

Supplementary Information

Alberto Scarampi

Contents

1 Methyl viologen treatment leads to enhanced light-dependent intracellular ROS accumulation	2
2 Adaptation to methyl viologen is not due to its abiotic degradation over time	3
3 Spot plates of strains grown in BG11 without or with MV (6 μM)	4
4 Addition of methyl viologen at various growth phases	5
5 The gene <i>prqA</i> is not essential for spontaneous evolution of MV resistance	6
6 Whole genome sequencing and bioinformatic analyses	7
6.1 Genome sequencing QC report	7
6.2 Sequenced strains compared to various substrains	11
6.3 SNPs Polimorphism	12
7 Extended methods	14
7.1 Buffers and media	14
7.1.1 0.25 M EDTA stock solution - pH8	14
7.1.2 1M Tris/HCl buffer - pH8	15
7.1.3 Lysis buffer for gDNA extraction ("Smoker B")	15
7.1.4 Photoautotrophic growth medium and plates (BG11)	15
8 Code	19
8.1 Comare variants across strains	19

1 Methyl viologen treatment leads to enhanced light-dependent intracellular ROS accumulation

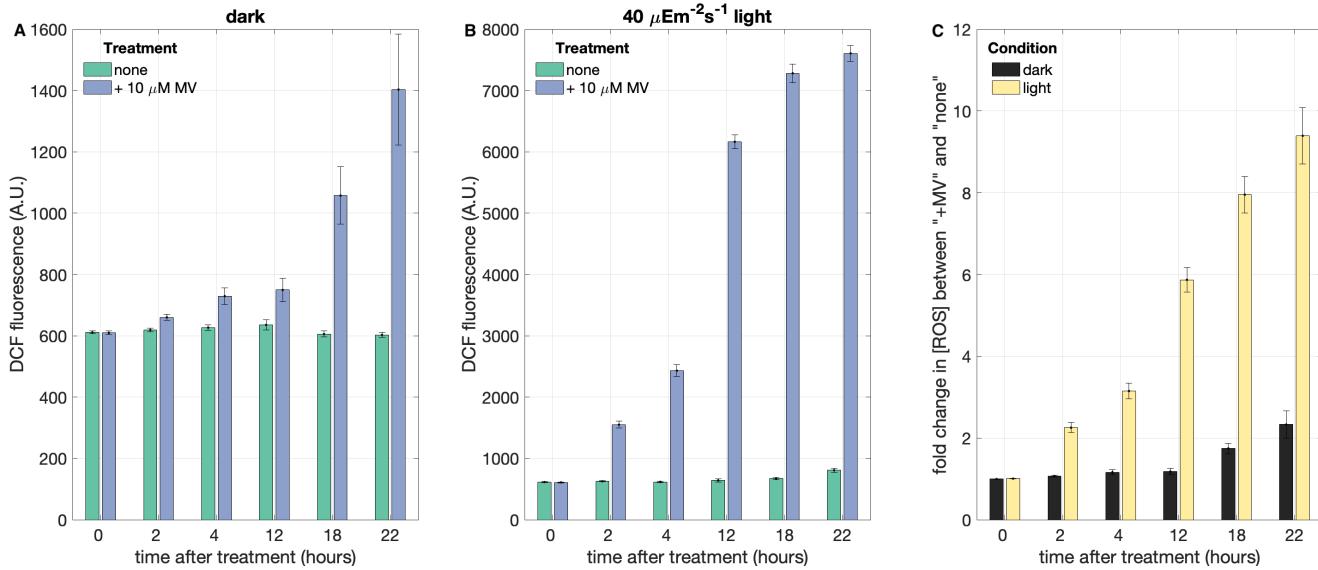


Figure 1: ROS quantification assay based on the substrate DCFH-DA, which becomes fluorescent upon reaction with intracellular reactive oxygen species. A) DCF fluorescence in cultures of *Synechocystis* at various time points after treatment with 10 μ M methyl viologen (in the dark). B) DCF fluorescence at various time points after treatment with 10 μ M MV (under 40 μ mol·m $^{-2}$ ·s $^{-1}$ of white light illumination). C) Fold changes in DCF fluorescence between cultures treated with methyl viologen and no treatment control in the darkness and light at various time points. Error bars represent standard error of the mean from three biological replicates (individual colonies treated independently).

2 Adaptation to methyl viologen is not due to its abiotic degradation over time

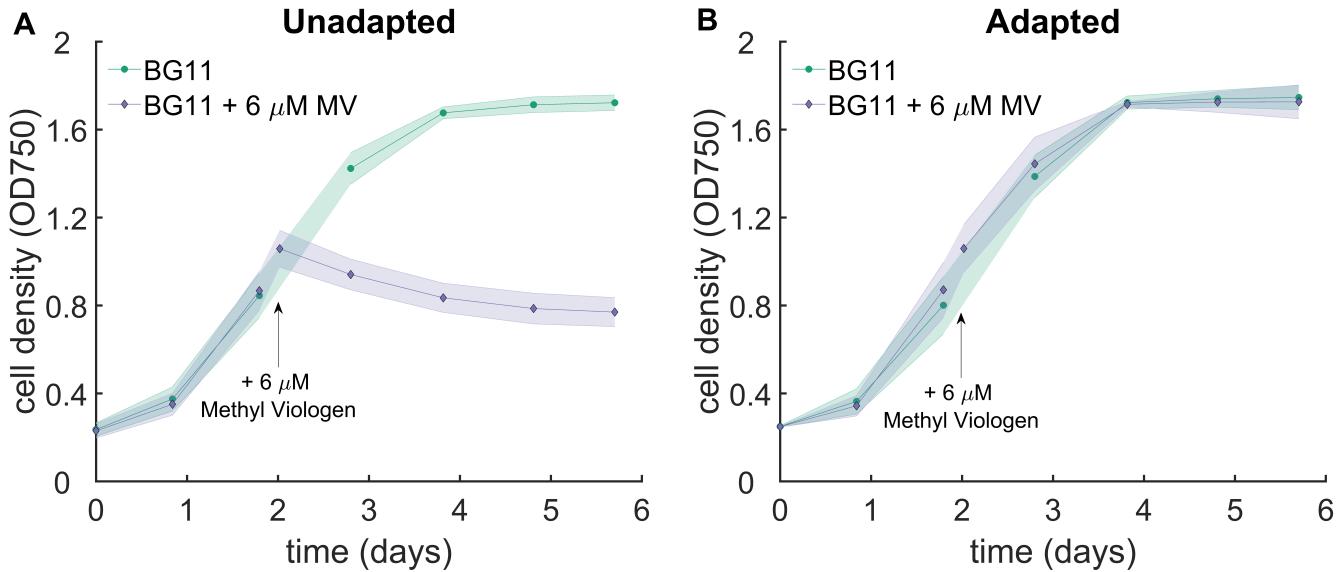


Figure 2: Growth curves of three independent unadapted (A) and MV-adapted (B) cultures of *Synechocystis* sp. PCC 6803 growing at 30°C under constant illumination (intensity = 100 $\mu\text{mol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$) in the absence (green trace) and upon addition of 6 μM methyl viologen (from a fresh stock) during exponential growth (purple trace). Shaded regions represent standard errors over the means from three biological replicates.

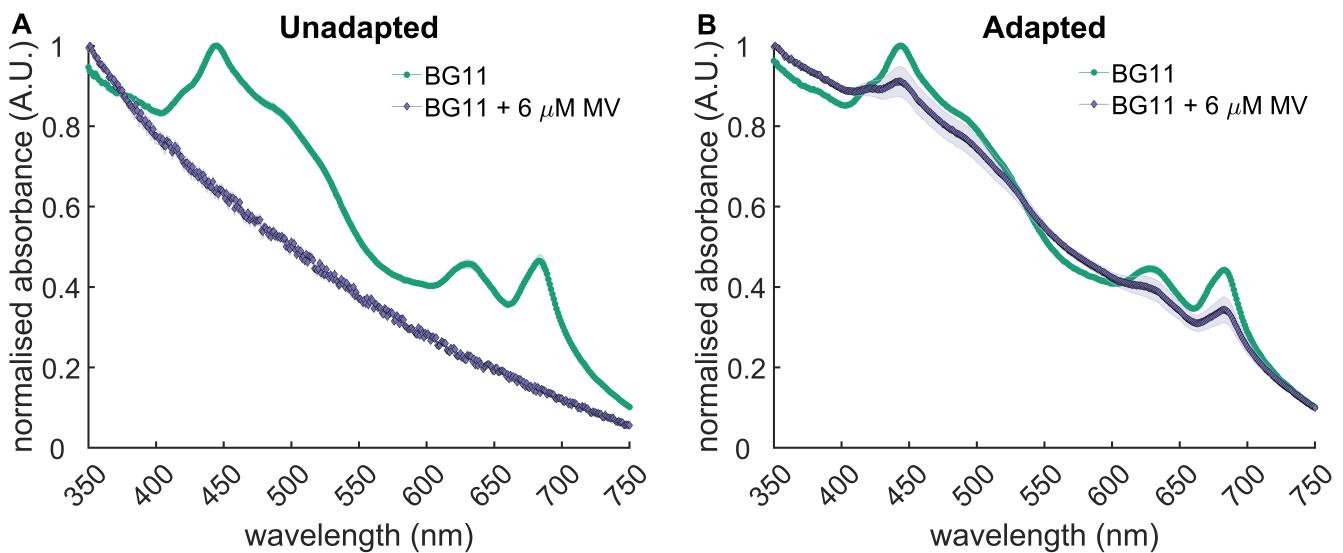


Figure 3: Absorbance spectra of unadapted (A) and adapted (B) cultures of *Synechocystis* sp. PCC 6803 after 6 days of growth at 30°C under constant illumination (intensity = 100 $\mu\text{mol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$) in the absence (green trace) and upon addition of 6 μM methyl viologen during mod-log phase (purple trace). Shaded regions represent standard errors over the means from three biological replicates.

3 Spot plates of strains grown in BG11 without or with MV (6 μ M)

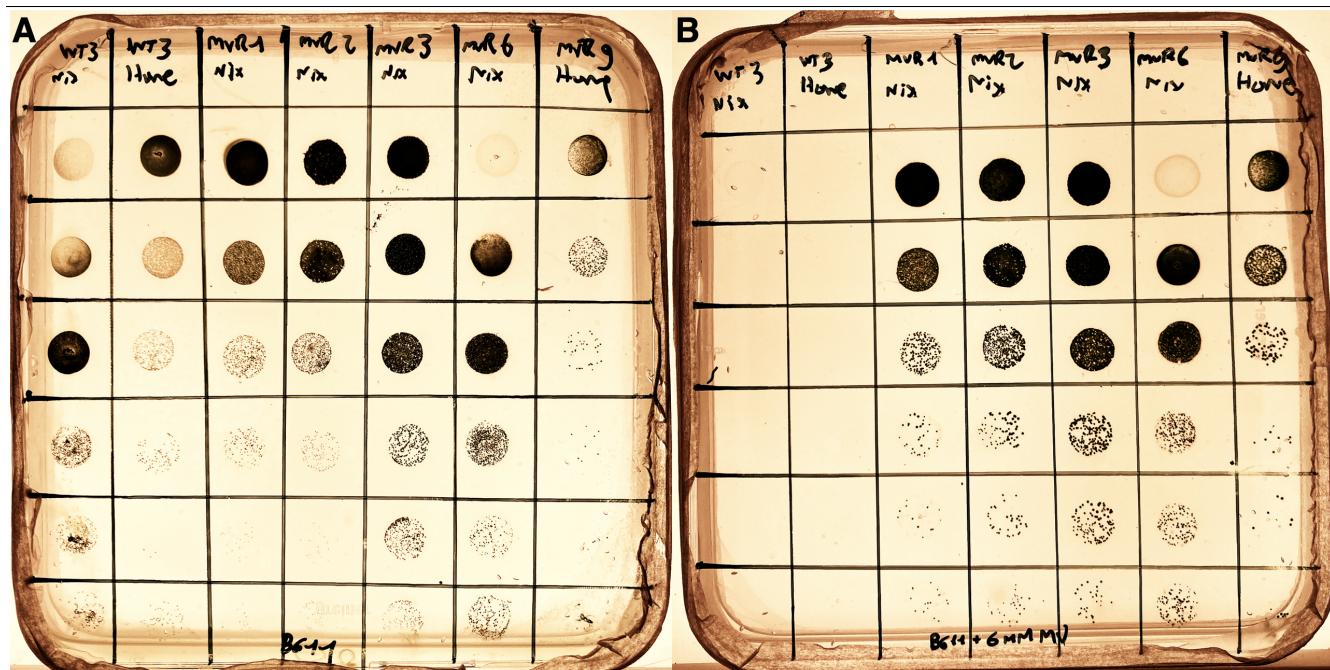


Figure 4: Spot plates of 2 unadapted (first two columns on each plates) and 5 adapted (columns 3 to 7) cultures of *Synechocystis* sp. PCC 6803 in BG11 plates without methyl viologen (A) and with the addition of 6 μ M methyl viologen (B). Rows correspond to consecutive 10-fold serial dilutions of culture inocula (all starting at $OD_{750} = 1$). Plates were grown at 30°C under constant illumination (intensity = 100 μ mol·m $^{-2}$ ·s $^{-1}$) for 10 days.

4 Addition of methyl viologen at various growth phases

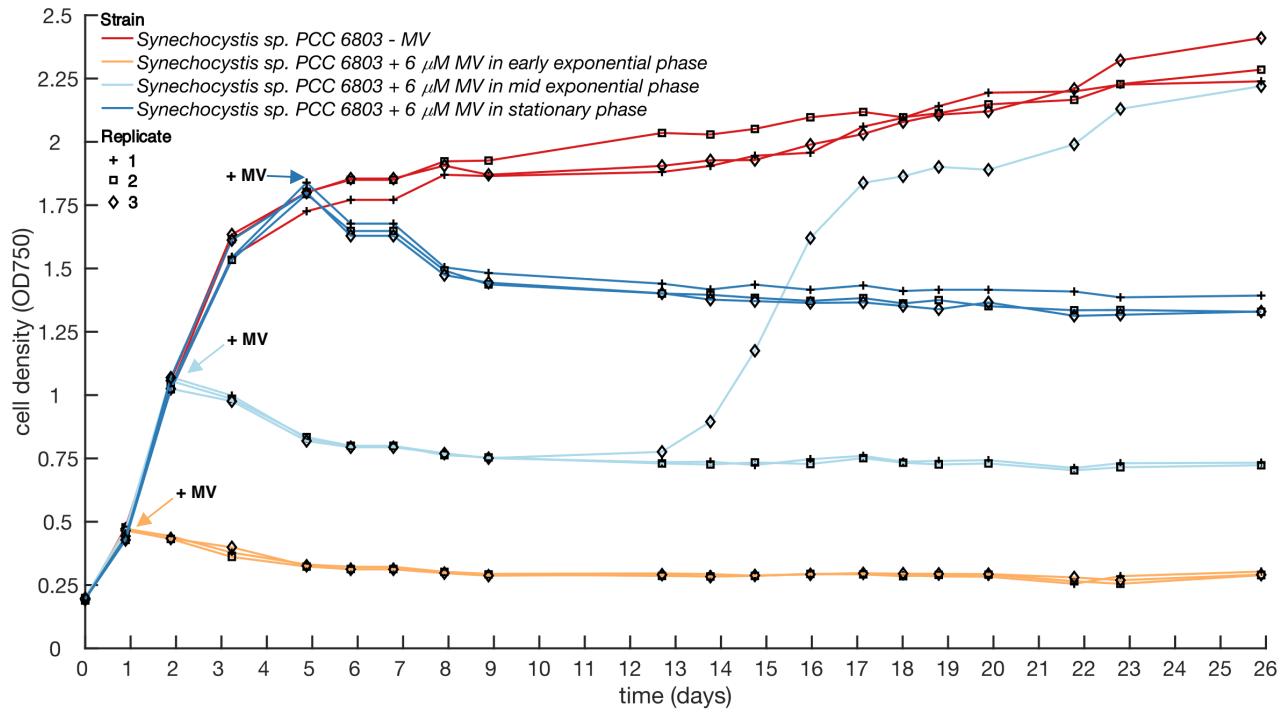


Figure 5: Growth curves of *Synechocystis* sp. PCC 6803 strains grown at 30 °C under constant illumination (intensity = 40 $\mu\text{mol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$) in the absence (red trace) and upon addition of 6 μM methyl viologen during various growth stages (as indicated by the arrows colored according to the legend on top right).

5 The gene *prqA* is not essential for spontaneous evolution of MV resistance

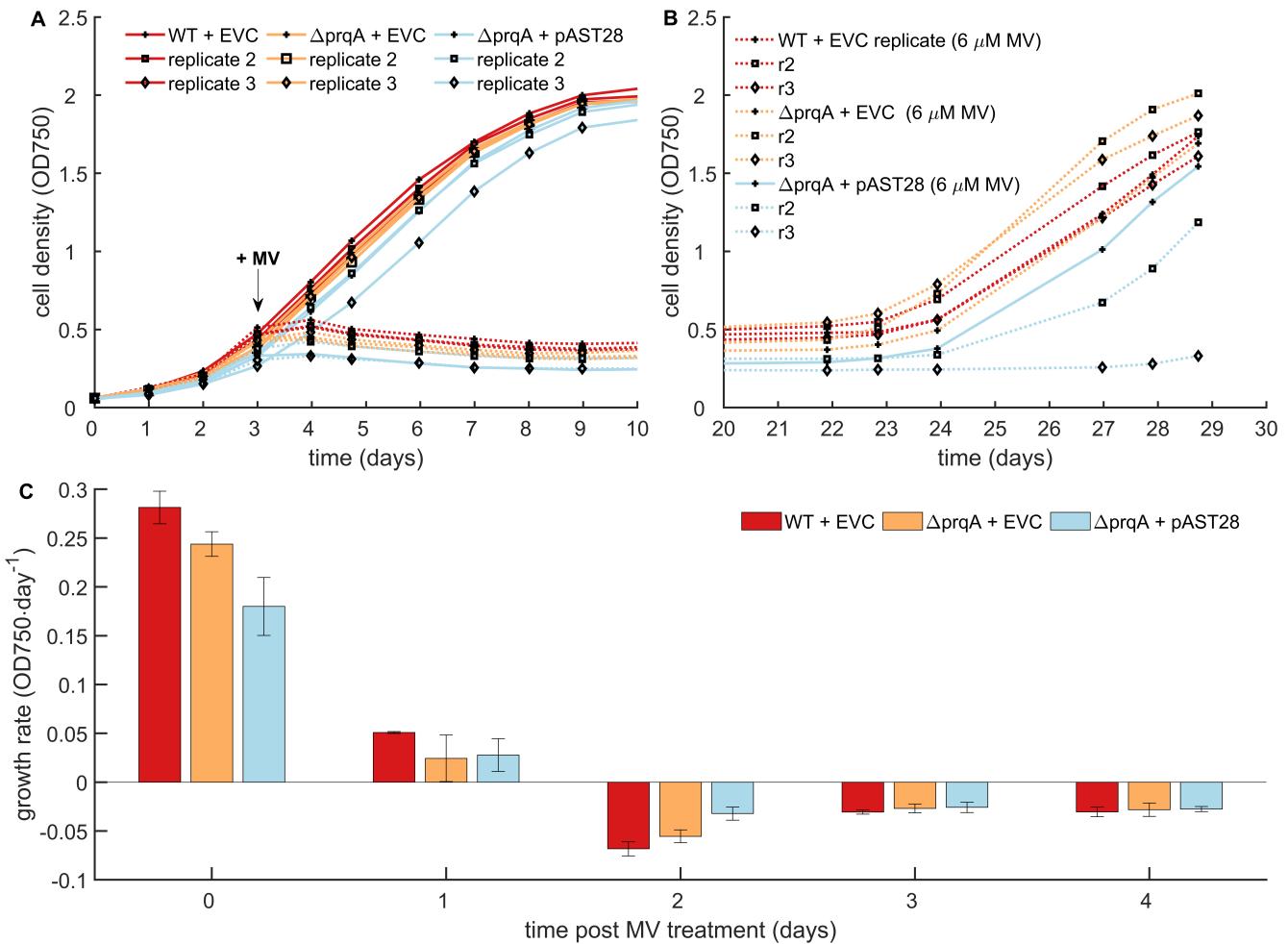


Figure 6: Growth curves of wild-type, $\Delta prqA$ and complemented $\Delta prqA$ strains of *Synechocystis* sp. PCC 6803 grown at 30 °C under constant illumination (intensity = 100 $\mu\text{mol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$) in the absence (green trace) and upon addition of 6 μM methyl viologen during exponential growth (purple trace).

6 Whole genome sequencing and bioinformatic analyses

6.1 Genome sequencing QC report

Sample	Concentration (ng/ μ l)	Volume (μ l)	Effective (%)	Error (%)	Q20 (%)	Q30 (%)	GC (%)
WT_Nixon	31.80	92	99.82	0.03	97.50	92.73	55.06
WT_Howe	27.80	98	99.78	0.03	96.75	90.99	46.50
mvR01_Nixon	36.40	95	99.77	0.03	97.43	92.60	52.66
mvR02_Nixon	25.00	97	99.78	0.03	97.34	92.30	47.98
mvR03_Nixon	30.00	101	99.81	0.03	97.65	93.02	48.64
mvR06_Nixon	44.60	98	99.80	0.03	97.19	91.96	51.59
mvR09_Howe	20.60	95	99.78	0.03	97.28	92.16	47.39
mvR10_Howe	20.20	95	99.86	0.03	97.66	93.39	47.83
mvR11_Howe	27.80	92	99.87	0.03	97.55	93.14	46.53
mvR12_Howe	19.60	98	99.86	0.03	97.65	93.31	47.50

Alignment of the trimmed and paired reads from whole genome sequencing experiments was carried out against several reference genome sequences of *Synechocystis* substrains (Figures 8 and 8). 7 shows the results of the type and number of mutations of both wild types as compared to all *Synechocystis* substrain reference genomes available in the NCBI datatbase. For both wild-types, the GT-Kazusa reference genome was the one that resulted in the least amount of SNPs.

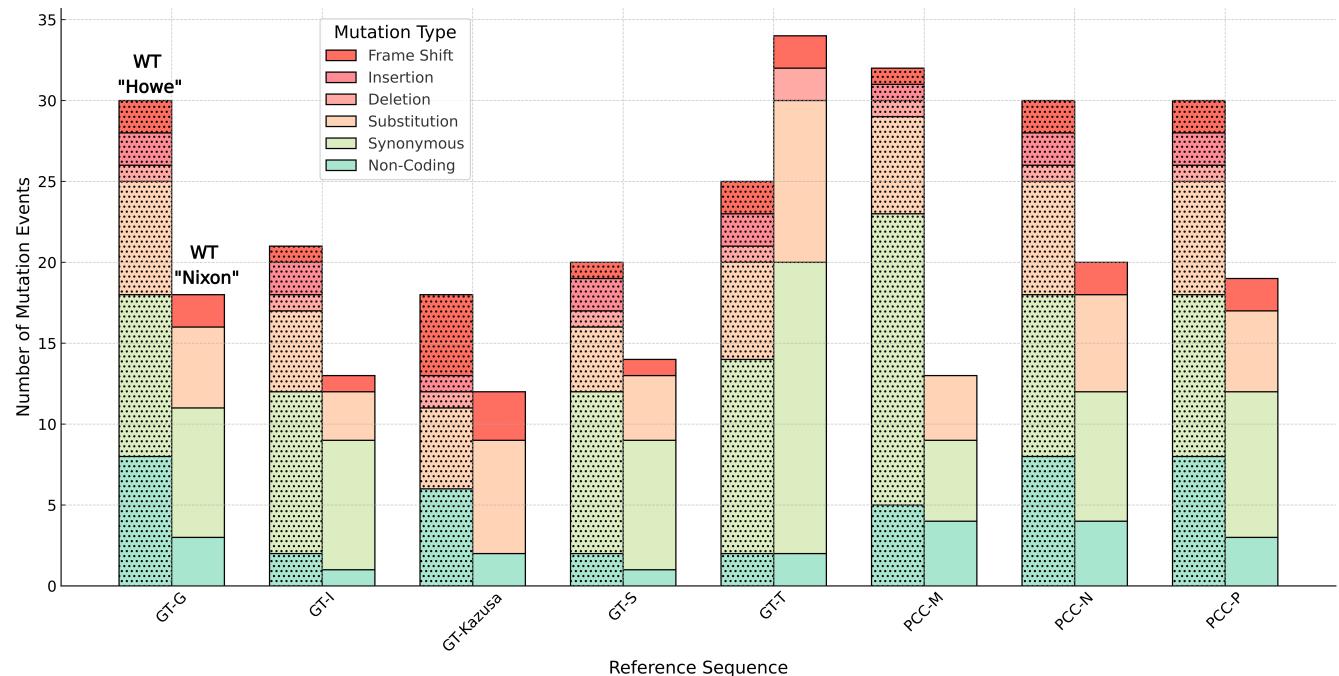


Figure 7: Overview of the variant analysis results.

The mutations were then filtered to identify only non-synonymous ones which results in mutated proteins.

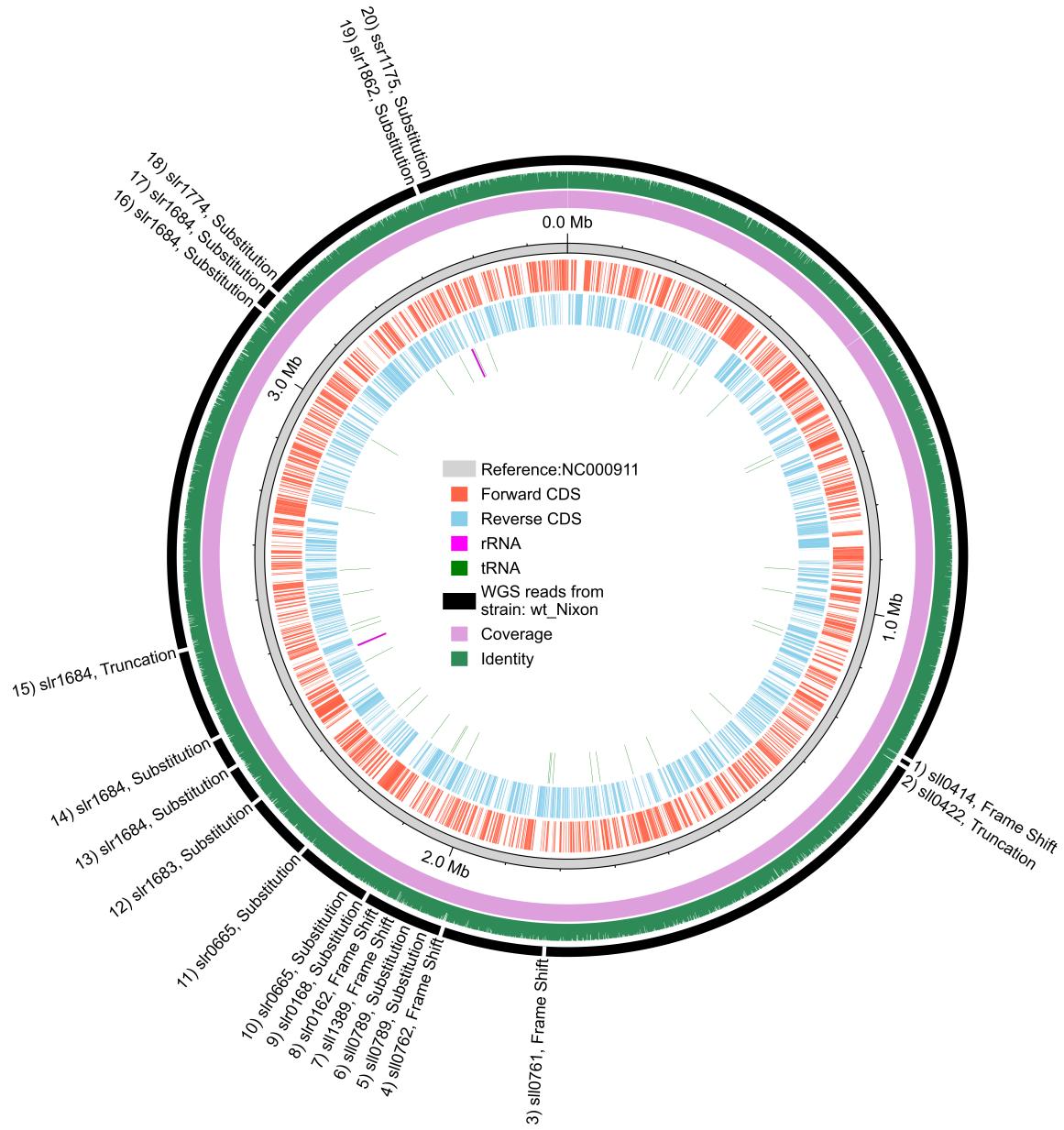


Figure 8: Trimmed and paired reads from whole genome sequencing experiments of *Synechocystis* "Nixon" wild-type aligned to the reference GT-"Kazusa" strain (NC000911). Numbered are mutation events found within coding sequences of the reference genome.

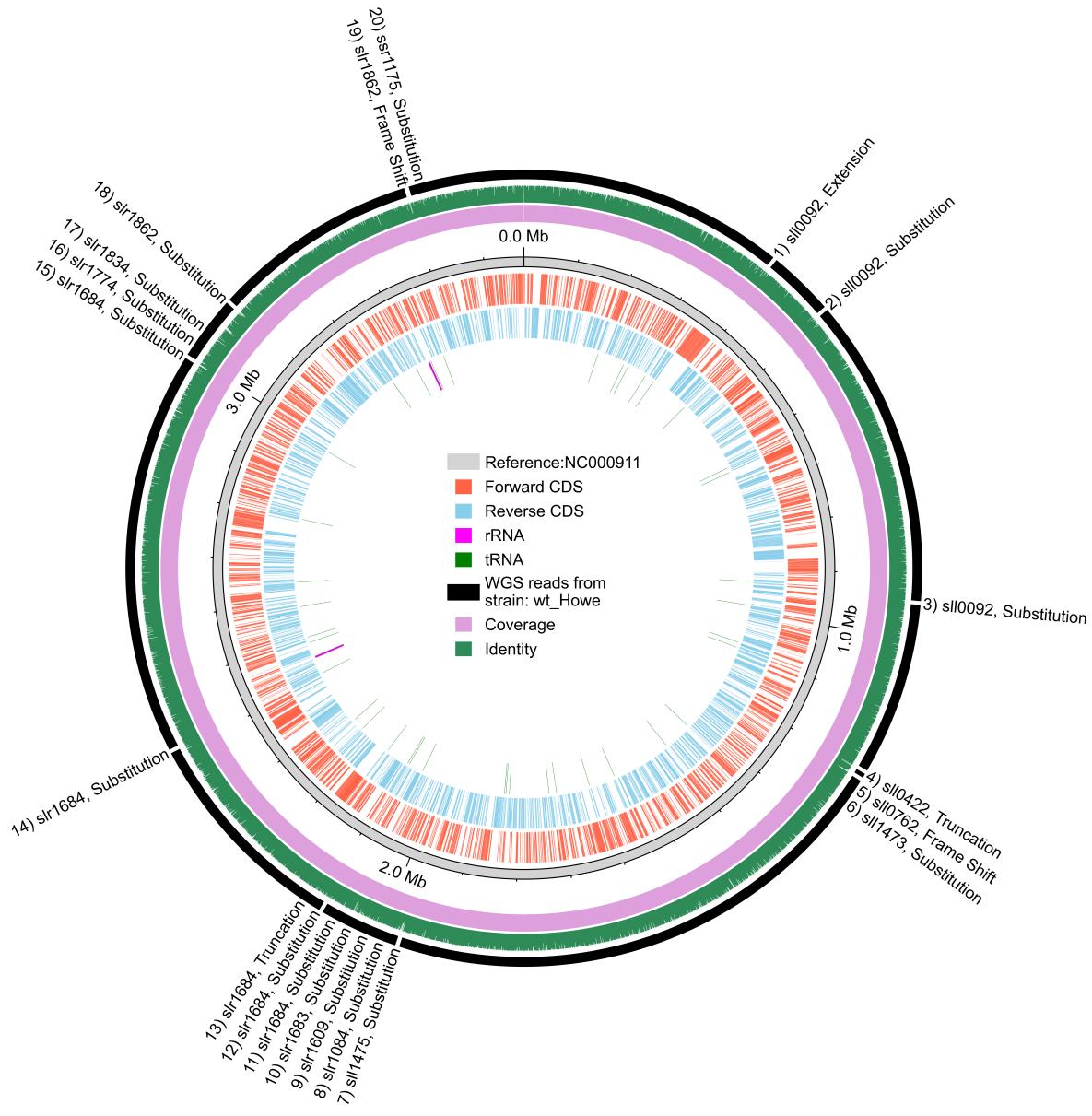


Figure 9: Trimmed and paired reads from whole genome sequencing experiments of *Synechocystis* "Howe" wild-type aligned to reference "Kazusa" strain (NC000911). Numbered are mutation events found within coding sequences of the reference genome.

Gene	Product	CDS Position	Change	Protein Effect	Amino Acid Change	Variant Frequency	Reference
1	ATP-dependent Clp protease ATP-binding subunit	580	T -> G	Substitution	K -> Q	0.997	GT-T
2	citrate synthase	434	A -> C	Substitution	M -> R	1	PCC-M
3	DUF4335 domain-containing protein	334	+C	Frame Shift		0.955	GT-Kazusa
4	hypothetical protein	302	-C	Frame Shift		0.952	GT-Kazusa
5	Ig-like domain-containing protein	8924	+C	Frame Shift		0.957	GT-Kazusa
6	sugar porter family MFS transporter	85	(C)8 -> (C)7	Frame Shift		0.919	GT-T
7	two-component system regulator RppA	403	CCA -> TCG	Substitution	W -> R	0.517 + 0.003	GT-Kazusa, GT-I, GT-S, GT-T, PCC-N, PCC-P, GT-G
8	two-component system regulator RppA	397	TTG -> GTC	Substitution	Q -> D	0.550 + 0.0005	GT-Kazusa, PCC-P, PCC-N, GT-T, GT-S, GT-I, GT-G
9	sensor histidine kinase RppB	1339	A -> C	Substitution	F -> V	0.501	PCC-M
10	sensor histidine kinase RppB	1336	T -> G	Substitution	I -> L	0.516	PCC-M
11	Rpn family nuclease/putative transposase	340	C -> A	Substitution	V -> L	0.509	PCC-M
12	hypothetical protein	430	(G)8 -> (G)7	Frame Shift		0.89 + 0.012	GT-Kazusa, PCC-P, PCC-N, PCC-M, GT-T, GT-S, GT-I, GT-G
13	serine/threonine-protein kinase	308	+T	Frame Shift		0.972 + 0	PCC-N, PCC-P
14	NAD(P)H-quinone oxidoreductase subunit F	1766	C -> T	Substitution	R -> Q	0.997	GT-T
15	EAL domain-containing protein	1134	(T)3 -> (T)2	Frame Shift		0.965	GT-G
16	anti-sigma regulatory factor	124	T -> C	Substitution	K -> E	1	GT-I
17	type II secretion system F protein	420	(G)9 -> (G)8	Frame Shift		0.835	GT-Kazusa
18	DUF4114 domain-containing protein	1207	A -> G	Substitution	K -> E	1	GT-Kazusa
19	solanesyl diphosphate synthase	205	+ACGGCG	Insertion	-> TA	0.554 + 0	GT-G, GT-I, GT-S, GT-T, PCC-N, PCC-P
20	bifunctional aconitate hydratase	31	G -> A	Substitution	D -> N	0.515 + 0	GT-T, GT-S, GT-I, GT-G
21	bifunctional aconitate hydratase	26	T -> C	Substitution	V -> A	0.518 + 0	GT-Kazusa, PCC-P, PCC-N, GT-T, GT-S, GT-I, GT-G
22	IS5 family transposase	443	+CATGGA	Insertion	G -> VHG	0.836 + 0.01	GT-Kazusa, PCC-P, PCC-N, PCC-M, GT-T, GT-S, GT-I, GT-G
23	glycosyltransferase family 4 protein	67	-GAACT GTCATC	Deletion	ELSI ->	0.74 + 0.043	GT-Kazusa, PCC-P, PCC-N, PCC-M, GT-T, GT-S, GT-I, GT-G
24	PP2C family protein-serine/threonine phosphatase	25	A -> T	Substitution	S -> C	1 + 0	GT-G, PCC-N, PCC-P
25	PP2C family protein-serine/threonine phosphatase	27	CTT -> AAA	Substitution	SL -> RK	0.645 + 0.005	GT-G
26	PP2C family protein-serine/threonine phosphatase	42	T -> A	Substitution	D -> E	0.6 + 0	GT-G, PCC-N, PCC-P
27	acylneuraminate cytidyltransferase	318	C -> G	Substitution	S -> R	1	PCC-M
28	nucleotidyltransferase family protein	74	T -> C	Substitution	Q -> R	0.534	PCC-M

Table 1: Unique mutations found in WT "Nixon"

Gene	Product	CDS Position	Change	Protein Effect	Amino Acid Change	Variant Frequency	Reference
1	transposase	232	A -> G	Substitution	Y -> H	0.557 + 0	GT-Kazusa, PCC-P, PCC-N, GT-T, GT-S, GT-I, GT-G
2	transposase	205	G -> A	Substitution	P -> S	0.523 + 0	GT-Kazusa, GT-G, GT-I
3	transposase	269	T -> C	Extension		0.614 + 0	GT-Kazusa, PCC-P, PCC-N, GT-T, GT-S, GT-I, GT-G
4	hypothetical protein	302	-C	Frame Shift		0.928	GT-Kazusa
5	PAS domain S-box protein	1397	G -> A	Substitution	A -> V	0.516	GT-Kazusa
6	ATP-binding protein	15	A -> T	Substitution	D -> E	0.5	GT-Kazusa
7	type IV pilus twitching motility protein PilT	674	-TGTGTA	Deletion	INK -> K	0.83	GT-T
8	Rpn family nuclease/putative transposase	340	TAA -> CAC	Substitution	L -> V	0.552	PCC-M
9	solanesyl diphosphate synthase	205	+CGGCG	Frame Shift		0.503 + 0	GT-G, GT-I, GT-S, GT-T, PCC-N, PCC-P
10	UPF0175 family protein	200	C -> A	Truncation		0.508	PCC-M
11	hypothetical protein	230	T -> A	Substitution	V -> D	0.547 + 0	GT-Kazusa, GT-S, GT-T
12	type IV pilus assembly protein PilM	266	T -> A	Substitution	V -> E	1	GT-T
13	phosphate acyltransferase PlsX	263	G -> T	Substitution	G -> V	0.998	PCC-N
14	AMP-binding protein	764	T -> G	Substitution	F -> C	1 + 0	GT-Kazusa, PCC-P, PCC-N, GT-T, GT-S, GT-I, GT-G
15	Rpn family nuclease/putative transposase	559	CTG -> ATT	Substitution	L -> I	0.611	PCC-M
16	Rpn family nuclease/putative transposase	550	G -> T	Substitution	A -> S	0.641	PCC-M
17	CHAT domain-containing protein	3013	A -> C	Substitution	T -> P	0.521	PCC-M
18	photosystem I core protein PsbA	1810	G -> A	Substitution	V -> I	0.994	GT-Kazusa
19	hypothetical protein	454	+C	Frame Shift		0.538	GT-Kazusa
20	PP2C family protein-serine/threonine phosphatase	25	AGC -> TGG	Substitution	S -> W	0.528 + 0.002	GT-G, PCC-N, PCC-P
21	PP2C family protein-serine/threonine phosphatase	29	T -> A	Truncation		0.543 + 0	GT-G, PCC-N, PCC-P
22	PP2C family protein-serine/threonine phosphatase	42	T -> CGA	Frame Shift		0.51 + 0	GT-G, PCC-N, PCC-P
23	transposase	205	G -> A	Substitution	P -> S	0.523 + 0	GT-S, GT-T, PCC-N, PCC-P

Table 2: Unique mutations found in WT "Howe"

6.2 Sequenced strains compared to various substrains

To visualise the SNPs similarity between various strains, a principal component analysis (PCA) was performed, where the feature matrix was constructed to represent each strain as a row and each reference genome as a column. The entries in this matrix were the total SNP counts for each strain against each reference genome. Thus, the PCA effectively reduced the multidimensional space spanned by the different reference genomes into a lower-dimensional representation. The PCA plot positions each strain based on its SNP profile across different reference genomes. Strains that are genetically similar in terms of SNP counts against multiple reference genomes would appear closer in the reduced-dimensional space.

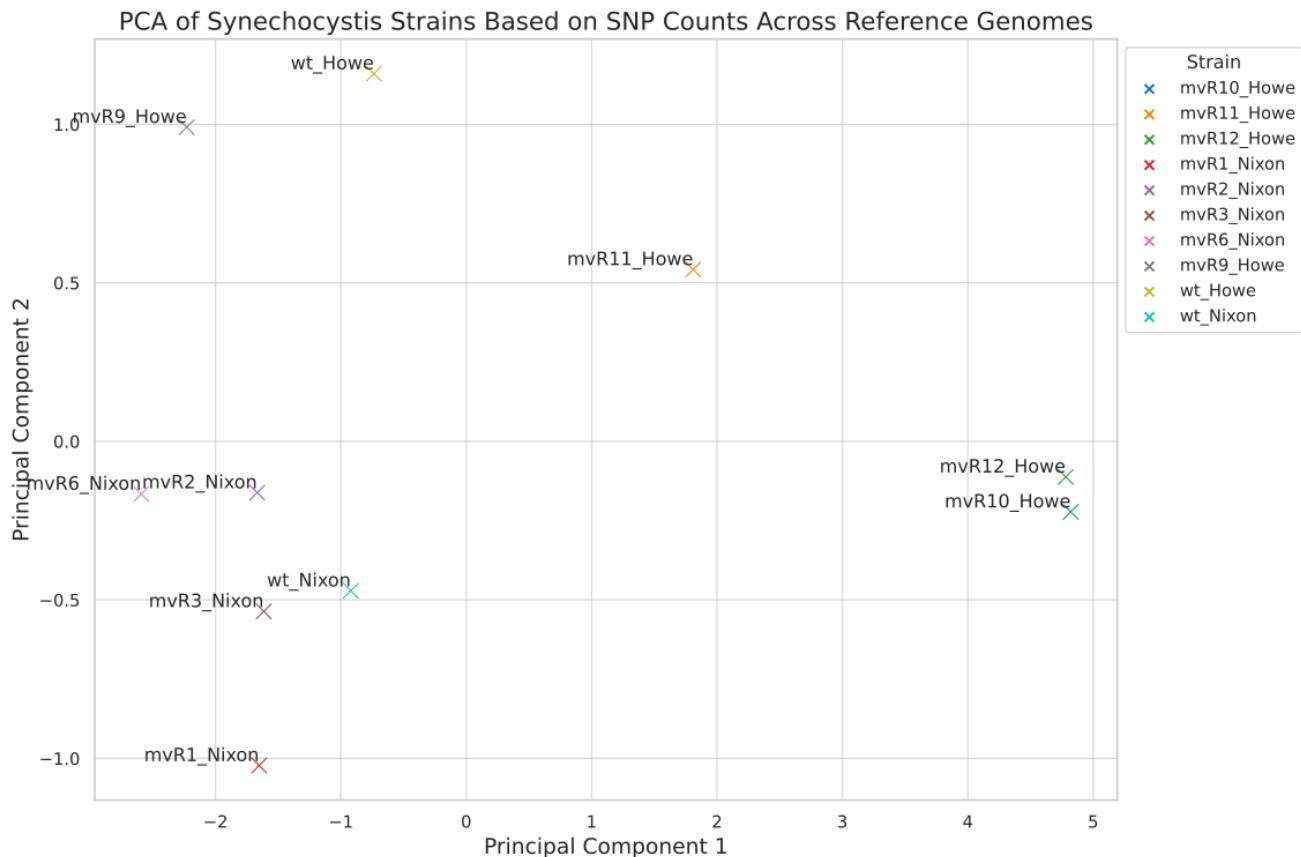


Figure 10: Principal component analysis based on SNPs count of the sequenced genomes

Principal Component 1 (PC1) captured approximately 88.22% of the total variance in the dataset. Principal Component 2 (PC2) accounted for about 5.37% of the total variance. Cumulative Explained Variance: Together, the first two principal components capture approximately 93.58% of the total variance. The first principal component (PC1) alone captures a significant proportion of the dataset's variance, indicating an effective reduction in dimensionality. The high cumulative explained variance (93.58%) implies that the first two principal components provide a statistically robust representation of the dataset's original variability. Therefore PCA analysis indicated that strains with the "Howe" and "Nixon" suffixes tend to cluster with their respective wild-types, corroborating the genomic closeness between mutants and their parental strains.

6.3 SNPs Polimorphism

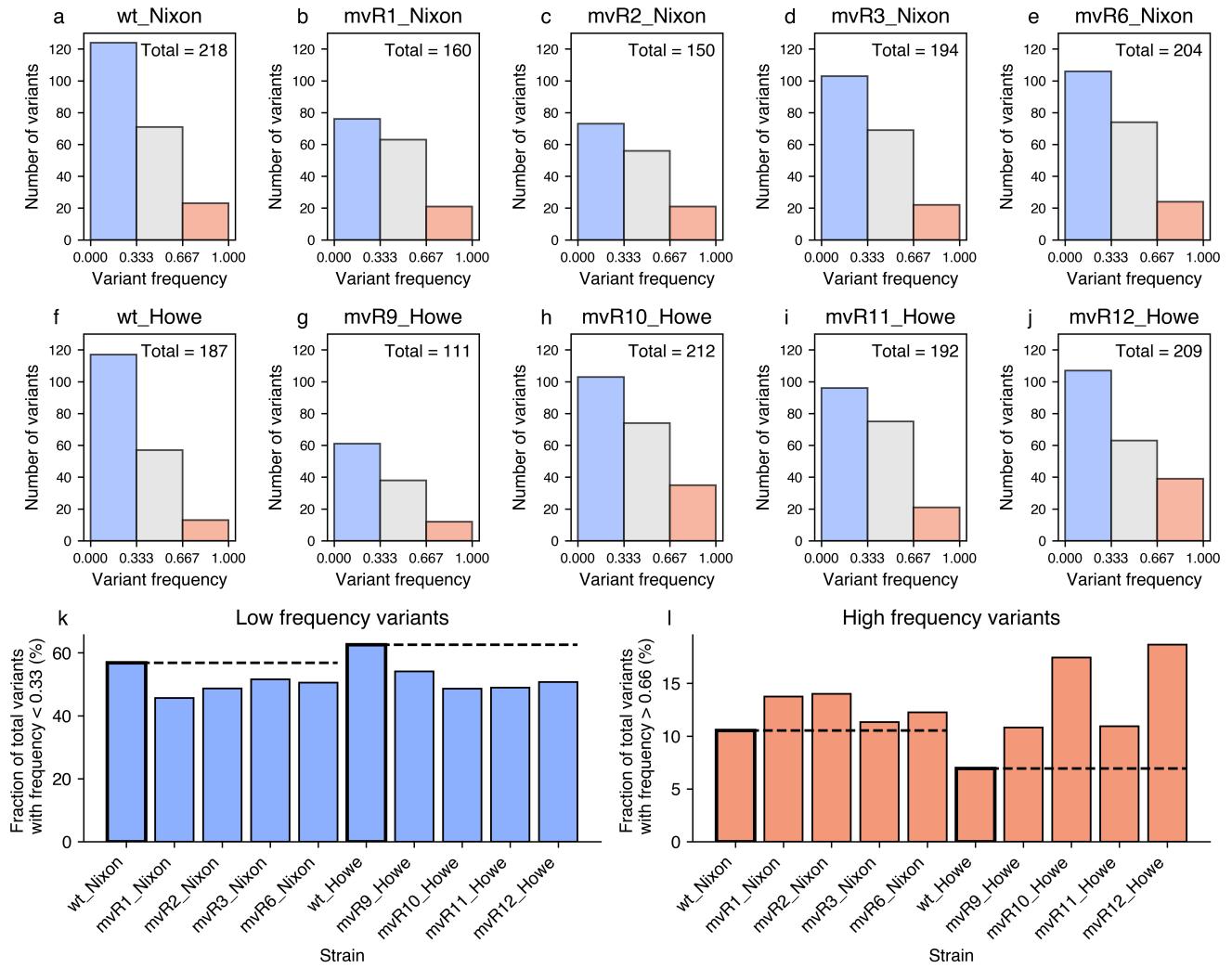


Figure 11: 3 bins, 0.3 and 0.6 as thresholds

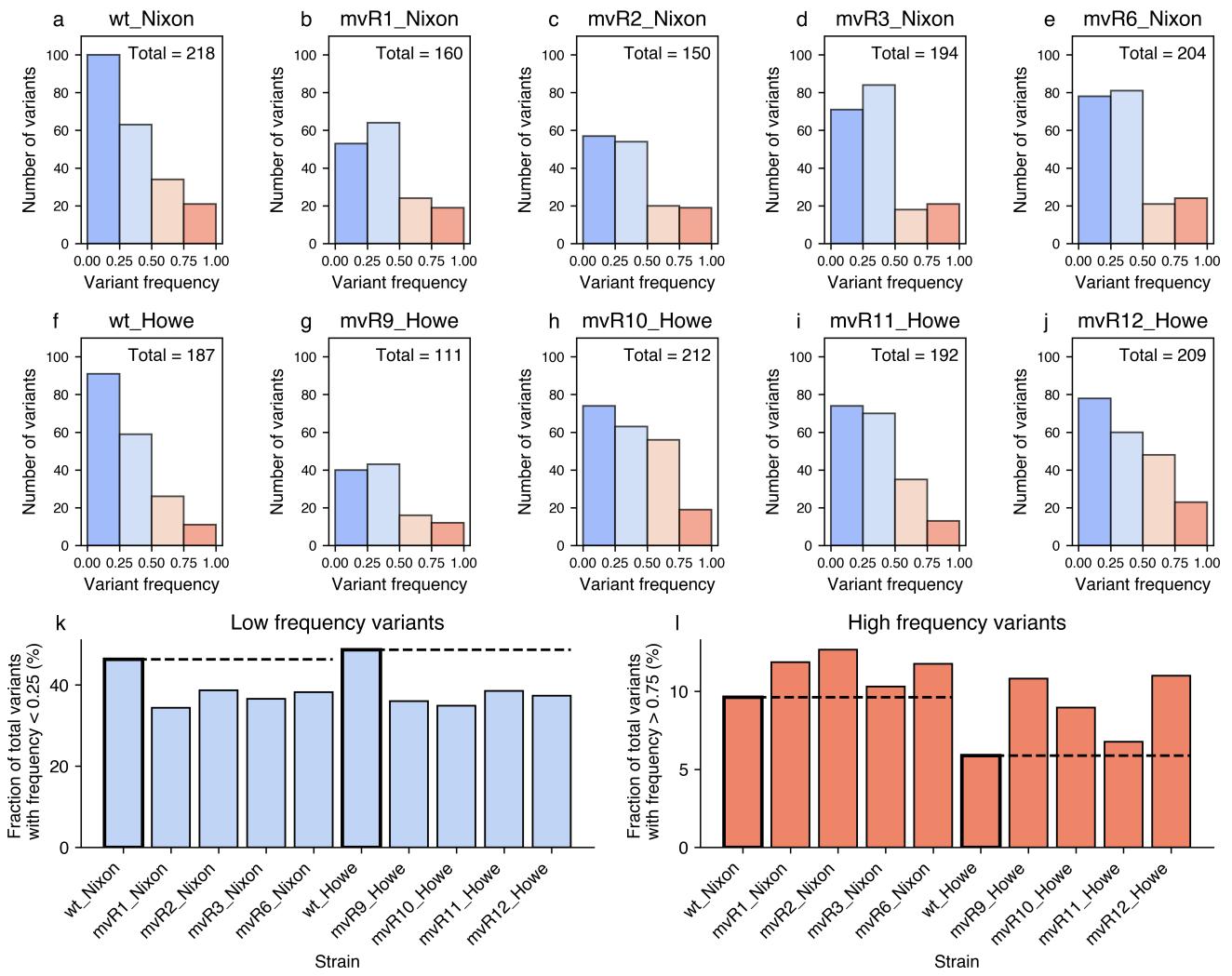


Figure 12: 4 bins, 0.25 and 0.75 as thresholds

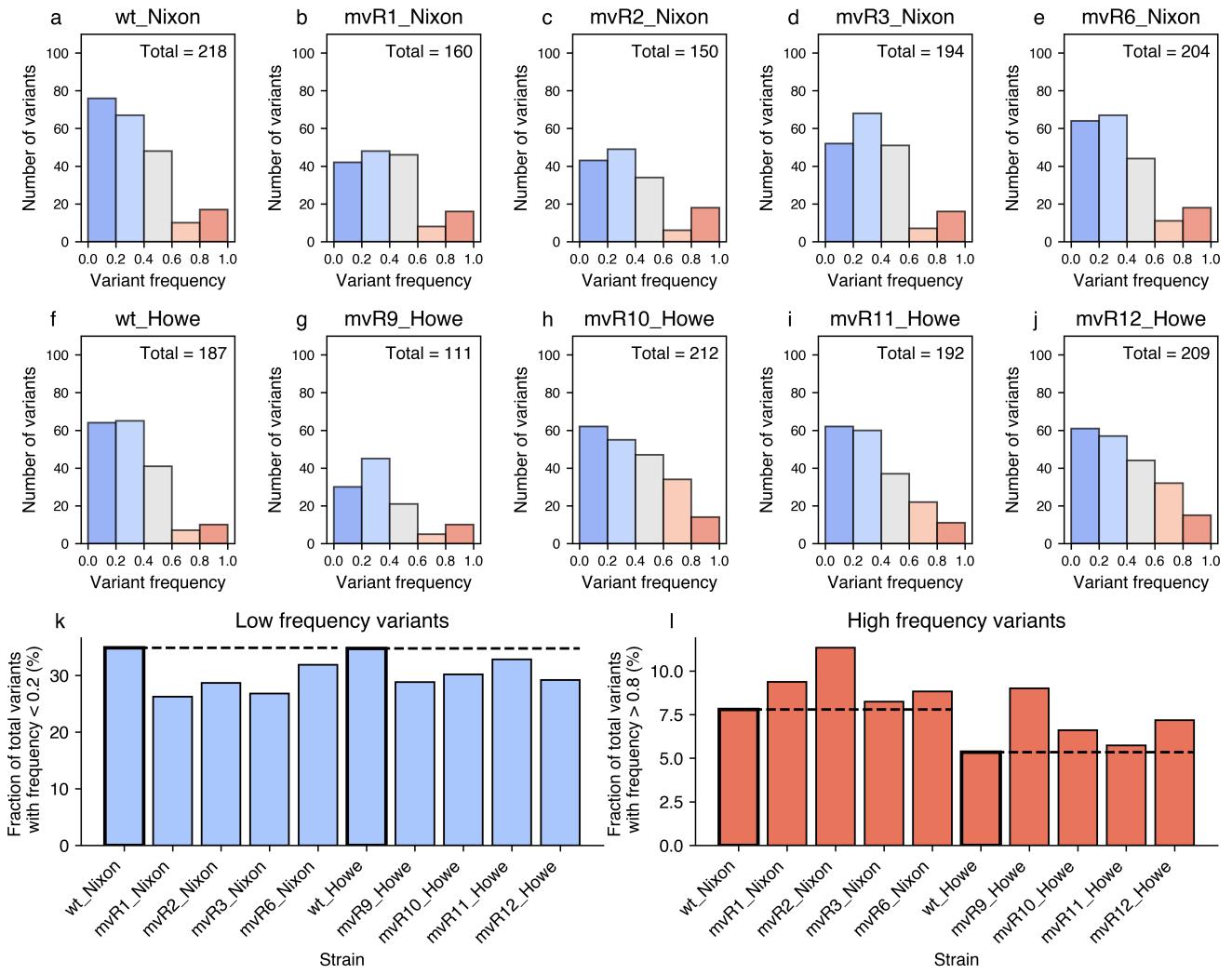


Figure 13: 5 bins, 0.2 and 0.8 as thresholds

7 Extended methods

7.1 Buffers and media

7.1.1 0.25 M EDTA stock solution - pH8

To prepare 100 ml of 0.25 M EDTA stock solution at pH = 8 (needed for preparing BG11):

1. Weigh 9.305 g of Na₂EDTA (FW 372.2)
2. Dissolve in ≈80 mL of DI water
3. Adjust pH to 8 with concentrated NaOH
4. Top up to final volume (100 mL) with deionised water
5. Autoclave

7.1.2 1M Tris/HCl buffer - pH8

To obtain 100 ml a 1M Tris/HCl buffer (required to prepare the pre-lysis buffer “Smoker B” for gDNA extraction):

1. Weigh 12.11 g of Tris
2. Dissolve in ≈80 ml of deionised water
3. Mix with magnetic stirrer until fully dissolved
4. Adjust pH to 8 with concentrated HCl
5. Top up to final volume (100 mL) with deionised water
6. Autoclave

7.1.3 Lysis buffer for gDNA extraction (“Smoker B”)

Smoker B (SB) buffer was used in the pre-lysis step for genomic DNA extraction of cyanobacteria (tested for *Synechocystis* and *Chroococcidiopsis*). It contains lyophilised lysozyme powder from chicken egg whites for cell lysis, and RNase and proteinase K to degrade unwanted RNA and protein materials, respectively. Lysozyme powder, RNase and proteinase K were kept in the freezer (-20°C). To prepare the buffer, the components listed in Table 3 were added in a 15 mL Falcon tube and vortex until fully dissolved. For genomic extraction, each sample requires 170 µl of Smoker B buffer. The solution was not autoclaved. Previously autoclaved water and stocks were used to prepare the solution fresh on the day of the experiment.

Ingredient	[Working]	[Stock]	Amount to add (Final volume = 10 mL)
1M Tris/HCl buffer (pH8)	50 mM	1000 mM	500 µl
0.5M EDTA stock solution	50 mM	500 mM	1 ml
Triton X-100	1 % (v/v)	100 % (v/v)	100 µl
Lysozyme (lyophilized powder)	800,000 (units/ml)	>=40,000 units/mg powder	200 mg
RNase	0.01 mg/ml	10	10 mg
DIH ₂ O	to final volume	N.A.	8.39 ml

Table 3: Ingredients and their working concentrations for preparing the buffer “Smoker B”

7.1.4 Photoautotrophic growth medium and plates (BG11)

The standard cyanobacterial photoautotrophic medium (BG11) was prepared according to a protocol adapted from (?). The medium comprises various stocks that are prepared separately, autoclaved for sterility, and subsequently combined to achieve the desired working concentrations.

To prepare 1 L of liquid medium:

1. Prepare all the stock solutions (except NaOH-TES) listed in Table 4 in Duran bottles of appropriate sizes
2. Autoclave all the stock solutions to minimise contaminations during long-term storage
3. Add \approx 100 mL of DI water into a 1L Duran bottle
4. Pour in all the stock solutions (except sodium bicarbonate and NaOH-TES buffer), in the order and with the volumes listed in Table 4
5. Top up to \approx 900 mL with additional DI water
6. Submerge a pH probe in the solution, add a stirring bar inside the bottle, and place it on a magnetic stirrer mixing continuously. Adjust pH with concentrated HCl until pH = 7.8
7. Autoclave
8. After the solution has cooled down to room temperature, add 10 mL of *previously autoclaved* bicarbonate stock solution
9. Top up to 1L with previously autoclaved DI water
10. Store at room temperature (if no antibiotics inside) and away from direct sunlight

To prepare solid BG11 plates: make 500 mL of 2X BG11 stock solution, 500 mL of 3% agar solution and then mix them to obtain a 1X-BG11 + 1.5% agar solution as detailed below:

1. Prepare and autoclave all the stock solutions as for the liquid medium with the addition of NaOH-TES buffer.
2. Add \approx 100 mL of DI water into a 1L Duran bottle
3. Pour in all the stock solutions (except sodium bicarbonate), in the order and with the volumes listed in Table 4. To prepare a 2X BG11 solution, use the same volumes of stocks as listed for 1X-BG11 but top up to half of the final 1X-BG11 volume (500 mL)
4. Top up to \approx 400 mL with DI water
5. Adjusting pH is not required for solid BG11 (as it contains NaOH-TES buffer)
6. Prepare 500 mL of 3% agar solution (2X) by dissolving 15g of agar in 500 mL of DI water in a 500 mL Duran bottle
7. Autoclave the 2X BG11 and agar solutions
8. Add 10 mL of previously autoclaved bicarbonate stock in the bottle containing 2X-BG11
9. Top up to precisely 500 mL with previously autoclaved DI water

10. Melt the agar solution (with a microwave), pour 500 mL of melted agar into the 1L Duran bottle containing the 2X BG11 solution. **To prevent explosions:** only add the melted agar solution into the 2X-BG11-containing bottle and not vice versa.
11. Mix the solution and pour into Petri dishes when still hot. If antibiotics are required, only add them when the solution reaches \approx 50°C.
12. Wait until the plates solidify and store (up to 3 months) in a dark fridge.

Stock	Component	[Stock]	[Working]	Amount	Units	V _{final}
100X-BG11 (100X)	NaNO ₃ MgSO ₄ ·7H ₂ O CaCl ₂ Citric Acid EDTA stock (pH 8.0)	1.76 M 30 mM 32.4 mM 3.1 mM 250 mM	17.6 mM 0.3 mM 0.324 mM 31 μM 2.5 mM	149.6 7.49 3.6 0.6 1.12	g g g g mL	1L
sodium bicarbonate (100X)	Na ₂ HCO ₃	1 M	10 mM	84.01	g	1L
Trace elements (1000X)	H ₃ BO ₃ MnCl ₂ · 4H ₂ O ZnSO ₄ · 7H ₂ O Na ₂ MoO ₄ · 2H ₂ O CuSO ₄ · 5H ₂ O Co(NO ₃) ₂ · 6H ₂ O	46 mM 9.1 mM 765 μM 1.61 mM 320.4 μM 172 μM	46 μM 9.1 μM 765 pM 1.61 μM 320.4 pM 172 pM	286 181 22 39 8 5	mg mg mg mg mg mg	100 mL
Phosphate buffer (1000X)	K ₂ HPO ₄	175.1 mM	0.175 mM	3.05	g	100 mL
sodium carbonate (1000X)	Na ₂ CO ₃	188.7 mM	0.189 mM	2	g	100 mL
Ferric citrate buffer (1000X)	C ₆ H ₅ FeO ₇	24.5 mM	0.245 mM	0.6	g	100 mL
TES-NaOH pH8.2 (100X)	TES free acid NaOH	1 M 1 M	10 mM add until pH 8.2 is obtained	90	g	1L
BG11 (1X) BG11 (2X)	100X-BG11 stock carbonate stock phosphate buffer trace elements stock ferric citrate stock TES-NaOH buffer ² bicarbonate stock ³	100X 1000X 1000X 1000X 1000X 1000X 100X	1X 1X 1X 1X 1X 1X 1X	10 1 1 1 1 10 10	mL mL mL mL mL mL mL	1L (liquid) 0.5L (solid) ¹

Table 4: Recipe for preparing BG11 medium. Stocks are prepared and autoclaved separately, and then added (with the exception of sodium bicarbonate) into a 1L Duran bottle to prepare 1X BG11. Adjust the pH to 7.8 with concentrated HCl when preparing liquid BG11 (1X) medium.

¹Top up with previously autoclaved DI water to a final volume of 1L for 1X-BG11 liquid medium or 500 mL for 2X-BG11 solution for making solid plates.

²NaOH-TES buffer is only added when preparing solid BG11 plates.

³Sodium bicarbonate stock solution is sterilised separately and added after autoclaving the BG11-containing 1L Duran bottle.

8 Code

8.1 Comare variants across strains

```
1 import pandas as pd
2 from Bio import SeqIO
3 import numpy as np
4 from pycirclize import Circos
5 from pycirclize.parser import Gff
6 from pycirclize.utils import ColorCycler
7 import matplotlib.pyplot as plt
8 from matplotlib.patches import Patch
9 from matplotlib.lines import Line2D
10 import os
11 import re
12
13 def wrap_text(text, max_length):
14     """
15     Wrap the text every max_length characters.
16     Parameters:
17         text (str): The text to wrap.
18         max_length (int): The maximum line length.
19     Returns:
20         wrapped_text (str): The wrapped text.
21     """
22     return '\n'.join(text[i:i+max_length] for i in range(0, len(text), max_length))
23
24 def min_percentage(s):
25     """Return the smallest percentage value in a string."""
26     if pd.isna(s):
27         return 0
28     if not isinstance(s, str):
29         s = str(s)
30     # Extract all values from the string
31     values = re.findall(r"(\d+(?:\.\d+)?)%", s)
32
33     # Convert to float, and if the value was a percentage, divide by 100
34     floats = [float(x.rstrip('%')) if '%' in x else float(x) * 100 for x in values]
35
36     return min(floats) if floats else 0
37
38 def adjust_positions(positions, min_distance):
39     """
40     Adjust positions to ensure a minimum distance between each position so
41     that mutated gene labels do not overlap
42
43     Parameters:
44         positions (list): List of positions, sorted in ascending order
45         min_distance (int): Minimum distance that should
46             be between each position
47
48     Returns:
49         new_positions (list): List of adjusted positions
50     """
```

```

51     new_positions = positions.copy()
52     for i in range(len(new_positions) - 1):
53         # Check if distance to next position is less than min_distance
54         if new_positions[i + 1] - new_positions[i] < min_distance:
55             # Adjust next position
56             new_positions[i + 1] = new_positions[i] + min_distance
57
58     # need to iterate until no more adjustments are needed,
59     # because an adjustment can cause a position to encroach on its next neighbor.
60     # If any adjustment is made, then the loop is run again.
61     if new_positions != positions:
62         new_positions = adjust_positions(new_positions, min_distance)
63
64     return new_positions
65
66 def plot_mutation_circos(ref):
67     # Load mutation data
68     all_data = pd.read_csv(f"../Data/variant_analysis/{ref}_filtered_mutations.csv"
69                           )
70     # Load the .gff file
71     gbk = Gff(f"../Data/genome_annotations/{ref}.gff")
72     # Initialize the Circos plot
73     circos = Circos(sectors={gbk.name: gbk.range_size})
74     sector = circos.get_sector(gbk.name)
75
76     # Plot outer track with xticks
77     major_ticks_interval = 1000000
78     minor_ticks_interval = 100000
79     outer_track = sector.add_track((75, 77))
80     outer_track.axis(fc="lightgrey")
81     outer_track.xticks_by_interval(
82         major_ticks_interval, label_formatter=lambda v: f"{v/10**6:.1f} Mb"
83     )
84     outer_track.xticks_by_interval(minor_ticks_interval, tick_length=1, show_label=
85                                     False)
86     # Extract unique strains from the first dataset
87     all_strains = all_data['Strain'].unique()
88     print(all_strains)
89     # Define colors for each strain
90     colors = ['tomato', 'skyblue', 'magenta', 'green', 'purple', 'brown', 'orange',
91               'lightgreen', 'cadetblue']
92     strain_color = {strain: color for strain, color in zip(all_strains, colors)}
93
94     # Group mutations by 'CDS' and select the first row from each group
95     # grouped_mutations = all_data.groupby('product').first().reset_index()
96     grouped_mutations = all_data[all_data['Protein_Effect'].notna() & all_data['
97                                   Protein_Effect'].ne('')].groupby(['locus_tag', 'Minimum', 'Change']).
98                                   first().reset_index()
99     # Create a mapping of strains and frequencies per mutation
100    # mutation_strain_map = all_data.groupby(['locus_tag', 'Minimum', 'Change']).
101        agg({'Strain': lambda x: ', '.join(x), 'Variant Frequency': lambda x: ', '
102              '.join(x)}).to_dict(orient='index')
103
104    mutation_strain_map = all_data.groupby(['locus_tag', 'Minimum', 'Change']).agg
105        ({{

```

```

98     'Strain': lambda x: ','.join(x),
99     'Variant_Frequency': lambda x: ','.join(map(str, x))
100 }).to_dict(orient='index')
101
102 # Group by 'locus_tag' and get the first entry for each unique gene
103 unique_grouped_mutations = grouped_mutations.groupby('locus_tag').first()
104
105 labels = [f"{row['product']}"] for _, row in unique_grouped_mutations.iterrows()
106     ↪ ]
107 pos_list = unique_grouped_mutations["Minimum"].values.tolist()
108 # Initialize new_labels and pos_list_CDSonly as empty lists
109 new_labels = []
110 pos_list_CDSonly = []
111 # Step 1: Define mutation_details before the loop
112 mutation_details = []
113 for i, (index, row) in enumerate(grouped_mutations.iterrows(), start=1):
114     freq = min_percentage(row['Variant_Frequency'])
115     if freq >= 75:
116         label = f"{i} {row['locus_tag']}" if row['Protein_Effect'] not in [
117             ↪ None, '' ] else ''
118         label_table = f"{i} {row['locus_tag']}"
119         new_labels.append(label)
120         pos_list_CDSonly.append(row['Minimum']) # Add this position to
121             ↪ pos_list_CDSonly
122         # Append mutation details to the dataframe
123         # Step 2: Append mutation details to mutation_details
124         amino_acid_change = row['Amino_Acid_Change']
125         if not amino_acid_change:
126             amino_acid_change = "fs"
127         mutation_strain_freq = mutation_strain_map.get((row['locus_tag'], row[
128             ↪ 'Minimum'], row['Change']), {})
129         mutation_details.append({
130             "Mutation": label_table,
131             "Gene": row['product'],
132             "Position": row['CDS_Position'],
133             "Change": amino_acid_change,
134             "Strain": wrap_text(mutation_strain_freq.get('Strain', ''), 15),
135             "Variant_Frequency": wrap_text(mutation_strain_freq.get('Variant_'
136                 ↪ Frequency', ''), 15)
137         })
138     else:
139         new_labels.append('')
140         pos_list_CDSonly.append(row['Minimum']) # Add this position to
141             ↪ pos_list_CDSonly
142 # Plot a track for each strain
143 for i, strain in enumerate(all_strains):
144     strain_data = all_data[all_data['Strain'] == strain]
145     radial_range = (69 - i*5, 73 - i*5)
146     strain_track = sector.add_track(radial_range, r_pad_ratio=0.1)
147     strain_track.axis(fc=strain_color[strain], alpha=0.4)
148     for index, mutation in strain_data.iterrows():
149         strain_track.rect(mutation['Minimum']-0, mutation['Minimum']+1000, fc="
150             ↪ white", ec=strain_color[strain], alpha=0.8, lw=1)
151         #strain_track.scatter([mutation['Minimum']
152             ↪ ], [50], color="black", s=10, marker=".", linewidths =0.5, ec="black"

```

```

146         ↪ ")
# Add the outermost track with labels for mutations
147     label_track = sector.add_track((82, 85)) # Adjust the radial range as needed
148
149 # Use new_labels instead of labels
150 pos_list = grouped_mutations["Minimum"].values.tolist()
151 # Plot outer xticks (labels)
152 label_track.xticks(
153     pos_list,
154     new_labels,
155     outer=True, # Make sure labels are plotted outside
156     tick_length=0, # Set tick_length to 0 so ticks won't show up
157     label_margin=1,
158     label_orientation="vertical", # Set label orientation to vertical
159     label_size= 10,
160 )
161 # Save the Circos plot
162 fig_circos = circos.plotfig()
163 _ = circos.ax.legend(
164     handles=[
165         Patch(color="gray", label=f"{ref}", alpha=0.6, edgecolor="black"),
166         Patch(color="tomato", label="mvR1", alpha=0.6, edgecolor="black"),
167         Patch(color="skyblue", label="mvR2", alpha=0.6, edgecolor="black"),
168         Patch(color="magenta", label="mvR3", alpha=0.6, edgecolor="black"),
169         Patch(color="green", label="mvR6", alpha=0.6, edgecolor="black"),
170         Patch(color="purple", label="mvR9", alpha=0.6, edgecolor="black"),
171         Patch(color="brown", label="mvR10", alpha=0.6, edgecolor="black"),
172         Patch(color="orange", label="mvR11", alpha=0.6, edgecolor="black"),
173         Patch(color="lightgreen", label="mvR12", alpha=0.6, edgecolor="black"),
174     ],
175     bbox_to_anchor=(0.5, 0.5),
176     loc="center",
177     ncols=1,
178     fontsize=10,
179 )
180 fig_circos.savefig(f"../Figures/WGS/{ref}_mutants_vs_WT_genome_views_New3.svg",
181     ↪ dpi=600)
# Step 3: Use mutation_details to generate the table
182 headers = list(mutation_details[1].keys()) # Convert dict_keys to list
183 cell_text = [list(d.values()) for d in mutation_details] # Get cell values
184 fig = plt.figure(figsize=(15,10), dpi=300)
185 ax = plt.subplot()
# Determine maximum text length in each column and use it as width
186 column_widths = [
187     max(len(str(x)) for x in column) * 8 if header in ["Amino_Acid_Change", "
188         ↪ product"] else max(len(str(x)) for x in column) * 1
189     for column, header in zip(zip(*([headers] + cell_text)), headers) # Adding
190         ↪ header to cell_text for each column
191 ]
# Calculate positions based on column widths
192 positions = np.cumsum([0] + column_widths) # Here we include the rightmost
193     ↪ border
194 ncols = len(headers)
195 nrows = len(cell_text)
print(f"Mutations_found:{nrows}")

```

```

196 # Add 1 for headers and adjust limits according to column widths
197 ax.set_xlim(0, sum(column_widths))
198 ax.set_ylim(0, nrows + 1)
199 # Add table's main text
200 for i in range(nrows):
201     for j, text in enumerate(cell_text[i]):
202         ax.annotate(
203             xy=(positions[j] + 0.5 * column_widths[j], nrows - i - 0.5), #
204             # Adjust Y position for text to align it in the center of the
205             # bounding box
206             text=text,
207             ha='center',
208             va='center',
209             fontsize=10 # Make header text larger
210         )
211 # Add column names
212 for index, header in enumerate(headers):
213     ax.annotate(
214         xy=(positions[index] + 0.5 * column_widths[index], nrows + 0.5), #
215         # Adjust Y position for header text to align it in the center of
216         # the bounding box – for 20 mutation change 1 to 0.5
217         text=header,
218         ha='center',
219         va='center',
220         weight='heavy', # Make header text bold
221         fontsize=12 # Make header text larger
222     )
223 # Add dividing lines
224 ax.plot([ax.get_xlim()[0], ax.get_xlim()[1]], [nrows, nrows], lw=1.5, color='black',
225         marker='', zorder=4)
226 ax.plot([ax.get_xlim()[0], ax.get_xlim()[1]], [0, 0], lw=1.5, color='black',
227         marker='', zorder=4)
228 for x in range(1, nrows):
229     ax.plot([ax.get_xlim()[0], ax.get_xlim()[1]], [x, x], lw=1.15, color='gray',
230             ls='--', zorder=3, marker='')
231 # Add vertical grid lines but exclude the outermost ones
232 for j in range(1, ncols): # Excluding outermost vertical lines
233     ax.plot([positions[j], positions[j]], [0, nrows], color='gray', linewidth
234             =1.15, ls='--', zorder=3, marker='')
235 ax.set_axis_off()
236 plt.savefig(
237     f"..Data/..Figures/WGS/Table_Mutants2_{ref}.png",
238     dpi=900,
239     transparent=False,
240     bbox_inches='tight'
241 )
242 print(f"Table_SavedSuccessfully as: .. / Figures/WGS/Table2_{ref}.png")
243
244     return
245 if __name__ == "__main__":
246     import argparse
247     parser = argparse.ArgumentParser(description='Filter_mutations_and_plot_a_
248         circos_diagram.')
249     parser.add_argument('-ref', help='The_reference_genome.')
250     args = parser.parse_args()

```

242

```
plot_mutation_circos(args.ref)
```