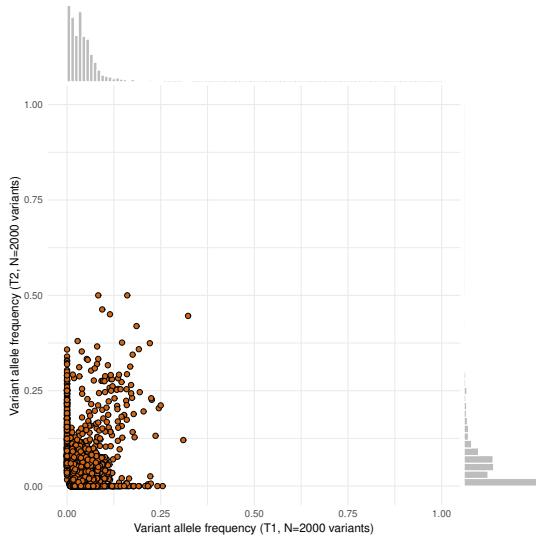


612

613 Bcftools was used to call for variants against the hg19 GRCh37 human genome.

```
bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcftools.snpeff_annot_genome.c
```

```
plot.baf(df = bcftools_af)
```

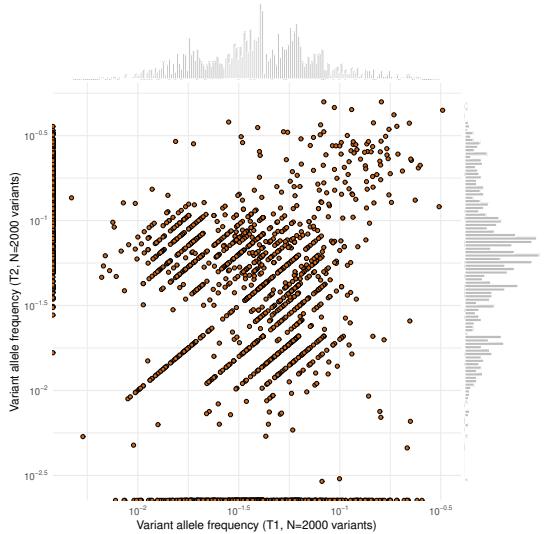


614

615 Logarithmic

```
plot.baf.logged(df = bcftools_af)
```

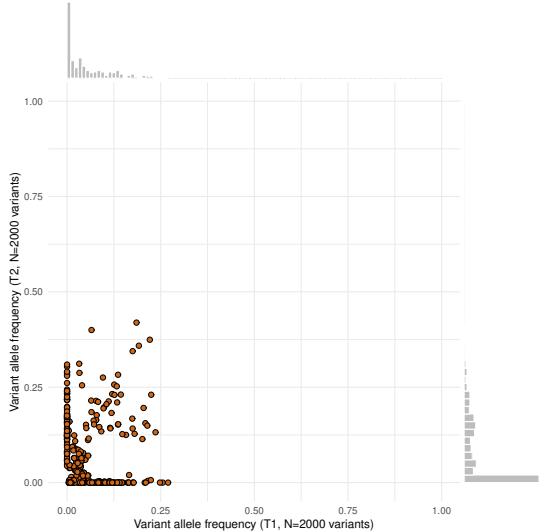
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 329 rows containing non-finite values (stat_bin).
Warning: Removed 489 rows containing non-finite values (stat_bin).



616
617

Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
bcftools_af <- read.table("./data/pyclone/stringent/output.4_mined.bcftools.snpeff_annot_genome.clinvar")
plot.baf(df = bcftools_af)
```

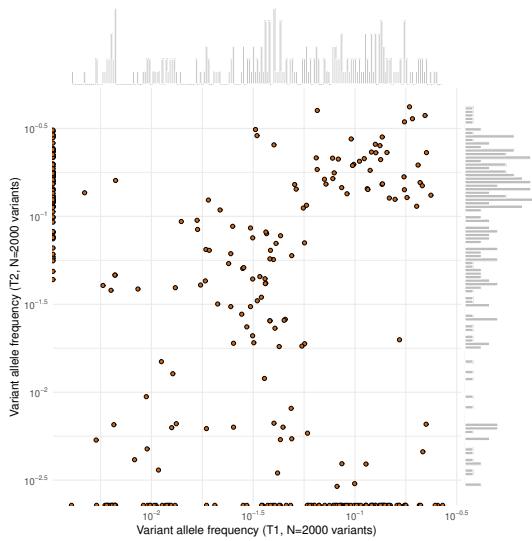


618
619
620

(Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

```
plot.baf.logged(df = bcftools_af)

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous y-axis
Warning: Removed 78 rows containing non-finite values (stat_bin).
Warning: Removed 97 rows containing non-finite values (stat_bin).
```

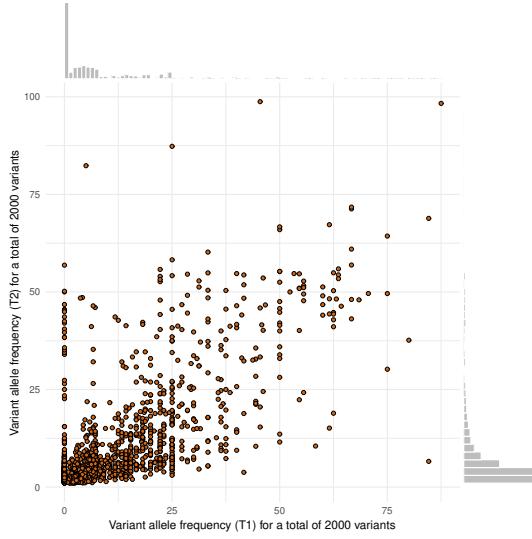


621

5.2.5 Individual 5

Varscan was used to call for variants against the hg19 GRCh37 human genome.

```
varscan_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.varscan2.snpEff_annot_genome.cl")
plot.af(df = varscan_af, pval = 1)
```

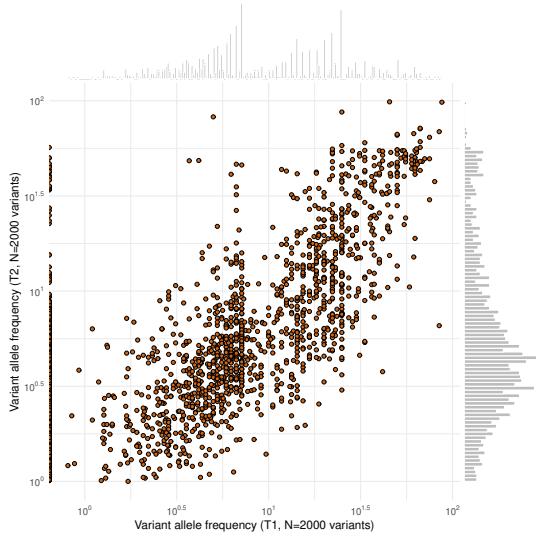


624

Logarithmic

```
plot.af.logged(df = varscan_af, pval = 1)
```

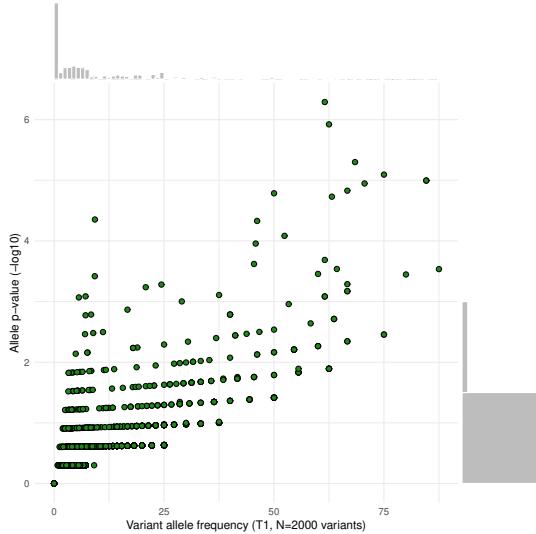
```
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Removed 677 rows containing non-finite values (stat_bin).
```



626
627

Allele frequency versus p-value of called variants. For each timepoint.

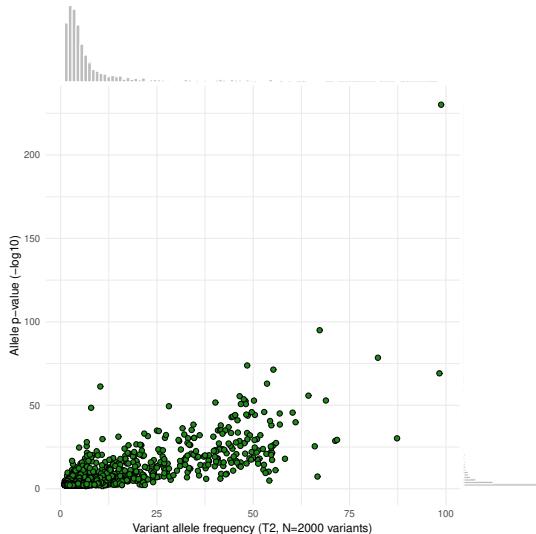
```
plot.pval.t1(df = varscan_af)
```



628
629

Allele frequency versus p-value of called variants. For each timepoint.

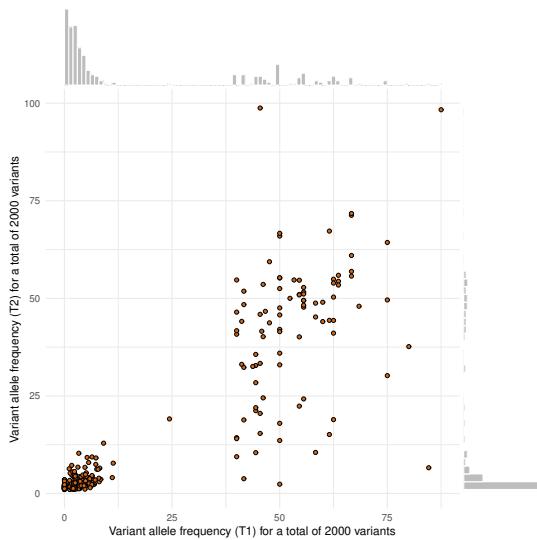
```
plot.pval.t2(df = varscan_af)
```



630
631

Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

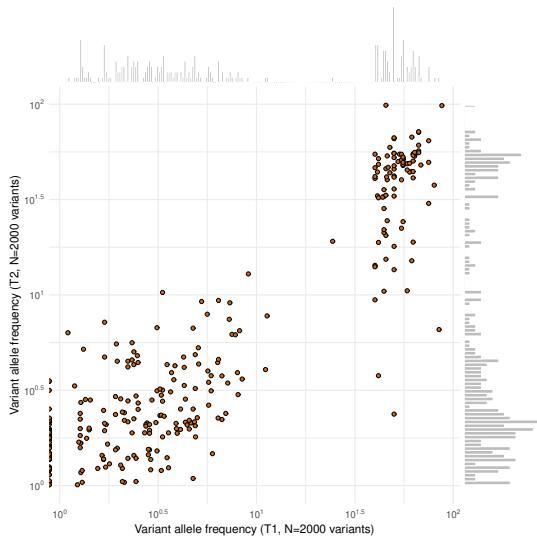
```
varscan_af <- read.table("./data/pyclone/stringent/output.4_mined.varscan2.snpeff_annot_genome.clinvar.c")  
plot.af(df = varscan_af, pval = 1)
```



632
633 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
634

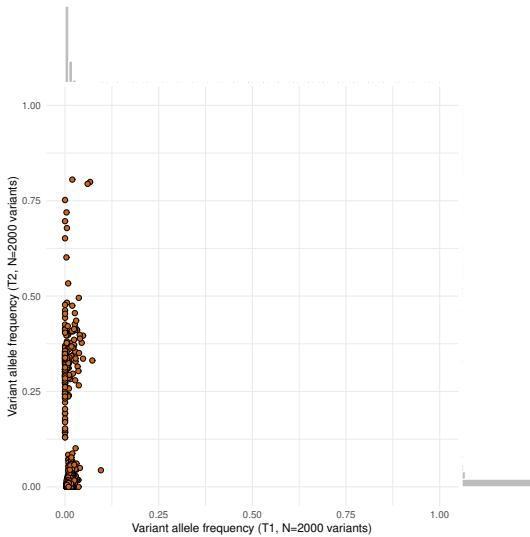
```
plot.af.logged(df = varscan_af, pval = 1)
```

Warning: Transformation introduced infinite values in continuous x-axis
Warning: Transformation introduced infinite values in continuous x-axis
Warning: Removed 51 rows containing non-finite values (stat_bin).



635
636 Bcftools was used to call for variants against the hg19 GRCh37 human genome.

```
bcftools_af<- read.table("./data/pyclone/reduced_variants/output.4_mined.bcfTools.snpeff_annot_genome.c")  
plot.baf(df = bcftools_af)
```

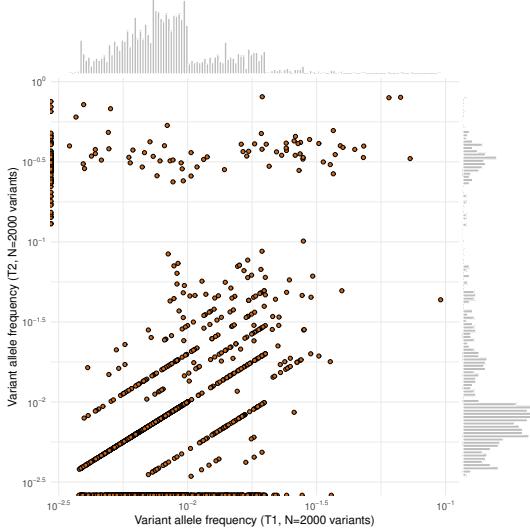


637
638

Logarithmic

```
plot.baf.logged(df = bcftools_af)
```

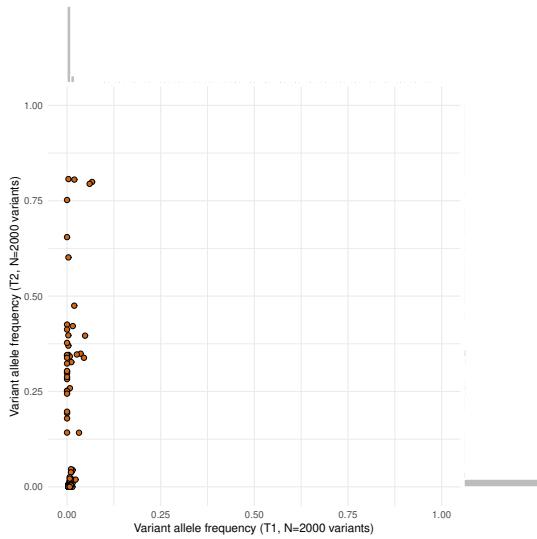
Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 95 rows containing non-finite values (stat_bin).
 Warning: Removed 964 rows containing non-finite values (stat_bin).



639
640

Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.

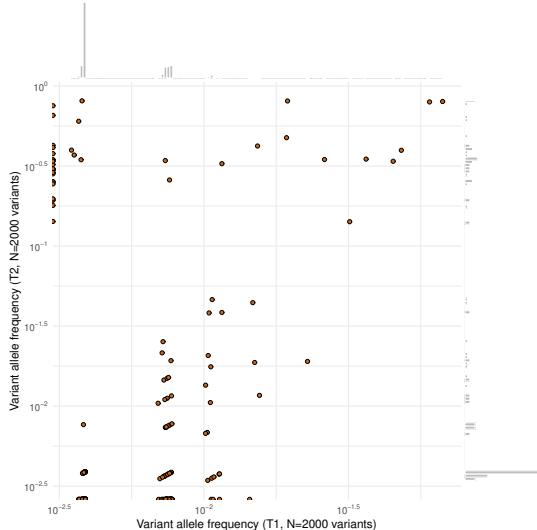
```
bcftools_af <- read.table("./data/pyclone/stringent/output.4_mined.bcftools_snpeff_annot_genome.clinvar"
plot.baf(df = bcftools_af)
```



641
642 (Logarithmic) Stringency increased to eliminate above 85% of variants, keeping a total of 300 sites called.
643

```
plot.baf.logged(df = bcftools_af)
```

Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Transformation introduced infinite values in continuous x-axis
 Warning: Transformation introduced infinite values in continuous y-axis
 Warning: Removed 21 rows containing non-finite values (stat_bin).
 Warning: Removed 205 rows containing non-finite values (stat_bin).



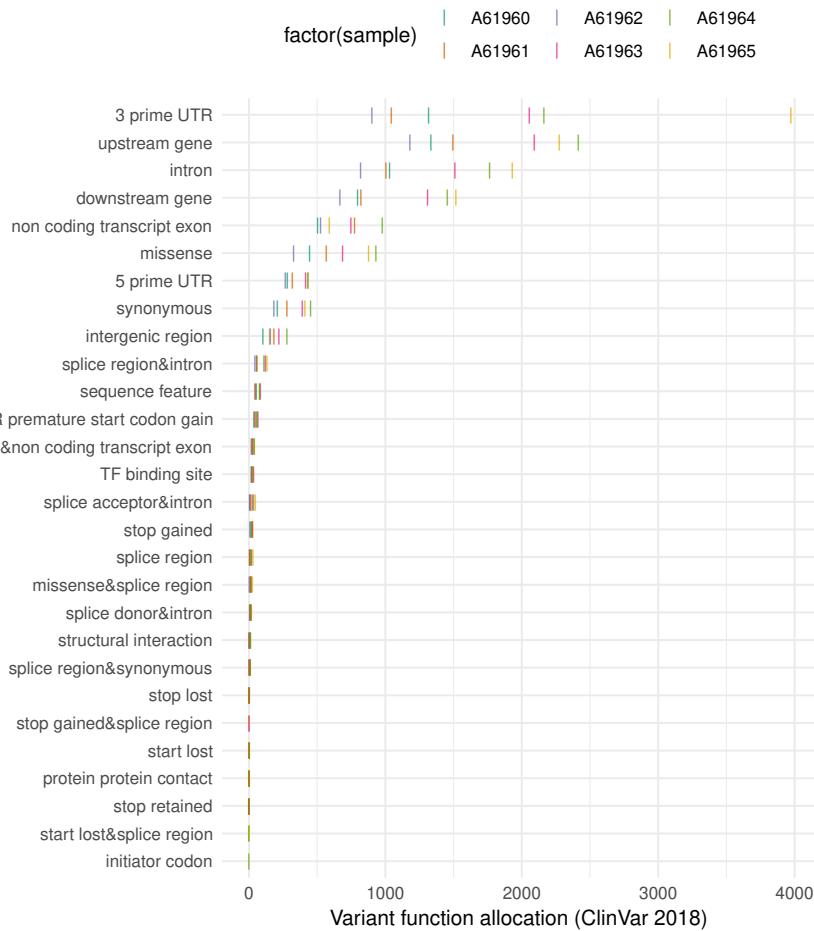
644
645 **5.3 Location of called variants**
646 Variants and their distribution based on their known locations obtained from annotations with different
647 data banks, dbSnp, clinVar, and exac.

```
read.table("./data/output.8_bash.summary.snv_functions.txt", header = TRUE) %>%
```

```

  mutate(fct = gsub("_variant", "", fct)) %>%
  mutate(fct = gsub("_", " ", fct)) %>%
  ggplot(aes(x = reorder(fct, count),
             y = count,
             color = factor(sample))) +
  geom_point(size = 3, shape = "|") +
  coord_flip() +
  scale_color_brewer(palette="Dark2") +
  theme_minimal() +
  theme(legend.position = "top") +
  xlab("") +
  ylab("Variant function allocation (ClinVar 2018)")

```



648

5.4 Cellular prevalence and clustering of tumor clones

649

PyClone was loaded with metrics that relay the number of reads specific for each allele (allele depth for reference and alternative alleles), copy numbers (ie., normal, minor, major counts, eg., 2-0-2), and position of each called variant. The output is then inferred from these metrics and cellular prevalence, clonal clusters, and allele frequency are then reported. Seed of 124565 was used with disconnected density clustering at 10,000 MCMC iterations with a major copy number prior and binomial distribution analysis.

650

5.4.1 Variant allele frequency by individual using Varscan2

651

Number and allocation of called single point mutations between all three exome sequenced individuals.

Distribution below include the position of the SNV, the chromosome, and the name of the gene.

```
ind.1 <- read.table("./data/pyclone/stringent/loci.ind1.562853.txt", header = T) %>%
```

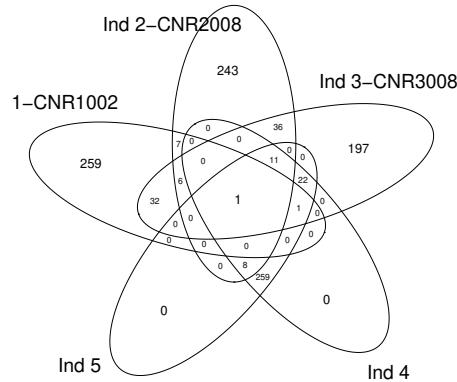
```

    mutate(sample_id = gsub("^.*ind","ind", sample_id))
ind.2 <- read.table("./data/pyclone/stringent/loci.ind2.563282.txt", header = T) %>%
    mutate(sample_id = gsub("^.*ind","ind", sample_id))
ind.3 <- read.table("./data/pyclone/stringent/loci.ind3.563283.txt", header = T) %>%
    mutate(sample_id = gsub("^.*ind","ind", sample_id))
ind.4 <- read.table("./data/pyclone/stringent/loci.ind4.563284.txt", header = T) %>%
    mutate(sample_id = gsub("^.*ind","ind", sample_id))
ind.5 <- read.table("./data/pyclone/stringent/loci.ind5.563285.txt", header = T) %>%
    mutate(sample_id = gsub("^.*ind","ind", sample_id))

ind.list <- list(unique(ind.1$mutation_id), unique(ind.2$mutation_id),
                 unique(ind.3$mutation_id), unique(ind.4$mutation_id),
                 unique(ind.5$mutation_id))
names(ind.list) <- c("Ind 1-CNR1002",
                     "Ind 2-CNR2008",
                     "Ind 3-CNR3008",
                     "Ind 4",
                     "Ind 5")

venn(ind.list)

```



659
660 Variant allele frequency (VAF) for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2
661 (T2).

```
plot_frequencies <- function(df, t1 = 0.9, t2 = 0.9) {
```

```

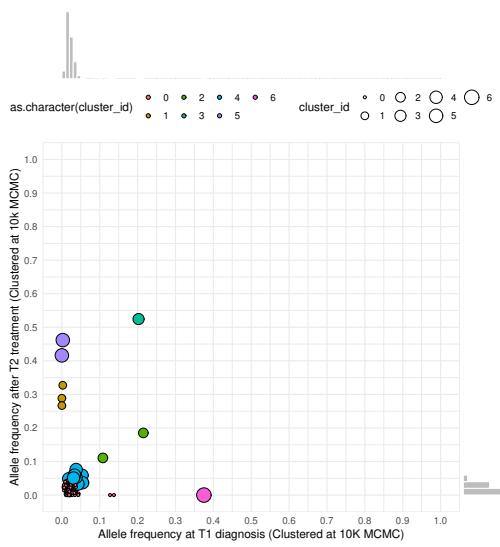
subdf <- df %>%
  select(mutation_id, cluster_id, sample_id, variant_allele_frequency) %>%
  spread(sample_id, variant_allele_frequency)

p <- subdf %>%
  ggplot(aes(x = subdf[, 3],
             y = subdf[, 4],
             label = mutation_id,
             fill = as.character(cluster_id))) +
  geom_point(aes(size = cluster_id), shape = 21) +
  theme_minimal() +
  theme(legend.position = "top") +
  expand_limits(x = c(0,1), y = c(0,1)) +
  scale_y_continuous(breaks = pretty(c(0,1), n = 10)) +
  scale_x_continuous(breaks = pretty(c(0,1), n = 10)) +
  xlab(paste0("Allele frequency at T1 diagnosis (Clustered at 10K MCMC)")) +
  ylab(paste0("Allele frequency after T2 treatment (Clustered at 10k MCMC)"))
ggMarginal(p, type = "histogram", fill="transparent",
            color = "white",
            xparams = list(binwidth = .01, fill = "grey"),
            yparams = list(binwidth = .02, fill = "grey"))

}

plot_frequencies(df = ind.1)

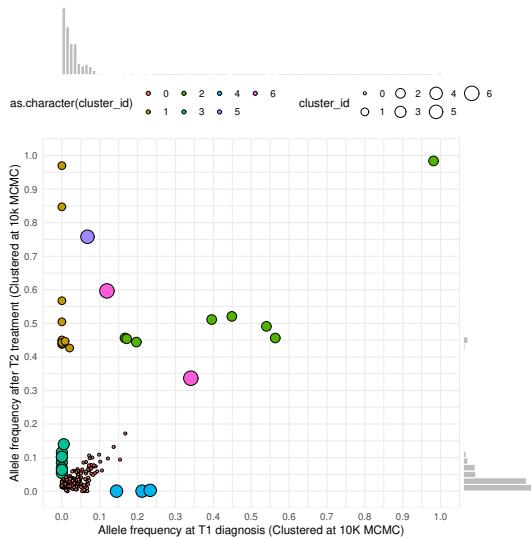
```



662

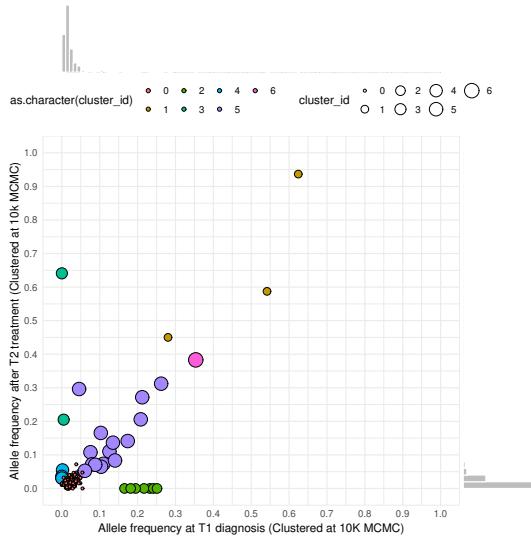
663 VAF for the individual anonymously labeled **CNR2008** for Time 1 (T1) and 2 (T2).

```
plot_frequencies(df = ind.2)
```



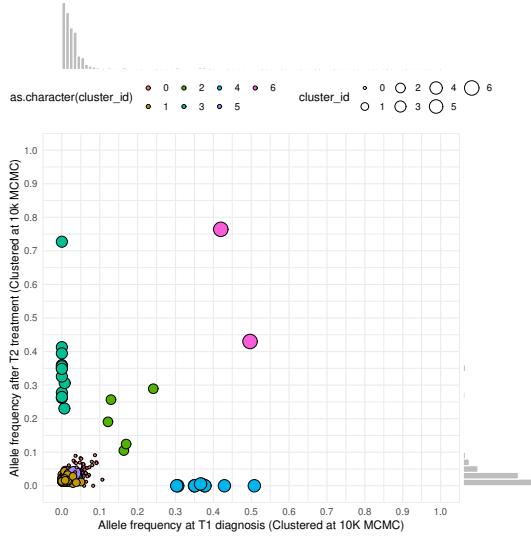
664
665 VAF for the individual anonymously labeled **CNR3008** for Time 1 (T1) and 2 (T2).

```
plot_frequencies(df = ind.3)
```



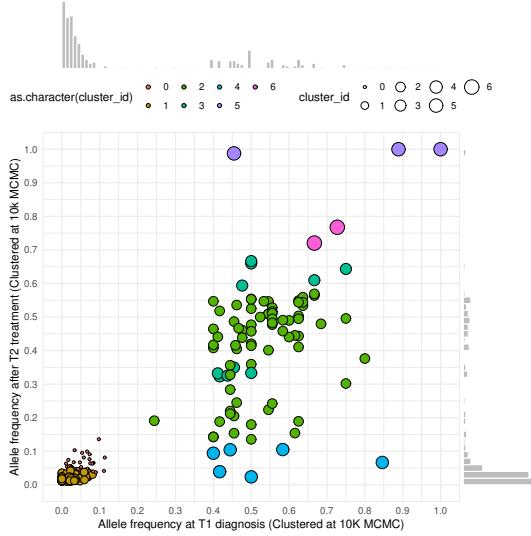
666
667 VAF for the individual **LY_CNSrel_001** for Time 1 (T1) and 2 (T2).

```
plot_frequencies(df = ind.4)
```



668
669 VAF for the individual **LY_CNSrel_003** for Time 1 (T1) and 2 (T2).

```
plot_frequencies(df = ind.5)
```



670

5.4.2 Cellular prevalence by individual using Varscan2

671 Cellular prevalence for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2 (T2).

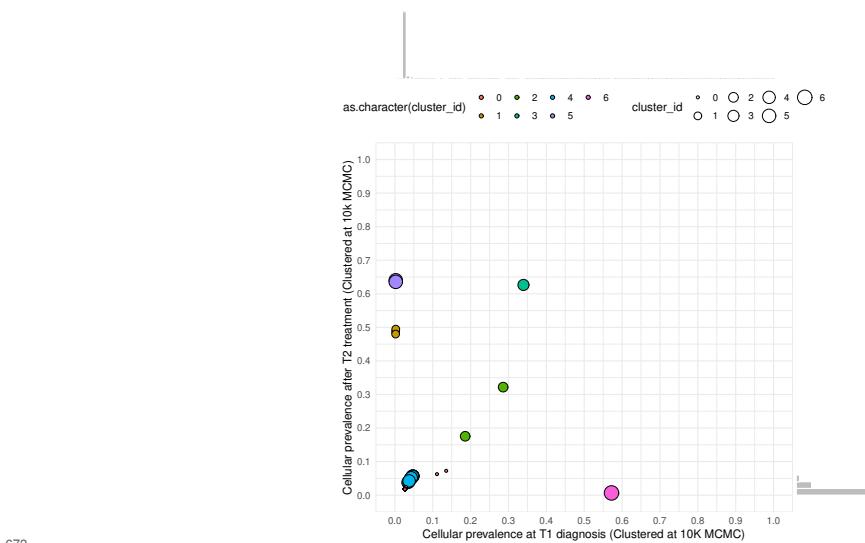
[†]Prevalence and VAF were generated by pyclone

```
plot_prevalence <- function(df, t1 = 0.9, t2 = 0.9) {
  subdf <- df %>%
    select(mutation_id, cluster_id, sample_id, cellular_prevalence) %>%
    spread(sample_id, cellular_prevalence)

  p <- subdf %>%
    ggplot(aes(x = subdf[,3],
               y = subdf[,4],
               label = mutation_id,
               fill = as.character(cluster_id))) +
    geom_point(aes(size = cluster_id), shape = 21) +
    theme_minimal() +
    theme(legend.position = "top") +
    expand_limits(x = c(0,1), y = c(0,1)) +
    scale_y_continuous(breaks = pretty(c(0,1), n = 10)) +
    scale_x_continuous(breaks = pretty(c(0,1), n = 10)) +
    xlab(paste0("Cellular prevalence at T1 diagnosis (Clustered at 10K MCMC)")) +
    ylab(paste0("Cellular prevalence after T2 treatment (Clustered at 10k MCMC)"))

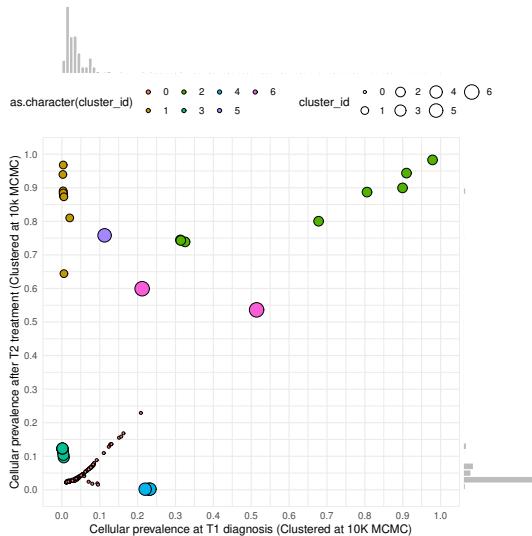
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = .01, fill = "grey"),
             yparams = list(binwidth = .02, fill = "grey"))
}

plot_prevalence(df = ind.1)
```



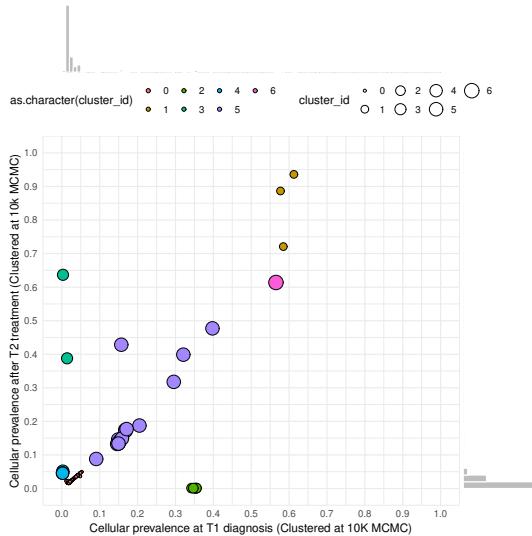
673
674 Cellular prevalence for the individual anonymously labeled **CNR2008** for Time 1 (T1) and 2 (T2).

```
plot_prevalence(df = ind.2)
```



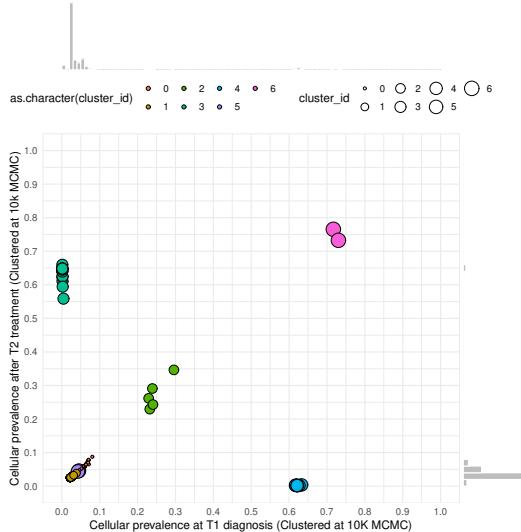
675
676 Cellular prevalence for the individual anonymously labeled **CNR3008** for Time 1 (T1) and 2 (T2).

```
plot_prevalence(df = ind.3)
```



677
678 Cellular prevalence for the individual anonymously labeled **LY_CNSrel_001** for Time 1 (T1) and 2 (T2).

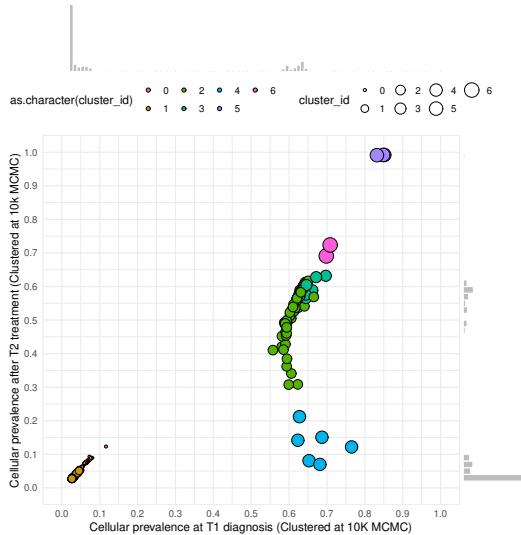
```
plot_prevalence(df = ind.4)
```



679

680 Cellular prevalence for the individual anonymously labeled LY_CNSrel_003 for Time 1 (T1) and 2 (T2).

```
plot_prevalence(df = ind.5)
```



681

5.4.3 Variant allele frequency by individual using Bcftools

682 Number and allocation of called single point mutations between all three exome sequenced individuals.

683 Distribution below include the position of the SNV, the chromosome, and the name of the gene.

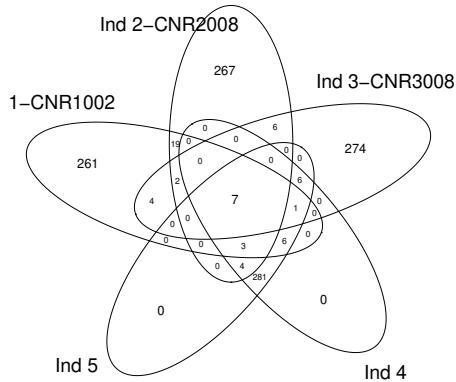
```
## without functions
```

```

ind.1 <- read.table("./data/pyclone/stringent/loci.ind1.563505.bcfTools.txt",
  mutate(sample_id = gsub("^.*ind", "ind", sample_id))
ind.2 <- read.table("./data/pyclone/stringent/loci.ind2.563506.bcfTools.txt",
  mutate(sample_id = gsub("^.*ind", "ind", sample_id))
ind.3 <- read.table("./data/pyclone/stringent/loci.ind3.563507.bcfTools.txt",
  mutate(sample_id = gsub("^.*ind", "ind", sample_id))
ind.4 <- read.table("./data/pyclone/stringent/loci.ind4.563508.bcfTools.txt",
  mutate(sample_id = gsub("^.*ind", "ind", sample_id))
ind.5 <- read.table("./data/pyclone/stringent/loci.ind5.563509.bcfTools.txt",
  mutate(sample_id = gsub("^.*ind", "ind", sample_id))

ind.list <- list(unique(ind.1$mutation_id), unique(ind.2$mutation_id),
  unique(ind.3$mutation_id), unique(ind.4$mutation_id),
  unique(ind.5$mutation_id))
names(ind.list) <- c("Ind 1-CNR1002",
  "Ind 2-CNR2008",
  "Ind 3-CNR3008",
  "Ind 4",
  "Ind 5")
venn(ind.list)

```



685
686 Variant allele frequency (VAF) for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2
687 (T2).

```
plot_frequency <- function(df, t1 = 0.9, t2 = 0.9) {
```

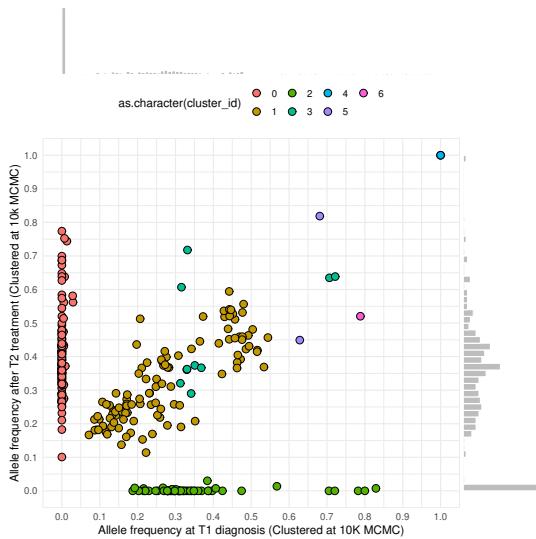
```

subdf <- df %>%
  select(mutation_id, cluster_id, sample_id, variant_allele_frequency) %>%
  spread(sample_id, variant_allele_frequency)
#   filter(cluster_id > 1)

p <- subdf %>%
  ggplot(aes(x = subdf[, 3],
             y = subdf[, 4],
             label = mutation_id,
             fill = as.character(cluster_id))) +
  geom_point(shape = 21, size = 3) +
  ## geom_text_repel(data = subdf,
  ##                   nudge_x = -0.1,
  ##                   nudge_y = -0.1,
  ##                   force = 5,
  ##                   segment.color = "grey50",
  ##                   direction = "y",
  ##                   segment.size = 0.1,
  ##                   size = 2) +
  theme_minimal() +
  theme(legend.position = "top") +
  expand_limits(x = c(0,1), y = c(0,1)) +
  scale_y_continuous(breaks = pretty(c(0,1), n = 10)) +
  scale_x_continuous(breaks = pretty(c(0,1), n = 10)) +
  xlab(paste0("Allele frequency at T1 diagnosis (Clustered at 10K MCMC)")) +
  ylab(paste0("Allele frequency after T2 treatment (Clustered at 10k MCMC)"))
  ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = .01, fill = "grey"),
             yparams = list(binwidth = .02, fill = "grey"))
}

plot_frequency(df = ind.1)
Warning: Removed 1 rows containing missing values (geom_point).

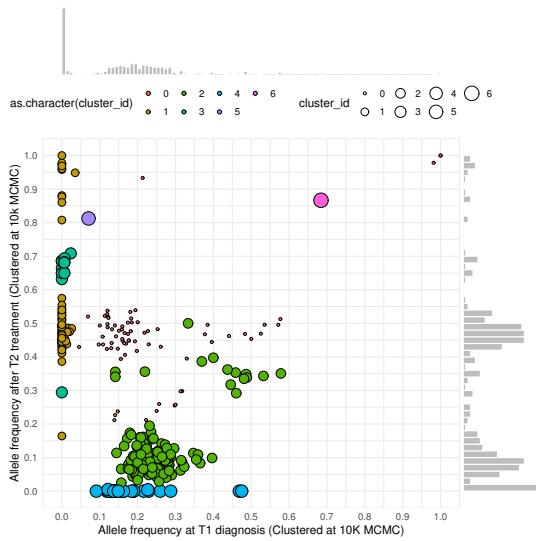
```



688

689 VAF for the individual anonymously labeled **CNR2008** for Time 1 (T1) and 2 (T2).

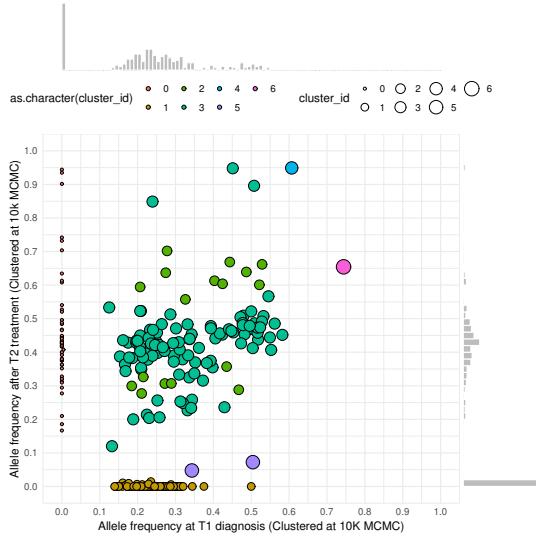
```
plot_frequencies(df = ind.2)
```



690
691

VAF for the individual anonymously labeled **CNR3008** for Time 1 (T1) and 2 (T2).

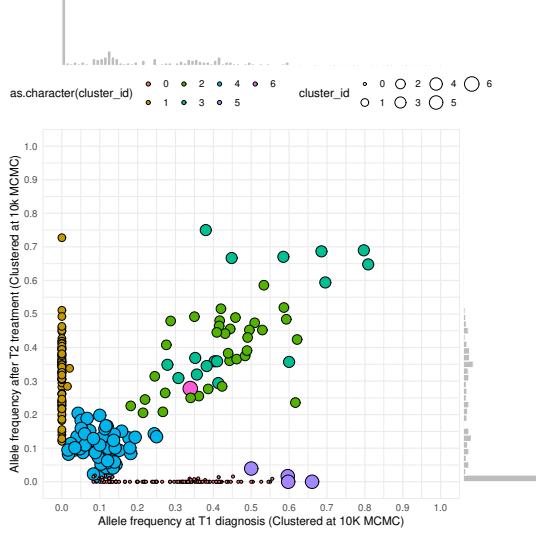
`plot_frequencies(ind.3)`



692
693

VAF for the individual anonymously labeled **LY_CNSrel_001** for Time 1 (T1) and 2 (T2).

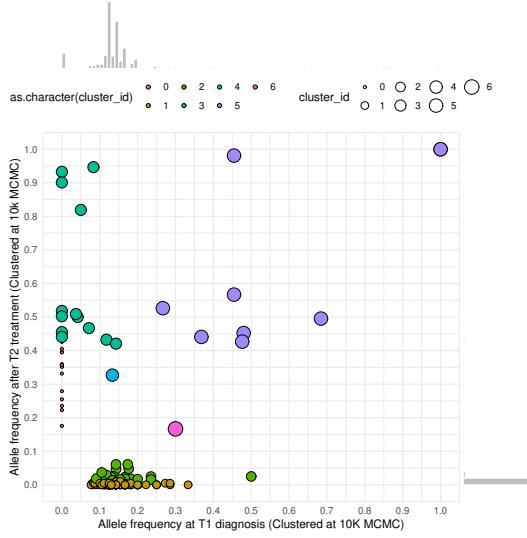
`plot_frequencies(ind.4)`



694
695

VAF for the individual anonymously labeled **LY_CNSrel_003** for Time 1 (T1) and 2 (T2).

```
plot_frequencies(ind.5)
```



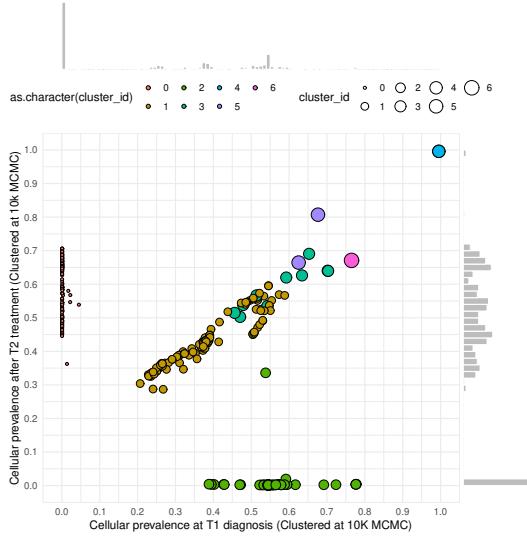
696

5.4.4 Cellular prevalence by individual using Bcftools

697 Cellular prevalence for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2 (T2).

[†]Prevalence and VAF were generated by pyclone

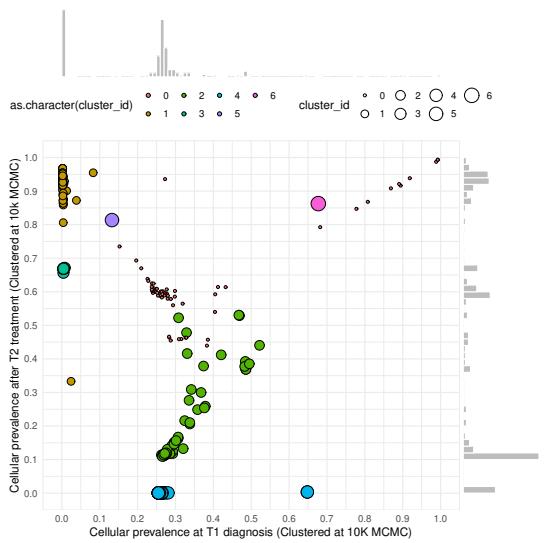
```
plot_prevalence(ind.1)
```



699

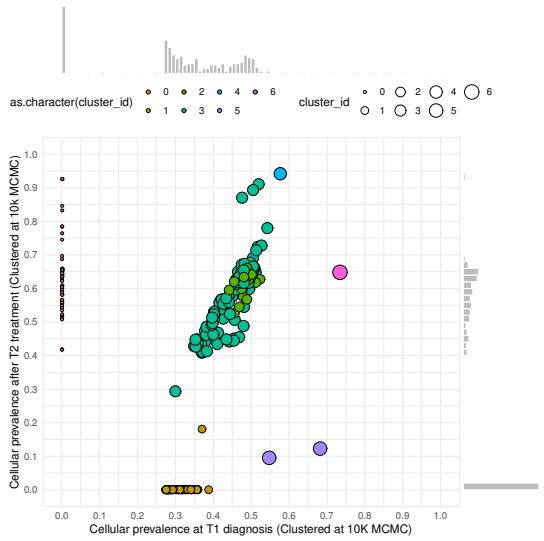
700 Cellular prevalence for the individual anonymously labeled **CNR2008** for Time 1 (T1) and 2 (T2).

```
plot_prevalence(ind.2)
```



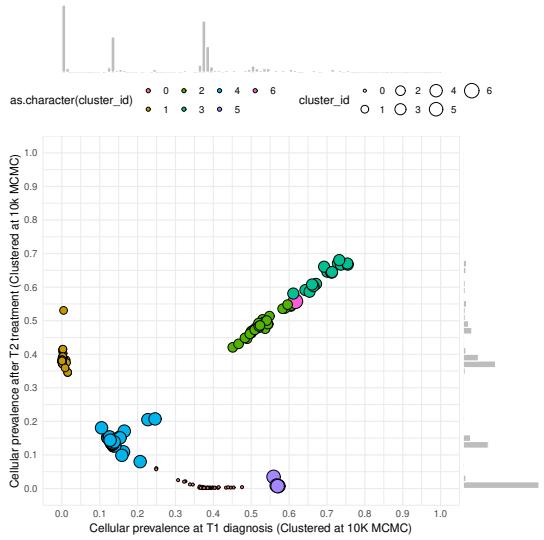
701
702 Cellular prevalence for the individual anonymously labeled **CNR3008** for Time 1 (T1) and 2 (T2).

`plot_prevalence(ind.3)`



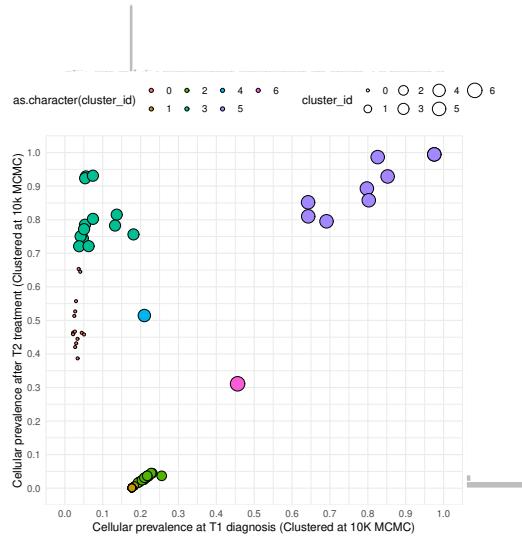
703
704 Cellular prevalence for the individual anonymously labeled **LY_CNSrel_001** for Time 1 (T1) and 2 (T2).

`plot_prevalence(ind.4)`



705
706 Cellular prevalence for the individual anonymously labeled **LY_CNSrel_003** for Time 1 (T1) and 2 (T2).

```
plot_prevalence(ind.5)
```



707

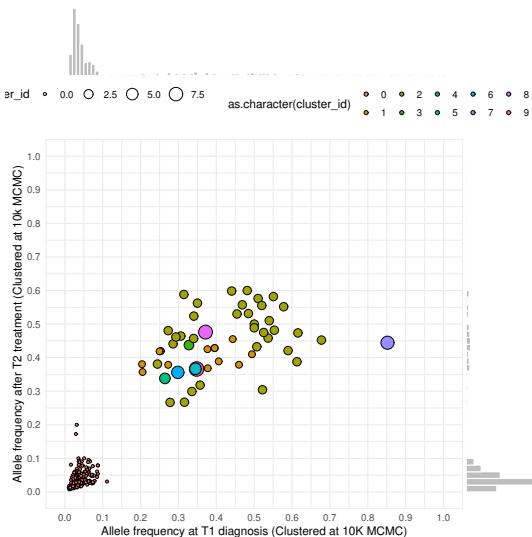
5.4.5 Variant allele frequency by individual using GATK

Number and allocation of called single point mutations between all three exome sequenced individuals. [↑ OLD!! Deprecated](#)
709 Distribution below include the position of the SNV, the chromosome, and the name of the gene.
710

```
ind.1 <- read.table("./data/pyclone/loci.520917.gatk.tsv", header = T) %>%
  mutate(sample_id = gsub("^.A61", "A61", sample_id))
```

711 Variant allele frequency (VAF) for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2
712 (T2).

```
plot_frequencies(ind.1)
```



713

5.4.6 Cellular prevalence by individual using GATK

714 Cellular prevalence for the individual anonymously labeled **CNR1002** for Time 1 (T1) and 2 (T2).
715

[↑Prevalence and VAF were generated by pyclone](#)

```
model.names <- ind.1 %>%
```

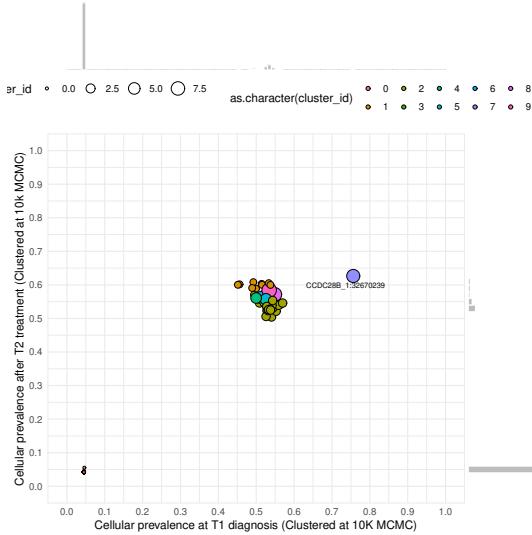
```

  select(mutation_id, cluster_id, sample_id, cellular_prevalence) %>%
  spread(sample_id, cellular_prevalence)

t1 = 0.7
t2 = 0.6

p <- ind.1 %>%
  select(mutation_id, cluster_id, sample_id, cellular_prevalence) %>%
  spread(sample_id, cellular_prevalence) %>%
  ggplot(aes(x = A61960,
               y = A61961,
               label = mutation_id,
               fill = as.character(cluster_id))) +
  geom_point(aes(size = cluster_id), shape = 21) +
  geom_text_repel(data = subset(model.names, A61961 >= t2 & A61960 >= t1),
                    nudge_x = -.02,
                    force = 5,
                    segment.color = "grey50",
                    direction = "y",
                    segment.size = 0.1,
                    size = 2.5) +
  theme_minimal() +
  theme(legend.position = "top") +
  expand_limits(x = c(0,1), y = c(0,1)) +
  scale_y_continuous(breaks = pretty(c(0,1), n = 10)) +
  scale_x_continuous(breaks = pretty(c(0,1), n = 10)) +
  xlab(paste0("Cellular prevalence at T1 diagnosis (Clustered at 10K MCMC)")) +
  ylab(paste0("Cellular prevalence after T2 treatment (Clustered at 10k MCMC)"))
ggMarginal(p, type = "histogram", fill="transparent",
             color = "white",
             xparams = list(binwidth = .01, fill = "grey"),
             yparams = list(binwidth = .02, fill = "grey"))

```



716

6 System information for this report

The version number of R and packages loaded for generating the vignette were:

```
## ## save(list=ls(pattern=".*/.*"), file="PD.Rdata")
```

```

sessionInfo()

R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: elementary OS 0.4.1 Loki

Matrix products: default
BLAS: /usr/lib/libblas/libblas.so.3.6.0
LAPACK: /usr/lib/lapack/liblapack.so.3.6.0

locale:
[1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8          LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8      LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8         LC_NAME=C
[9] LC_ADDRESS=C                 LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8   LC_IDENTIFICATION=C

attached base packages:
[1] stats      graphics    grDevices utils      datasets   methods
[7] base

other attached packages:
[1] bindrcpp_0.2.2    reshape_0.8.7    ggridges_0.5.0
[4] cowplot_0.9.3     ggpubr_0.1.7     magrittr_1.5
[7] ggExtra_0.8       ggrepel_0.8.0    paletteer_0.1.0
[10] plyr_1.8.4        finalfit_0.7.4   Hmisc_4.1-1
[13] Formula_1.2-3    survival_2.42-6  brotools_0.2
[16] scales_1.0.0      DescTools_0.99.23  igraph_1.1.2
[19] tidyverse_0.8.0    dplyr_0.7.6      ggplot2_3.0.0
[22] gplots_3.0.1      latticeExtra_0.6-28 RColorBrewer_1.1-2
[25] lattice_0.20-35   gdata_2.18.0     knitr_1.20

loaded via a namespace (and not attached):
[1] splines_3.4.4      gtools_3.5.0      shiny_1.0.5
[4] assertthat_0.2.0    expm_0.999-2     highr_0.6
[7] pillar_1.1.0       backports_1.1.1   glue_1.3.0
[10] digest_0.6.18     checkmate_1.8.5   colorspace_1.3-2
[13] htmltools_0.3.6    httpuv_1.3.5     Matrix_1.2-11
[16] pkgconfig_2.0.2    purrr_0.2.4      xtable_1.8-2
[19] mvtnorm_1.0-7     manipulate_1.0.1  htmlTable_1.11.2
[22] tibble_1.4.2       withr_2.1.2      nnet_7.3-12
[25] lazyeval_0.2.1    mime_0.5       evaluate_0.12
[28] MASS_7.3-47       foreign_0.8-70   tools_3.4.4
[31] data.table_1.11.6  stringr_1.3.1   munsell_0.5.0
[34] cluster_2.0.7-1   compiler_3.4.4   caTools_1.17.1
[37] rlang_0.2.2       grid_3.4.4     rstudioapi_0.7
[40] htmlwidgets_1.2    miniUI_0.1.1.1  labeling_0.3
[43] bitops_1.0-6      base64enc_0.1-3  boot_1.3-20
[46] gtable_0.2.0      reshape2_1.4.3   R6_2.2.2
[49] gridExtra_2.3     bindr_0.1.1     KernSmooth_2.23-15
[52] stringi_1.2.4    Rcpp_0.12.19   rpart_4.1-13
[55] acepack_1.4.1     tidyselect_0.2.4
```