

Creation of active gene-lists

Lucas Michel Todó

31/03/2020

Contents

1 Active/Inactive Gene lists	2
1.1 Microarray Data: Red Signal	2
1.2 Microarray Data: Areas	2
1.3 Load RNA-Seq Data	4
1.4 Create table for classification	6
1.5 Create Lists according to thresholds	7
1.6 Some results	9
1.7 Venn Diagrams	10
2 Code	13
2.1 Load Packages and functions	13
2.2 Microarray Data: Red Signal	14
2.3 Microarray Data: Areas	15
2.4 Load RNA-Seq Data	18
2.5 Load Annotation	20
2.6 Create Join DF	20
2.7 Create Lists according to thresholds	22
2.8 Some checks and results	25
2.9 Comparison of 12B vs 10G differences	26
2.10 Venn Diagram of results	27

1 Active/Inactive Gene lists

Our aim is to create a unified table that assigns to each gene in the P.falciparum gnome a expresion state. We will define 5 possible expresion states:

- Active
 - Active (no-variant genes)
 - Variant Active
 - Variant Repressed
- Inactive (no-variant genes)
- Not-Settable

1.1 Microarray Data: Red Signal

We willl load the red signal and transform it into percentiles. For each gene we pick the "Aver.2Higher" column from the original microarrays data table. This column corresponds to the average between the two highest red signals among available timepoints.

Red Signal DataFrame

	Gene_id	Red_12B	Red_10G	Red_3D7B	Percent_12B	Percent_10G
1	mal_mito_3	22579.33333	36436.73333	30636.8250	96.0466005	98.4721161
2	PF3D7_0100100	104.17083	215.03542	264.4000	5.6913675	10.9625668
3	PF3D7_0100200	1357.70000	72.67917	724.1500	29.7555386	4.0106952
4	PF3D7_0100300	283.98333	291.69583	7540.9042	12.5668449	12.9870130
5	PF3D7_0100400	193.77500	35.01250	161.0833	9.8930481	1.0504202
6	PF3D7_0100600	31.58333	33.40417	31.5250	0.2291826	0.4392666
		Percent_3D7B	MaxRedPercentDiff	MeanRedPercent		
1	97.8800611	2.425516	97.4662592			
2	11.1153552	5.423988	9.2564298			
3	19.9006875	25.744843	17.8889738			
4	81.3407181	68.773873	35.6315253			
5	8.0595875	8.842628	6.3343519			
6	0.2387319	0.210084	0.3023937			

1.2 Microarray Data: Areas

We will load the areas data to calculate FC among strains. For each gene, we select the time interval (right, left, mid or sides) for which we find the

maximum difference among strains (between highest and lowest). We will also add a column to check if this time interval corresponds to the interval of maximum expression for each strain.

Areas DataFrame

	Gene_id	1_12B	r_12B	m_12B	s_12B	1_10G	r_10G
1	mal_mito_3	30.592496	61.080128	49.676556	41.99607	25.372470	62.38873
2	MAL13P1.415_oldname	5.423269	8.488971	1.289779	12.62246	6.117132	10.59524
3	MAL13P1.65_oldname	18.322430		NA	17.593468	NA	14.071128
4	MAL7P1.142_oldname	9.389247	12.807814	10.340803	11.85626	13.661078	14.52676
5	MAL8P1.310_oldname		NA	NA	NA	NA	NA
6	MAL8P1.90_oldname		NA	NA	NA	NA	NA
		m_10G	s_10G	1_3D7B	r_3D7B	m_3D7B	s_3D7B
1	49.805504	37.95570	25.484634	62.83441	50.462696	37.856349	30.592496
2	3.676218	13.03616	1.789873	10.51234	3.691753	8.610459	6.117132
3	NA	NA	19.333324		NA	NA	19.333324
4	13.401610	14.78623	7.099032	13.34518	12.041177	8.403034	13.661078
5	NA	NA		NA	NA	NA	NA
6	NA	NA		NA	NA	NA	NA
		MaxRight	MinRight	MaxMid	MinMid	MaxSides	MinSides
1	62.83441	61.080128	50.462696	49.676556	41.99607	37.856349	5.220025
2	10.59524	8.488971	3.691753	1.289779	13.03616	8.610459	4.327259
3	NA	NA		NA	NA	NA	5.262196
4	14.52676	12.807814	13.401610	10.340803	14.78623	8.403034	6.562046
5	NA	NA		NA	NA	NA	NA
6	NA	NA		NA	NA	NA	NA
		DifMid	DifSides	Interval	MaxDif	areaFC	area_12B
1	0.7861401	4.139719		Left	5.220025	0.3881060	30.592496
2	2.4019733	4.425700		Sides	4.425700	0.3290483	12.622460
3	NA	NA		Left	5.262196	0.3912413	18.322430
4	3.0608067	6.383199		Left	6.562046	0.4878845	9.389247
5	NA	NA	No Data		NA	NA	NA
6	NA	NA	No Data		NA	NA	NA
		MaxArea	MinArea				
1	30.59250	25.372470					
2	13.03616	8.610459					
3	19.33332	14.071128					
4	13.66108	7.099032					
5	NA	NA					
6	NA	NA					

1.3 Load RNA-Seq Data

We will use publicly available data from PlasmoDB to create a reference expression percentile for each gene. All data-sets are from RNA-Seq studies in the 3D7 strain. We are using 4 different data-sets:

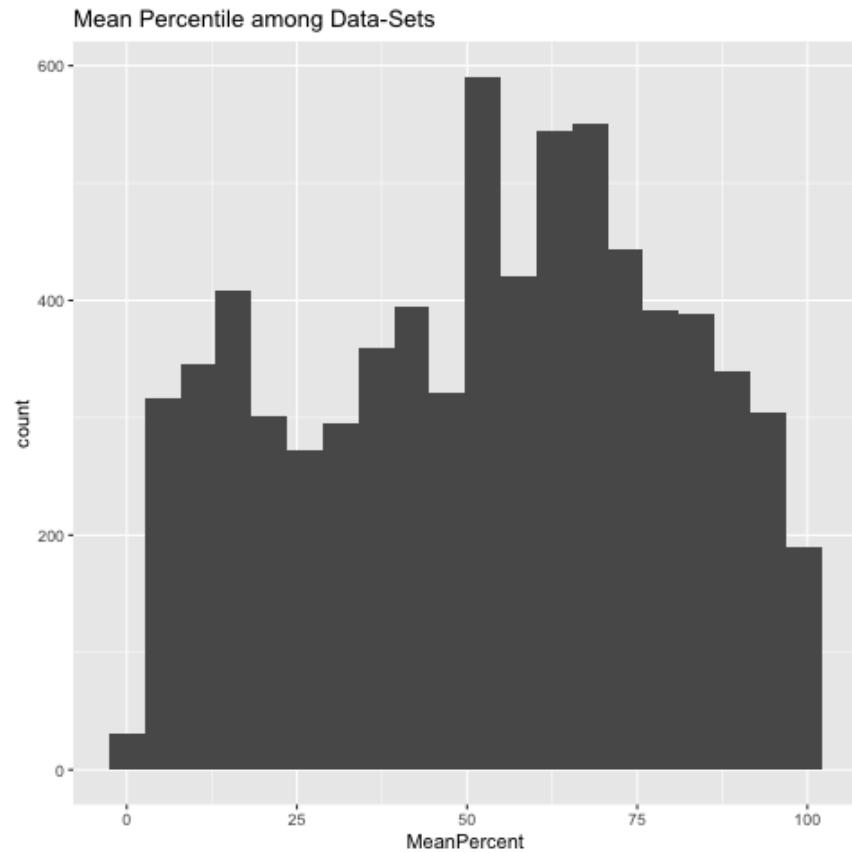
- Otto et.al.
- Hoeijmakers et.al.
- Toenhake et.al.
- Bartfai et.al.

We use only uniquely mapped reads and scaled data when available. For each Data-Set we first create a column representing the maximum value among timepoints for each gene. We then convert this column into percentile values. Finally we average this percentile values among all Data-Sets.

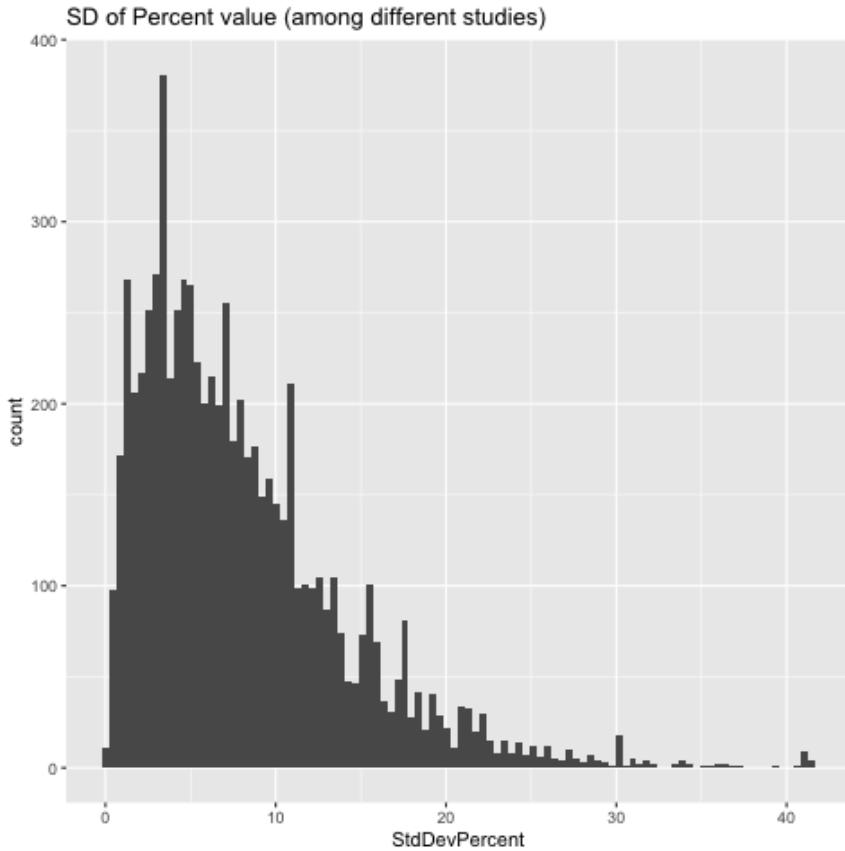
RNA-Seq DataFrame

	Gene_id	Otto_Max_pcnt	Hoeij_Max_pcnt	Toen_Max_pcnt	Bart_Max_pcnt
1	PF3D7_0100100	26.104032	24.855996	17.577239	18.240531
2	PF3D7_0100200	7.366032	25.693838	13.833130	18.764182
3	PF3D7_0100300	22.045732	3.587013	2.469890	1.885146
4	PF3D7_0100400	15.290627	17.315413	10.830861	14.993891
5	PF3D7_0100500	25.885844	16.145924	24.471985	9.399546
6	PF3D7_0100600	7.915867	4.468494	3.717926	3.054634
	MeanPercent	StdDevPercent			
1	21.694449	3.818402			
2	16.414296	6.711285			
3	7.496945	8.421970			
4	14.607698	2.356468			
5	18.975825	6.664638			
6	4.789230	1.873182			

We plot the resulting distribution of percentiles (it should be almost flat).



We plot the standard deviation of the percentile values among different Data-Sets and we can see that for the vast majority of genes it doesn't go above 10.



1.4 Create table for classification

Joining and summarizing all the previous data we create a a table that will let us classify the genes according to their expression state.

To classify genes, we add a column for each strain were we classify genes according to their expression relative to other strains. To create this column we have taken the maximum and minimum area value for each gene and divided this interval in 3 equal regions. Each strain then gets a value of man/mid/max according to which region it falls.

gene-min---|---mid---|---gene-max

Final DataFrame

	Gene_id	Variant	Percent_12B	Percent_10G	Percent_3D7B	MaxRedPercentDif
1	mal_mito_3	FALSE	96.0466005	98.4721161	97.8800611	2.425516

2	PF3D7_0100100	TRUE	5.6913675	10.9625668	11.1153552		5.423988	
3	PF3D7_0100200	TRUE	29.7555386	4.0106952	19.9006875		25.744843	
4	PF3D7_0100300	TRUE	12.5668449	12.9870130	81.3407181		68.773873	
5	PF3D7_0100400	TRUE	9.8930481	1.0504202	8.0595875		8.842628	
6	PF3D7_0100600	TRUE	0.2291826	0.4392666	0.2387319		0.210084	
	MeanRedPercent	Interval	areaFC	area_12B	area_10G	area_3D7B	MaxArea	
1	97.4662592	Left	0.3881060	30.59250	25.372470	25.48463	30.59250	
2	9.2564298	Left	0.8879945	31.76998	43.713503	38.29839	43.71350	
3	17.8889738	Right	3.5015562	52.33320	5.237266	43.02460	52.33320	
4	35.6315253	Left	3.3957657	31.78908	33.073957	77.46213	77.46213	
5	6.3343519	Left	2.0671154	23.21846	7.706832	35.50954	35.50954	
6	0.3023937	No Data	NA	NA	NA	NA	NA	NA
	MinArea	MeanPercent	rel_12B	rel_10G	rel_3D7B	Gene_name		
1	25.372470	92.568511	max	min	min	null		
2	31.769976	21.694449	min	max	mid	VAR		
3	5.237266	16.414296	max	min	max	RIF		
4	31.789080	7.496945	min	min	max	VAR		
5	7.706832	14.607698	mid	min	max	RIF		
6	NA	4.789230	NA	NA	NA	RIF		
			Annot					
1			unspecified product					
2	erythrocyte	membrane protein 1, PfEMP1						
3		rifin						
4	erythrocyte	membrane protein 1, PfEMP1						
5		rifin						
6		rifin						

1.5 Create Lists according to thresholds

Now that we have all the data loaded in, we can start to set labels for each gene.

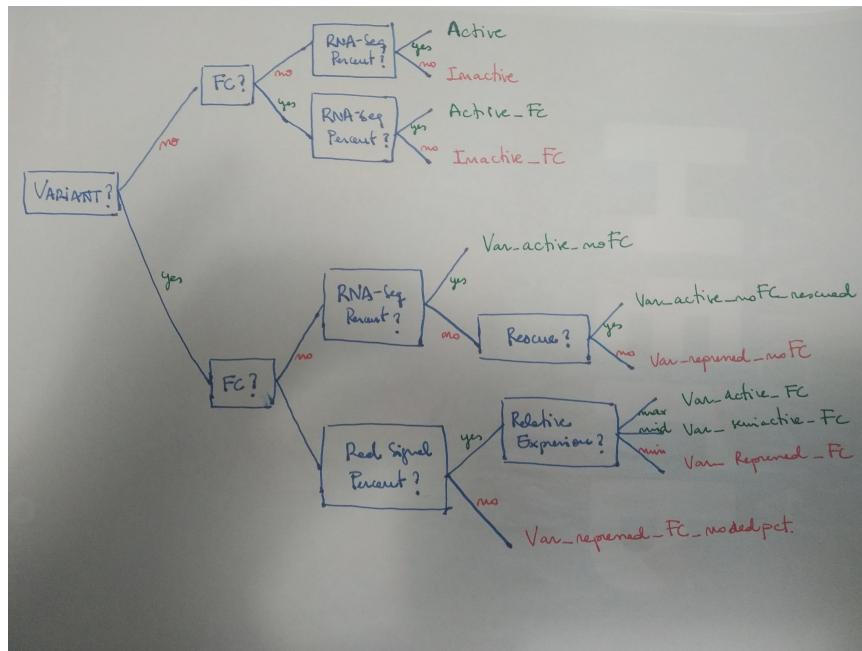
We have set the following thresholds:

- (rna_pcnt) RNA-Seq mean percentile: 25%
- (red_pcnt) Red Signal Percentile (by sample): 25%
- (red_rescue) Red Signal Mean Percentile for "rescuing": 40%
- (area_fc) Area log2 Fold-Change: 1

In addition to these thresholds we will use 2 more columns to set the categories:

- Variant: a column stating if gene is variant/non-variant
- Relative Expression (by strain): a column where each gene is set to min/mid/max according to its expression level relative to the other strains.

We will have the following categories with the following logic:



Genes that lack either an areaFC value or a RNA-Seq mean percentile are currently set to "Non-Settable".

Gene-State DataFrame

	Gene_id	Variant	Percent_12B	Percent_10G	Percent_3D7B	MaxRedPercentDif	
1	mal_mito_3	FALSE	96.0466005	98.4721161	97.8800611	2.425516	
2	PF3D7_0100100	TRUE	5.6913675	10.9625668	11.1153552	5.423988	
3	PF3D7_0100200	TRUE	29.7555386	4.0106952	19.9006875	25.744843	
4	PF3D7_0100300	TRUE	12.5668449	12.9870130	81.3407181	68.773873	
5	PF3D7_0100400	TRUE	9.8930481	1.0504202	8.0595875	8.842628	
6	PF3D7_0100600	TRUE	0.2291826	0.4392666	0.2387319	0.210084	
	MeanRedPercent	Interval	areaFC	area_12B	area_10G	area_3D7B	MaxArea
1	97.4662592	Left	0.3881060	30.59250	25.372470	25.48463	30.59250

2	9.2564298	Left	0.8879945	31.76998	43.713503	38.29839	43.71350
3	17.8889738	Right	3.5015562	52.33320	5.237266	43.02460	52.33320
4	35.6315253	Left	3.3957657	31.78908	33.073957	77.46213	77.46213
5	6.3343519	Left	2.0671154	23.21846	7.706832	35.50954	35.50954
6	0.3023937	No Data	NA	NA	NA	NA	NA
		MinArea	MeanPercent	rel_12B	rel_10G	rel_3D7B	Gene_name
1	25.372470	92.568511	max	min	min	null	
2	31.769976	21.694449	min	max	mid	VAR	
3	5.237266	16.414296	max	min	max	RIF	
4	31.789080	7.496945	min	min	max	VAR	
5	7.706832	14.607698	mid	min	max	RIF	
6	NA	4.789230	NA	NA	NA	RIF	
		Annot				state_12B	
1		unspecified product				Active	
2	erythrocyte membrane protein 1, PfEMP1				Var_Repressed_noFC		
3		rifin			Var_Active_FC		
4	erythrocyte membrane protein 1, PfEMP1	Var_Repressed_FC_noredpcnt					
5		rifin	Var_Repressed_FC_noredpcnt				
6		rifin		Not_settable			
		state_10G			state_3D7B	category_12B	
1		Active			Active	Active	
2	Var_Repressed_noFC		Var_Repressed_noFC	Var_Repressed			
3	Var_Repressed_FC_noredpcnt	Var_Repressed_FC_noredpcnt	Var_Repressed_FC_noredpcnt	Var_Active			
4	Var_Repressed_FC_noredpcnt		Var_Active_FC	Var_Repressed			
5	Var_Repressed_FC_noredpcnt	Var_Repressed_FC_noredpcnt	Var_Repressed_FC_noredpcnt	Var_Repressed			
6		Not_settable		Not_settable	Not_settable	Not_settable	
		category_10G	category_3D7B				
1		Active	Active				
2	Var_Repressed	Var_Repressed					
3	Var_Repressed	Var_Repressed					
4	Var_Repressed	Var_Active					
5	Var_Repressed	Var_Repressed					
6	Not_settable	Not_settable					

1.6 Some results

This is a table with the number of genes in each state for each strain.

States Table

Strain	Active	Active_FC	Inactive	Inactive_FC	Not_settable	Var_Active_FC
--------	--------	-----------	----------	-------------	--------------	---------------

1	12B	7025	29	1110	36	655	8
2	10G	7025	29	1110	36	655	7
3	3D7B	7025	29	1110	36	655	89
			Var_Active_noFC	Var_Active_noFC_rescued	Var_Repressed_FC		
1			118		3	7	
2			118		3	5	
3			118		3	3	
			Var_Repressed_FC_noredpcnt	Var_Repressed_noFC	Var_Semiactive_FC		
1				168	128	1	
2				172	128	NA	
3				92	128	NA	

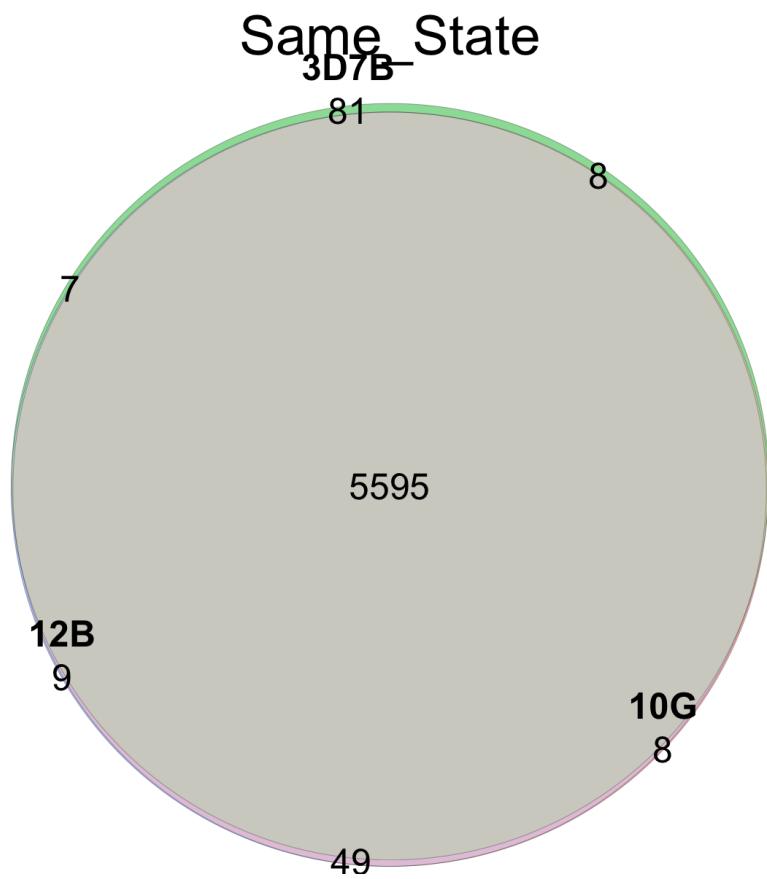
And this are the Clag genes

	Gene_id	Variant	Percent_12B	Percent_10G	Percent_3D7B	MaxRedPercentDiff
1	PF3D7_0302200	TRUE	78.68602	99.57983	42.32238	57.25745
2	PF3D7_0302500	TRUE	98.81589	84.39649	98.98778	14.59129
		MeanRedPercent	Interval	areaFC	area_12B area_10G area_3D7B	MaxArea
1		73.52941	Right	4.823089	37.32948 88.19577	23.32522 88.19577
2		94.06672	Right	3.452270	100.12065 53.68762	99.58767 100.12065
		MinArea	MeanPercent	rel_12B rel_10G rel_3D7B	Gene_name	
1	23.32522	76.36804	min	max	min	CLAG3.2
2	53.68762	88.72404	max	min	max	CLAG3.1
			Annot		state_12B	state_10G
1	cytoadherence	linked asexual protein	3.2	Var_Repressed_FC	Var_Active_FC	
2	cytoadherence	linked asexual protein	3.1	Var_Active_FC	Var_Repressed_FC	
			state_3D7B	category_12B	category_10G category_3D7B	
1	Var_Repressed_FC	Var_Repressed		Var_Active	Var_Repressed	
2	Var_Active_FC		Var_Active	Var_Repressed	Var_Active	

1.7 Venn Diagrams

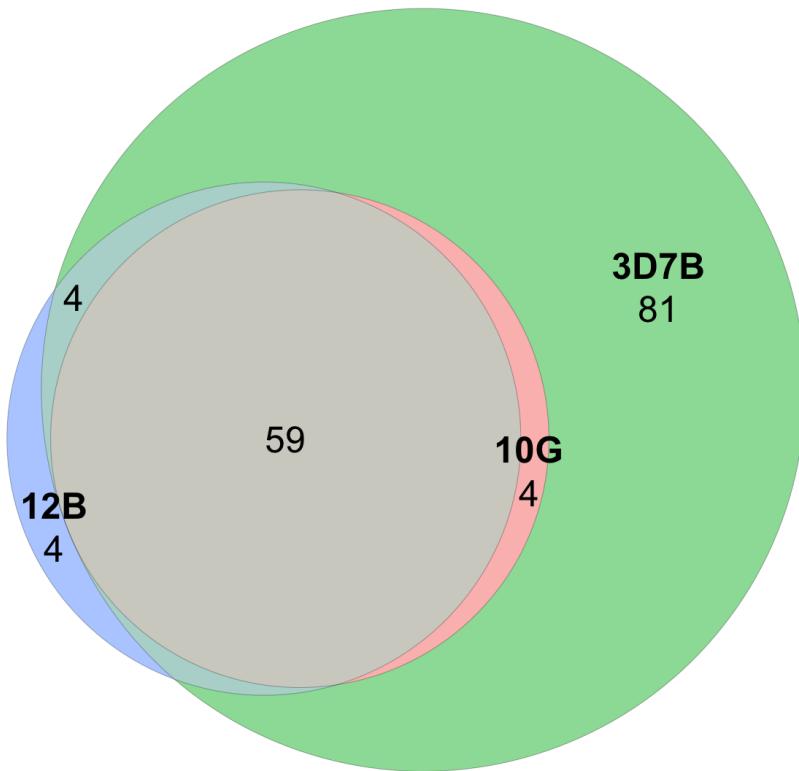
Here we plot Venn diagrams to visualize how the strains compare.

First we check how many genes share the same "state" among strains:

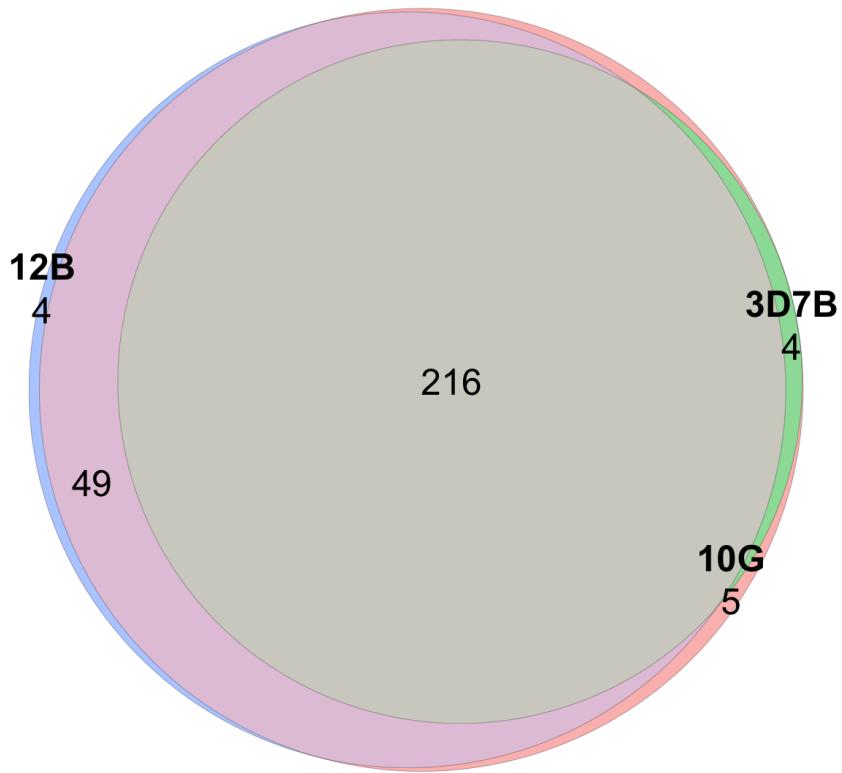


Then we make venn diagrams for the "Var_active" and "Var_Repressed" states.

Var_Active_Genes



Var_Repressed_Genes



2 Code

2.1 Load Packages and functions

```
#### Imports ####

library(readxl)
library(tidyverse)
library(eulerr)

#### Max Dif function ####

max_dif <- function(vect){
  mx <- max(vect, na.rm = T)
  mn <- min(vect, na.rm = T)
```

```

if (is.infinite(mx) | is.infinite(mn)) {
  md <- NA
} else {
  md <- mx - mn
}
return(md)
}



## 2.2 Microarray Data: Red Signal


#### Red Signal DF ####

## Read translation table
map <- read.csv('./Data/oldnames_table.csv', stringsAsFactors = F)
excl <- "./Data/3D7_Variantome_AllData_withGam.xls"

## Import Red Signal table
red <- read_excel(excl, sheet = 4)

colnames(red)[1] <- "Old_id"

red_df <- red %>%
  select(Old_id,
         Red_12B = `Aver.2Higher1.2B.`,
         Red_10G = `Aver.2Higher10G.`,
         Red_3D7B = `Aver.2Higher3D7-B.`) %>%
  left_join(map, by='Old_id')

## Manually add gene ids that do not appear on map (we blasted the probes to assign them)
red_df <- red_df %>%
  mutate(Gene_id = ifelse(Old_id == 'MAL8P1.310', 'PF3D7_0830200', Gene_id)) %>%
  mutate(Gene_id = ifelse(Old_id == 'PFI0905W', 'PF3D7_0918500', Gene_id)) %>%
  mutate(Gene_id = ifelse(Old_id == 'PFL1580W', 'PF3D7_1232700', Gene_id))

## Collapse gene ids that appear more than once (mean expression)
red_df <- red_df %>%
  select(-Old_id) %>%
  group_by(Gene_id) %>% summarize_all(list(mean))

## Filter out rows with untranslated gene ids (marked by '_oldname')

```

```

red_df <- red_df %>%
  filter(!grepl('_oldname', Gene_id))

## Transform into percentiles

red_df <- red_df %>%
  mutate(Percent_12B = (rank(Red_12B)/length(Red_12B))*100) %>%
  mutate(Percent_10G = (rank(Red_10G)/length(Red_10G))*100) %>%
  mutate(Percent_3D7B = (rank(Red_3D7B)/length(Red_3D7B))*100)

## Add max percentile dif

red_df <- red_df %>%
  mutate(MaxRedPercentDif= apply(select(., contains('Percent_')), 1, max_dif)) %>%
  mutate(MeanRedPercent = apply(select(., contains('Percent_')), 1, mean))

print(red_df, width = 200)
hist(red_df$MeanRedPercent)

```

2.3 Microarray Data: Areas

```

##### Areas DF #####
# Import Areas table

area <- read_excel(excl, sheet = 2)

colnames(area)[1] <- "Old_id"

area_df <- area %>%
  select(Old_id,
         l_12B = `left.1.2b`,
         r_12B = `right.1.2b`,
         m_12B = `mid.1.2b`,
         s_12B = `sides.1.2b`,
         l_10G = `left.10g`,
         r_10G = `right.10g`,
         m_10G = `mid.10g`,
         s_10G = `sides.10g`,
         l_3D7B = `left.3d7b`,

```

```

r_3D7B = `right.3d7b`,
m_3D7B = `mid.3d7b`,
s_3D7B = `sides.3d7b`) %>%
  mutate_at(vars(-Old_id), as.numeric) %>%
  left_join(map, by='Old_id') %>%
  select(-Old_id) %>%
  group_by(Gene_id) %>% summarize_all(list(mean))

print(area_df, width = 200)

area_df <- area_df %>%
  mutate(MaxLeft = apply(select(., contains('l_')), 1, max)) %>%
  mutate(MinLeft = apply(select(., contains('l_')), 1, min)) %>%
  mutate(MaxRight = apply(select(., contains('r_')), 1, max)) %>%
  mutate(MinRight = apply(select(., contains('r_')), 1, min)) %>%
  mutate(MaxMid = apply(select(., contains('m_')), 1, max)) %>%
  mutate(MinMid = apply(select(., contains('m_')), 1, min)) %>%
  mutate(MaxSides = apply(select(., contains('s_')), 1, max)) %>%
  mutate(MinSides = apply(select(., contains('s_')), 1, min)) %>%
  mutate(DifLeft = MaxLeft - MinLeft) %>%
  mutate(DifRight = MaxRight - MinRight) %>%
  mutate(DifMid = MaxMid - MinMid) %>%
  mutate(DifSides = MaxSides - MinSides)

print(area_df, width = 200)

## Add max interval and difference

maxinterval <- area_df %>%
  select(Gene_id, contains('Dif')) %>%
  pivot_longer(-Gene_id, names_to = 'Interval', values_to = 'MaxDif') %>%
  group_by(Gene_id) %>%

```

```

filter(rank(-MaxDif, ties.method = "first") == 1) %>%
mutate(Interval = ifelse(is.na(MaxDif), 'No Data', Interval)) %>%
mutate(Interval = case_when(Interval == 'DifLeft' ~ 'Left',
                             Interval == 'DifRight' ~ 'Right',
                             Interval == 'DifMid' ~ 'Mid',
                             Interval == 'DifSides' ~ 'Sides',
                             Interval == 'No Data' ~ 'No Data')) %>%
mutate(areaFC = MaxDif/13.45)

maxinterval

area_df <- area_df %>%
  left_join(maxinterval, by = 'Gene_id')

print(area_df, width = 400)

## Select appropiate area for each gene and add max and min areas

area_df <- area_df %>%
  mutate(area_12B = case_when(
    Interval == 'Left' ~ l_12B,
    Interval == 'Right' ~ r_12B,
    Interval == 'Mid' ~ m_12B,
    Interval == 'Sides' ~ s_12B,
    Interval == 'No Data' ~ NA_real_)) %>%
  mutate(area_10G = case_when(
    Interval == 'Left' ~ l_10G,
    Interval == 'Right' ~ r_10G,
    Interval == 'Mid' ~ m_10G,
    Interval == 'Sides' ~ s_10G,
    Interval == 'No Data' ~ NA_real_)) %>%
  mutate(area_3D7B = case_when(
    Interval == 'Left' ~ l_3D7B,
    Interval == 'Right' ~ r_3D7B,
    Interval == 'Mid' ~ m_3D7B,
    Interval == 'Sides' ~ s_3D7B,
    Interval == 'No Data' ~ NA_real_)) %>%
  mutate(MaxArea = apply(select(., contains('area_')), 1, max)) %>%
  mutate(MinArea = apply(select(., contains('area_')), 1, min))

```

```
print(area_df, width = 400)
```

2.4 Load RNA-Seq Data

```
## ##### Old approax #####
## Select max percentile directly

## otto <- read_delim("./Data/RNA_Seq_Percentiles/PlasmoDB_Otto.csv", delim=";") %>%
##   select(Gene_id = `Gene ID`, MaxPercOtto = `Max %ile (Within Chosen Samples)`)
## hoeij <- read_delim("./Data/RNA_Seq_Percentiles/PlasmoDB_Hoeijmakers.csv", delim=";") %>%
##   select(Gene_id = `Gene ID`, MaxPercHoeij = `Max %ile (Within Chosen Samples)`)
## toen <- read_delim("./Data/RNA_Seq_Percentiles/PlasmoDB_Toenke.csv", delim=";") %>%
##   select(Gene_id = `Gene ID`, MaxPercToen = `Max %ile (Within Chosen Samples)`)
## bart <- read_delim("./Data/RNA_Seq_Percentiles/PlasmoDB_Bartfai.csv", delim=";") %>%
##   select(Gene_id = `Gene ID`, MaxPercBart = `Max %ile (Within Chosen Samples)`)

## Otto Data-Set

csv <- './Data/RNA_Seq_Percentiles/rnaseq_otto_normvals.csv'

otto <- read_delim(csv, delim=";") %>%
  select(Gene_id = `Gene ID`, contains('unique'))

otto <- otto %>%
  mutate(Max = apply(select(., contains('unique')), 1, max)) %>%
  mutate(Otto_Max_pcnt = (rank(Max)/length(Max))*100)

otto <- otto %>% select(Gene_id, Otto_Max_pcnt)
otto

hist(otto$Otto_Max_pcnt)

## Hoeijmakers Data-Set

csv <- './Data/RNA_Seq_Percentiles/rnaseq_hoeijmakers_normvals.csv'

hoeij <- read_delim(csv, delim=";") %>%
  select(Gene_id = `Gene ID`, contains('scaled'))
```

```

hoeij <- hoeij %>%
  mutate(Max = apply(select(., contains('scaled'))), 1, max)) %>%
  mutate(Hoeij_Max_pcnt = (rank(Max)/length(Max))*100)

hoeij <- hoeij %>% select(Gene_id, Hoeij_Max_pcnt)
hoeij

hist(hoeij$Hoeij_Max_pcnt)

## Toenhake Data-Sets

csv <- './Data/RNA_Seq_Percentiles/rnaseq_toen_normvals.csv'

toen <- read_delim(csv, delim=",") %>%
  select(Gene_id = `Gene ID`, contains('unique'))

toen <- toen %>%
  mutate(Max = apply(select(., contains('unique'))), 1, max)) %>%
  mutate(Toen_Max_pcnt = (rank(Max)/length(Max))*100)

toen <- toen %>% select(Gene_id, Toen_Max_pcnt)
toen

hist(toen$Toen_Max_pcnt)

## Bartfai Data-Sets

csv <- './Data/RNA_Seq_Percentiles/rnaseq_bartfai_normvals.csv'

bart <- read_delim(csv, delim=",") %>%
  select(Gene_id = `Gene ID`, contains('scaled'))

bart <- bart %>%
  mutate(Max = apply(select(., contains('scaled'))), 1, max)) %>%
  mutate(Bart_Max_pcnt = (rank(Max)/length(Max))*100)

bart <- bart %>% select(Gene_id, Bart_Max_pcnt)
bart

```

```

hist(bart$Bart_Max_pcnt)

## Join DF
rna_df <- otto %>%
  full_join(hoeij) %>%
  full_join(toen) %>%
  full_join(bart)

## Add mean and sd
rna_df <- rna_df %>%
  mutate(MeanPercent = apply(select(., -Gene_id), 1, mean)) %>%
  mutate(StdDevPercent = apply(select(., -Gene_id), 1, sd))

print(rna_df, width=200)

hist(rna_df$MeanPercent, breaks = 20)
hist(rna_df$StdDevPercent, breaks = 100)

**

```

2.5 Load Annotation

```

annot_df <- read_delim('./Data/plasmoDB_geneAnnot.csv', delim = ';' ) %>%
  select(Gene_id = `Gene ID`,
         Gene_name = `Gene Name or Symbol`,
         Annot = `Product Description`)

print(annot_df, width=200)

```

2.6 Create Join DF

```

print(red_df, width = 200)
print(area_df, width = 200)
print(rna_df, width = 200)

all_df <- select(red_df, Gene_id, contains('Percent'), MeanRedPercent) %>%
  full_join(select(area_df, Gene_id, Interval, contains('area')), by = 'Gene_id') %>%
  full_join(select(rna_df, Gene_id, MeanPercent), by = 'Gene_id')

```

```

## Add Variant Genes information

cvg <- read_excel("./Data/CVG_list_jan2020_final.xlsx", sheet = "Final")

final_df <- cvg %>%
  select("Gene_id" = `Gene ID`, "Variant" = `Final Customized`) %>%
  right_join(all_df, by = 'Gene_id') %>%
  mutate(Variant = recode(Variant, YES = TRUE, NO = FALSE, .missing = FALSE))

print(final_df, width = 200)

## Here we create a dplyr function.
## To be able to use variables (for colnames) we need to use the special quote functions
## Colnames to use inside functions must be "enquoted" before usage and preceded by !!
## Colnames to assign must be "enquoted" first, preceded by !! and assigned by :=

## First create a col where we set categories for each gene according relative express
## For each gene: gene-min----/---mid----/---gene-max

relexprs <- function(vect){
  if (any(is.na(vect))){
    return(NA)
  } else {
    labs = c('min', 'mid', 'max')
    lab <- cut(vect, 3, labels = labs)[1]
    return(as.character(lab))
  }
}

set_relexprs <- function(df, outcol, areacol){
  outcol <- enquo(outcol)
  areacol <- enquo(areacol)
  df %>%
    mutate (!! outcol := apply(select(., !! areacol, MaxArea, MinArea), 1, relexprs))
}

final_df <- final_df %>%
  set_relexprs(rel_12B, area_12B) %>%
  set_relexprs(rel_10G, area_10G) %>%

```

```

set_relexprs(rel_3D7B, area_3D7B)

## Add annotation
final_df <- left_join(final_df, annot_df, by = 'Gene_id')

print(final_df, width = 200)



### 2.7 Create Lists according to thresholds



print(final_df, width = 200)

th_rnapcnt <- 25
th_redpcnt <- 25
th_redrescue <- 40
th_areaFC <- 1

## Here we create a dplyr function.
##To be able to use variables (for colnames) we need to use the special quote functions
## Colnames to use inside functions must be "enquoted" before usage and preceded by !!
## Colnames to assign must be "enquoted" first, preceded by !! and assigned by :=

set_state <- function(df, statecol, redcol, relcol){

  statecol <- enquo(statecol)
  redcol <- enquo(redcol)
  relcol <- enquo(relcol)

  df <- df %>%
    mutate (!! statecol := case_when(
      ## Actiu
      !Variant &
      areaFC < th_areaFC &
      MeanPercent >= th_rnapcnt ~ 'Active',
      ## Inactiu
      !Variant &
      areaFC < th_areaFC &
      MeanPercent < th_rnapcnt ~ 'Inactive',
      ## No variant amb FC
      TRUE ~ 'No variant'))

  return(df)
}

```

```

!Variant &
areaFC > th_areaFC &
MeanPercent >= th_rnapcnt ~ 'Active_FC', 

!Variant &
areaFC > th_areaFC &
MeanPercent < th_rnapcnt ~ 'Inactive_FC', 

## Var actiu
Variant &
areaFC < th_areaFC &
MeanPercent >= th_rnapcnt ~ 'Var_Active_noFC', # noFC

Variant &
areaFC < th_areaFC &
MeanPercent < th_rnapcnt &
MeanRedPercent >= th_redrescue ~ 'Var_Active_noFC_rescued', # noFC, resc

Variant &
areaFC >= th_areaFC &
!! redcol >= th_redpcnt &
!! relcol == 'max' ~ 'Var_Active_FC', # Variant, FC, redpcnt, max

Variant &
areaFC >= th_areaFC &
!! redcol >= th_redpcnt &
!! relcol == 'mid' ~ 'Var_Semiactive_FC', # Variant, FC, redpcnt, mid

## Var repressed
Variant &
areaFC < th_areaFC &
MeanPercent < th_rnapcnt &
MeanRedPercent < th_redrescue ~ 'Var_Repressed_noFC', # noFC, noRescue

Variant &
areaFC >= th_areaFC &
!! redcol >= th_redpcnt &
!! relcol == 'min' ~ 'Var_Repressed_FC', # Variant, FC, redpcnt, min

Variant &

```

```

areaFC >= th_areaFC &
  !! redcol < th_redpcnt ~ 'Var_Repressed_FC_noredpcnt', # Variant, FC, ...

## Not settable
is.na(areaFC) | is.na(MeanPercent) ~ 'Not_settable',
TRUE ~ 'Wrong!'))}

## The 'TRUE ~ ...' handles rows that do not match any of previous patterns.
## Here we use it to make sure all rows are set (no "Wrong!" appearing)

return(df)
}

set_category <- function(df, statecol, categorycol){

  statecol <- enquo(statecol)
  categorycol <- enquo(categorycol)

  df <- df %>%
    mutate (!! categorycol := case_when(
      !! statecol == 'Active' ~ 'Active',
      !! statecol == 'Inactive' ~ 'Inactive',
      !! statecol == 'Active_FC' ~ 'Active',
      !! statecol == 'Inactive_FC' ~ 'Inactive',
      !! statecol == 'Var_Active_noFC' ~ 'Var_Active',
      !! statecol == 'Var_Active_noFC_rescued' ~ 'Var_Active',
      !! statecol == 'Var_Repressed_noFC' ~ 'Var_Repressed',
      !! statecol == 'Var_Active_FC' ~ 'Var_Active',
      !! statecol == 'Var_Semiactive_FC' ~ 'Var_Semiactive',
      !! statecol == 'Var_Repressed_FC' ~ 'Var_Repressed',
      !! statecol == 'Var_Repressed_FC_noredpcnt' ~ 'Var_Repressed',
      !! statecol == 'Not_settable' ~ 'Not_settable',
      TRUE ~ 'No_Category!')))

  return(df)
}

## We now set each gene to it's state

```

```

state_df <- final_df %>%
  set_state(state_12B, Percent_12B, rel_12B) %>%
  set_state(state_10G, Percent_10G, rel_10G) %>%
  set_state(state_3D7B, Percent_3D7B, rel_3D7B) %>%
  set_category(state_12B, category_12B) %>%
  set_category(state_10G, category_10G) %>%
  set_category(state_3D7B, category_3D7B)

print(state_df, width = 400)

```

2.8 Some checks and results

```

## We check no rows are set to "Wrong!"
state_df %>%
  filter(state_12B == 'Wrong!' | state_10G == 'Wrong!' | state_3D7B == 'Wrong!') %>%
  print(width = 400)

state_df %>%
  filter(category_12B == 'No_Category' |
         category_10G == 'No_Category' |
         category_3D7B == 'No_Category')

## Save results
write.csv(state_df, './Results_Tables/state_df_rna25_red25_reddif0_areal.csv')

## Create a table with number of each state per strain
state_table <- bind_rows(table(state_df$state_12B),
                         table(state_df$state_10G),
                         table(state_df$state_3D7B)) %>%
  replace_na(list(Var_Semiactive = 0)) %>%
  mutate(Strain = c('12B', '10G', '3D7B')) %>%
  select(Strain, everything())

## Create a table with differences between 12B and 10G
dif12B_10G <- state_df %>%
  filter(state_12B != state_10G) %>%
  select(Gene_id, contains('12B'), contains('10G'), Gene_name, Annot)

## Check Clags
clags <- state_df %>%

```

```

filter(Gene_id == 'PF3D7_0302500' | Gene_id == 'PF3D7_0302200')

write.csv(clags, './Results_Tables/clag_genes.csv')

print(state_table, width = 200)
summary(rna_df)

2.9 Comparison of 12B vs 10G differences

excl <- "./Data/10Gvs1p2B.xlsx"

## Import 12B vs 10G differences by transcription table
trans_difs <- read_delim('./Data/transDif_12Bvs10G.csv', delim = ';') %>%
  rename(Old_id = X1) %>%
  left_join(map, by = 'Old_id')

print(trans_difs, width = 400)

trans_difs %>%
  select(Old_id, Gene_id)

dif12B_10G

table(trans_difs$Gene_id %in% dif12B_10G$Gene_id)
table(dif12B_10G$Gene_id %in% trans_difs$Gene_id)

allids <- unique(c(dif12B_10G$Gene_id, trans_difs$Gene_id))

compare_12Bvs10G <- state_df %>%
  filter(Gene_id %in% allids) %>%
  select(Gene_id,
         Variant, areaFC,
         MeanRedPercent,
         contains('12B'),
         contains('10G'),
         Gene_name,
         Annot) %>%
  mutate(TransDif = Gene_id %in% trans_difs$Gene_id) %>%
  mutate(Dif_state = category_12B != category_10G)

```

```

print(compare_12Bvs10G, width = 400)
write.csv(compare_12Bvs10G, './Results_Tables/gens_dif12B_10G.csv')

```

2.10 Venn Diagram of results

```

makeIntersects <- function(a,b,c){

  a_b <- intersect(a, b)
  a_c <- intersect(a, c)
  b_c <- intersect(b, c)
  a_b_c <- intersect(a_b, c)

  abc <- a_b_c
  ab <- a_b[!a_b %in% a_b_c]
  ac <- a_c[!a_c %in% a_b_c]
  bc <- b_c[!b_c %in% a_b_c]

  a <- a[!a %in% ab & !a %in% ac & !a %in% abc]
  b <- b[!b %in% ab & !b %in% bc & !b %in% abc]
  c <- c[!c %in% ac & !c %in% bc & !c %in% abc]

  return(list(a = a, b = b, c = c, ab = ab, bc = bc, ac = ac, abc = abc))
}

customEuler <- function(a,b,c, name){

  intersects <- makeIntersects(a,b,c)
  areas <- lapply(intersects, function(x) length(x))

  fit <- euler(c(A=areas$a, B=areas$b, C=areas$c,
                  "A&B"=areas$ab, "A&C"=areas$ac, "B&C"=areas$bc,
                  "A&B&C" = areas$abc))

  d <- plot(fit, fills = list(fill = c("#619cff", "#f8766d", "#00ba38"), alpha = 0.5),
            edges = list(lwd = 0.1), quantities = list(quantities = T),
            labels = list(labels=c("12B", "10G", "3D7B")),
            main = name)

  ggsave(d, filename = paste0('./Plots/', "venn_", name, ".png"),
}

```

```

    device = "png", width = 10, height = 10, units = "cm")

  plot(d)
  print(fit)
}

## Var active

va_12B <- state_df %>%
  filter(category_12B == "Var_Active") %>%
  select(Gene_id) %>%
  pull()

va_10G <- state_df %>%
  filter(category_10G == "Var_Active") %>%
  select(Gene_id) %>%
  pull()

va_3D7B <- state_df %>%
  filter(category_3D7B == "Var_Active") %>%
  select(Gene_id) %>%
  pull()

customEuler(va_12B, va_10G, va_3D7B, 'Var_Active_Genes')

## Var inactive

vi_12B <- state_df %>%
  filter(category_12B == "Var_Repressed") %>%
  select(Gene_id) %>%
  pull()

vi_10G <- state_df %>%
  filter(category_10G == "Var_Repressed") %>%
  select(Gene_id) %>%
  pull()

vi_3D7B <- state_df %>%

```

```

filter(category_3D7B == "Var_Repressed") %>%
select(Gene_id) %>%
pull()

customEuler(vi_12B, vi_10G, vi_3D7B, 'Var_Repressed_Genes')

## Same category
## We fuse the gene_id with it's state so that genes with same state will be exactly the same

print(state_df, width = 400)

genestate <- state_df %>%
  mutate(gs_12B = paste(Gene_id, category_12B, sep = "_")) %>%
  mutate(gs_10G = paste(Gene_id, category_10G, sep = "_")) %>%
  mutate(gs_3D7B = paste(Gene_id, category_3D7B, sep = "_")) %>%
  select(Gene_id,
         category_12B, category_10G, category_3D7B,
         gs_12B, gs_10G, gs_3D7B)

customEuler(genestate$gs_12B, genestate$gs_10G, genestate$gs_3D7B, 'Same_State')

x <- makeIntersects(genestate$gs_12B, genestate$gs_10G, genestate$gs_3D7B)}

```