Determining the impact of mitochondrial dysfunction on stem cell dynamics and proliferation within the colon

Volume II of II

Craig Stamp

BSc (Hons) MRes

This thesis is submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Wellcome Trust Centre for Mitochondrial Research

Institute of Neuroscience

Newcastle University

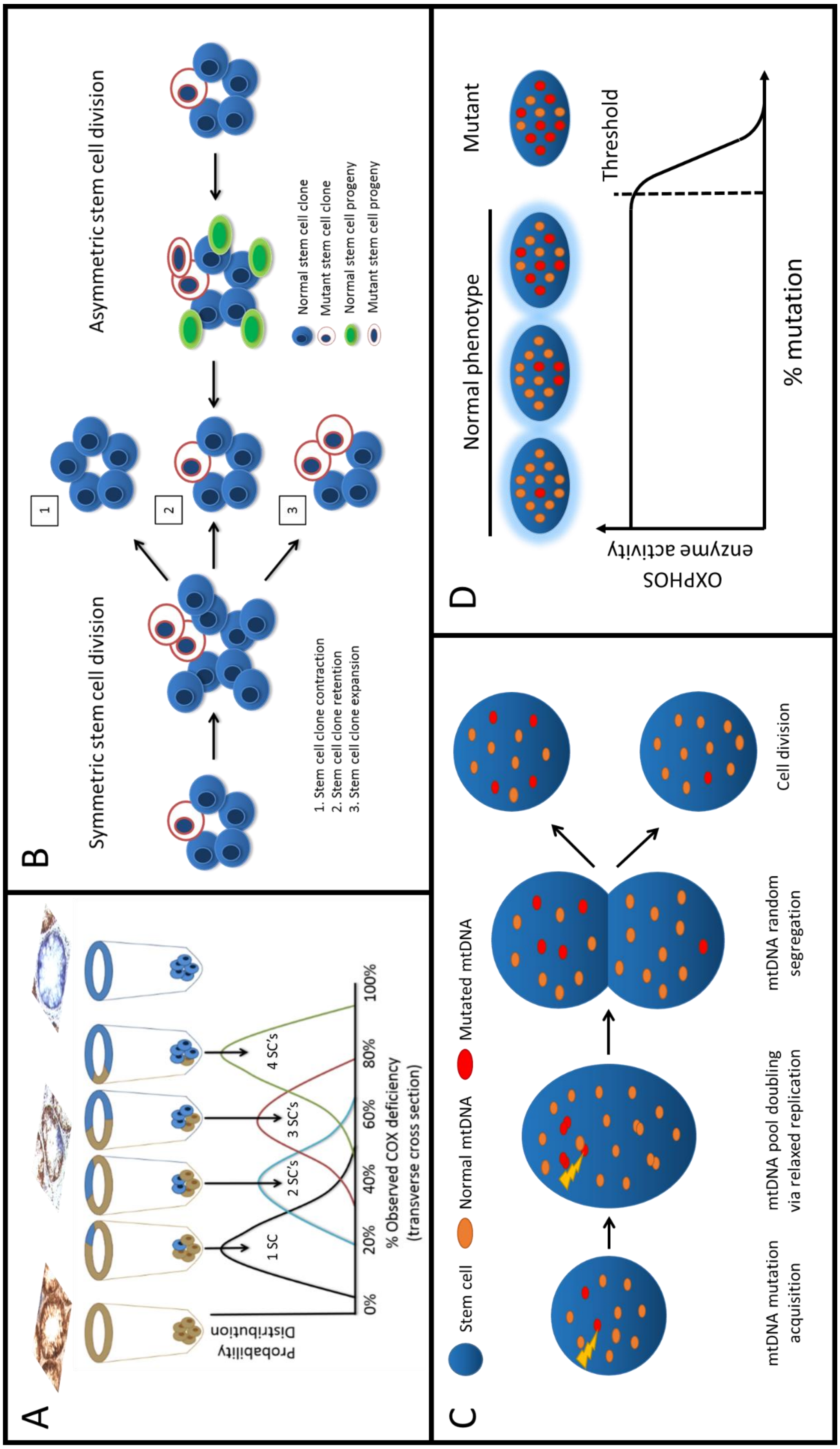
September 2015

Volume II Appendices

1.1. **HUMAN COLON NICHE SUCCESSION MODEL DIAGRAM**

Figure 1.1: Model overview

The model simulating COX deficient stem cell expansion and contraction within its niche is made up of several individual models that have to be simulated in conjunction with one another. (A) The biological data gathered is in the form of percentage COX deficiency for individual crypts whereas the model data is in the form of number of stem cells COX deficient. Therefore a model was developed that relates COX deficiency percentage of individual transverse crypts to number of COX deficient stem cells present within the niche via probability distributions, given the total number of stem cells present was specified (Section 2.2.10.3). This meant that the biological data and the simulated data were in the same form. (B) The first level of the model simulates the stem cell dynamics that occur within the stem cell niche of the crypt. Parameters involved include number of stem cells, number of time points (with each time point encompassing each stem cell dividing once), the probability of a stem cell undergoing symmetric stem cell division and asymmetric stem cell division, and also the probability of a stem becoming mutated upon division. Symmetric stem cell division can lead to stem cell clone contraction and expansion whereas asymmetric stem cell division can only lead to the same number of stem cells being COX deficient. (C) The latter models incorporated a more realistic simulation of the mutated mtDNA heteroplasmy which would determine the point at which a stem cell would become COX deficient. A stem cell would contain a fixed number of mtDNA molecules which would double according to relaxed replication, then undergo random segregation to produce two daughter cells. Mutated mtDNA is able to clonally expand or clonally contract via this mechanism. MtDNA molecules become mutated according to a parameterised mutation rate via random mutagenesis (ROS induced) and mutations incorporated during replication, as shown. The fate of each daughter cell is determined as in (B). Therefore each cell being simulated has a related heteroplasmy percentage. (D) As each cell has a heteroplasmy percentage, the state of the cell would be switched from normal to COX deficient once the parameterised threshold level has been reached.


A
Probability Distribution
1 SC
2 SC's
3 SC's
4 SC's
0%
20%
40%
60%
80%
100%
% Observed COX deficiency
(transverse cross section)
B
Symmetric stem cell division
1
2
3
1. Stem cell clone contraction
2. Stem cell clone retention
3. Stem cell clone expansion
Asymmetric stem cell division
Normal stem cell clone
Mutant stem cell clone
Normal stem cell progeny
Mutant stem cell progeny
C
Stem cell
Normal mtDNA
Mutated mtDNA
mtDNA mutation acquisition
mtDNA pool doubling via relaxed replication
mtDNA random segregation
Cell division
D
Normal phenotype
Mutant
Threshold
OXPHOS enzyme activity
% mutation

1.2. **HUMAN COLON COX DEFICIENCY RAW DATA**

Sample Number	**Age**	**Number of crypts**	**Number of fully COX deficient crypts**	**Number of partially COX deficient crypts**	**COX deficiency proportion of individual crypts (mean +/- SD)**
1	17	1063	1	0	N/A
2	18	529	0	2	0.40 +/- 0.07
3	21	1188	2	4	0.30 +/- 0.19
4	24	635	0	0	N/A
5	25	807	0	3	0.12 +/- 0.03
6	25	573	2	1	0.79
7	25	1233	0	4	0.42 +/- 0.20
8	25	1024	3	0	N/A
9	26	579	2	2	0.29 +/- 0.17
10	26	1384	3	12	0.36 +/- 0.24
11	27	594	1	5	0.35 +/- 0.17
12	27	1819	0	7	0.42 +/- 0.23
13	31	551	0	10	0.32 +/- 0.25
14	32	1790	6	19	0.37 +/- 0.21
15	32	666	0	0	N/A
16	33	1314	10	28	0.26 +/- 0.19
17	34	359	0	1	0.34
18	34	1134	4	16	0.32 +/- 0.19
19	34	1356	2	6	0.26 +/- 0.20
20	35	1674	7	8	0.44+/- 0.24
21	37	564	0	5	0.24 +/- 0.24
22	37	1927	5	28	0.47 +/- 0.21
23	37	1163	25	21	0.42 +/- 0.26
24	37	1608	24	24	0.41 +/- 0.20
25	37	1970	24	62	0.31 +/- 0.17
26	38	1115	18	38	0.34 +/- 0.24
27	38	693	5	11	0.25 +/- 0.20
28	38	478	4	10	0.26 +/- 0.28
29	38	1341	15	19	0.35 +/- 0.23
30	38	973	4	4	0.49 +/- 0.25
31	39	1250	5	5	0.25 +/- 0.13
32	39	928	13	14	0.31 +/- 0.16
33	40	1389	1	16	0.28 +/- 0.20

34	40	2339	62	43	0.44 +/- 0.27
35	40	892	20	16	0.39 +/- 0.22
36	40	816	3	7	0.35 +/- 0.19
37	40	977	12	19	0.44 +/- 0.31
38	41	1345	38	45	0.48 +/- 0.22
39	41	405	1	15	0.37 +/- 0.26
40	41	2040	26	30	0.33 +/- 0.23
41	42	1237	0	3	0.35 +/- 0.12
42	42	1103	1	4	0.22 +/- 0.08
43	42	1449	10	26	0.40 +/- 0.23
44	42	1117	25	25	0.47 +/- 0.27
45	43	762	0	9	0.30 +/- 0.20
46	43	680	31	22	0.40 +/- 0.22
47	43	730	10	23	0.33 +/- 0.25
48	43	1287	22	9	0.29 +/- 0.15
49	43	625	2	9	0.25 +/- 0.17
50	43	1074	5	11	0.26 +/- 0.12
51	43	1408	12	9	0.25 +/- 0.15
52	43	1060	7	14	0.48 +/- 0.28
53	44	1271	7	27	0.44 +/- 0.24
54	44	1869	10	22	0.47 +/- 0.24
55	45	467	9	13	0.43 +/- 0.24
56	45	2144	24	31	0.36 +/- 0.24
57	45	1239	19	22	0.42 +/- 0.24
58	45	750	6	14	0.44 +/- 0.25
59	46	946	13	24	0.34 +/- 0.18
60	46	885	5	21	0.49 +/- 0.27
61	46	648	8	13	0.36 +/- 0.19
62	47	713	6	18	0.33 +/- 0.18
63	47	399	3	0	N/A
64	47	545	29	28	0.47 +/- 0.26
65	48	1256	11	17	0.42 +/- 0.20
66	48	665	28	5	0.55 +/- 0.31
67	48	824	9	25	0.22 +/- 0.17
68	48	1239	84	54	0.44 +/- 0.28
69	49	1278	8	16	0.38 +/- 0.18
70	49	667	0	7	0.37 +/- 0.12

71	49	687	6	3	0.33 +/- 0.15
72	49	546	2	12	0.23 +/- 0.21
73	50	1366	26	49	0.41 +/- 0.22
74	50	1674	23	27	0.25 +/- 0.18
75	50	944	2	22	0.28 +/- 0.17
76	50	765	15	6	0.33 +/- 0.22
77	50	516	11	20	0.11 +/- 0.06
78	50	1050	17	10	0.35 +/- 0.21
79	50	2209	30	59	0.36 +/- 0.24
80	50	1633	16	13	0.51 +/- 0.33
81	50	898	16	8	0.20 +/- 0.16
82	51	545	1	2	0.47 +/- 0.16
83	51	671	9	12	0.42 +/- 0.23
84	51	1305	25	38	0.35 +/- 0.18
85	51	521	11	17	0.37 +/- 0.20
86	51	1102	4	5	0.31 +/- 0.29
87	52	698	35	30	0.44 +/- 0.19
88	52	1060	3	16	0.40 +/- 0.26
89	52	2142	116	65	0.41 +/- 0.24
90	52	900	2	10	0.30 +/- 0.25
91	53	1170	6	22	0.37 +/- 0.20
92	53	981	17	18	0.53 +/- 0.23
93	55	663	10	7	0.43 +/- 0.16
94	55	716	38	29	0.47 +/- 0.19
95	55	753	20	16	0.42 +/- 0.22
96	55	460	6	9	0.43 +/- 0.25
97	55	656	22	14	0.34 +/- 0.25
98	56	1138	34	45	0.48 +/- 0.26
99	56	1343	10	18	0.47 +/- 0.26
100	56	1083	18	24	0.35 +/- 0.24
101	56	1258	42	57	0.44 +/- 0.24
102	56	1654	31	56	0.38 +/- 0.20
103	57	1198	48	51	0.36 +/- 0.22
104	57	1612	20	42	0.42 +/- 0.24
105	57	565	30	9	0.35 +/- 0.17
106	57	1036	4	11	0.33 +/- 0.25
107	57	724	82	25	0.36 +/- 0.21

108	58	1857	29	58	0.42 +/- 0.23
109	58	526	13	7	0.64 +/- 0.23
110	58	433	1	10	0.39 +/- 0.28
111	58	940	90	25	0.42 +/- 0.26
112	59	130	3	4	0.57 +/- 0.16
113	59	836	53	49	0.33 +/- 0.24
114	59	1787	41	47	0.40 +/- 0.24
115	59	717	31	50	0.42 +/- 0.25
116	60	229	25	17	0.49 +/- 0.22
117	60	810	140	62	0.39 +/- 0.28
118	60	1693	160	72	0.47 +/- 0.26
119	61	1025	51	56	0.44 +/- 0.26
120	61	850	73	33	0.41 +/- 0.23
121	61	1391	32	26	0.33 +/- 0.24
122	63	1309	199	37	0.42 +/- 0.18
123	63	1507	36	71	0.34 +/- 0.23
124	63	763	46	44	0.35 +/- 0.22
125	63	864	18	13	0.48 +/- 0.29
126	63	338	33	9	0.30 +/- 0.34
127	64	314	13	7	0.46 +/- 0.22
128	66	1409	90	51	0.44 +/- 0.23
129	66	826	42	30	0.33 +/- 0.23
130	66	1166	172	152	0.51 +/- 0.23
131	66	901	28	10	0.39 +/- 0.29
132	66	1890	52	65	0.36 +/- 0.23
133	67	1659	209	80	0.42 +/- 0.24
134	68	1152	30	55	0.45 +/- 0.20
135	68	807	22	3	0.51 +/- 0.25
136	68	1780	143	90	0.39 +/- 0.25
137	68	866	108	54	0.38 +/- 0.24
138	68	1375	17	34	0.40 +/- 0.27
139	70	1054	39	58	0.40 +/- 0.26
140	71	594	49	21	0.46 +/- 0.19
141	71	776	18	10	0.39 +/- 0.22
142	72	598	133	72	0.44 +/- 0.22
143	72	454	16	21	0.33 +/- 0.19
144	73	660	39	36	0.41 +/- 0.21

145	76	593	61	32	0.49 +/- 0.21
146	76	664	29	21	0.38 +/- 0.23
147	77	412	37	41	0.47 +/- 0.25
148	78	650	66	32	0.39 +/- 0.25

1.3. **CELL CYCLE KINETICS RAW DATA**

1.3.1. **LPA446 count and convergence data**

LPA446 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Positive	Positive	78	47	10	41	7
2	Positive	Positive	58	30	2	29	6
3a	Positive	Positive	50	23	5	28	5
3b	Positive	Positive	41	27	3	29	6
4	Positive	Positive	54	29	9	33	7
5a	Positive	Positive	20	16	1	16	5
5b	Positive	Positive	38	25	0	29	3
5c	Positive	Positive	46	32	3	30	3
6	Positive	Positive	56	36	8	38	7
7b	Positive	Positive	49	31	8	36	9
7c	Positive	Positive	54	40	16	42	5
8	Positive	Positive	42	30	10	31	5
9	Positive	Positive	41	25	5	31	5
10	Positive	Positive	64	43	14	42	5
11a	Positive	Positive	50	36	9	38	4
11b	Positive	Positive	67	40	14	39	7
11c	Positive	Positive	57	36	13	42	9
12	Positive	Positive	71	47	10	68	5
13a	Positive	Positive	51	40	15	39	3
13b	Positive	Positive	62	53	12	41	5
14	Positive	Positive	50	40	11	35	5
15	Positive	Positive	52	39	13	41	9
16	Positive	Positive	55	37	3	46	5
17	Positive	Positive	49	29	15	42	8
18	Positive	Positive	59	31	12	41	12
19	Positive	Positive	45	40	12	36	8
20	Positive	Positive	57	44	15	48	6
21	Positive	Positive	44	33	10	36	9
22	Positive	Positive	39	32	7	33	3
23	Positive	Positive	58	31	11	38	6
24	Positive	Positive	43	23	8	32	5
25	Positive	Positive	46	21	6	30	5
26	Positive	Positive	41	26	6	33	5
27	Positive	Positive	36	22	2	30	5
28	Positive	Positive	32	20	3	27	9
29	Positive	Positive	45	32	10	34	5
30	Positive	Positive	35	24	6	27	6
31	Positive	Positive	32	26	7	27	8
32	Positive	Positive	39	25	6	33	5

33	Positive	Positive	47	35	12	40	7
34	Positive	Positive	46	33	14	39	4
35	Positive	Positive	46	31	11	35	6
36	Positive	Positive	53	37	12	48	5
37a	Positive	Positive	52	40	8	48	5
37b	Positive	Positive	63	52	6	51	2
38a	Positive	Positive	39	21	6	23	6
38b	Positive	Positive	40	19	3	32	3
39	Positive	Positive	48	22	7	32	9
40	Positive	Positive	33	23	3	26	2
41	Positive	Positive	48	33	5	35	4
42	Positive	Positive	51	28	3	36	4
43	Positive	Positive	50	43	3	44	6
44	Positive	Positive	48	37	9	39	5
45	Positive	Positive	59	37	8	49	6
46	Positive	Positive	39	23	5	29	3
47	Positive	Positive	42	29	6	32	6
48	Positive	Positive	48	22	8	31	5
49a	Positive	Positive	44	32	6	33	4
49b	Positive	Positive	49	28	4	37	6
50	Positive	Positive	52	35	11	44	6
51a	Positive	Positive	60	38	7	48	2
51b	Positive	Positive	54	40	7	40	4
52	Positive	Positive	48	33	7	41	3

LPA446 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	1	2	1	0	0	3	0	1	8	25	3	0	0	8	26
2	0	0	3	0	0	0	0	3	0	2	21	3	0	0	4	22
3a	0	0	2	1	0	0	0	2	1	4	14	6	0	0	3	17
3b	0	2	4	0	0	0	0	0	0	1	20	2	0	0	0	12
4	0	5	2	0	0	0	0	0	0	4	14	8	0	0	4	17
5a	0	1	2	1	0	0	1	0	0	0	11	1	0	0	1	2
5b	0	0	3	0	0	0	0	0	0	0	19	7	0	0	3	6
5c	0	1	2	0	0	0	0	0	0	2	23	2	0	0	4	12
6	0	3	2	1	0	0	1	0	1	4	24	3	0	0	2	15
7b	1	1	5	0	0	0	2	0	2	4	18	5	0	0	1	10
7c	1	0	3	1	0	0	0	0	4	11	20	2	0	0	6	6
8	0	0	3	0	0	0	0	2	2	8	16	2	0	0	3	6
9	1	0	2	1	0	0	0	1	2	2	21	2	0	0	0	9
10	0	4	1	0	0	0	0	0	7	3	26	1	0	0	9	13
11a	0	2	0	2	0	0	0	0	1	6	22	5	0	0	6	6
11b	0	2	3	0	0	0	1	1	4	8	19	3	0	0	7	19
11c	2	1	3	2	0	0	0	1	2	8	19	5	0	0	5	9
12	0	1	4	0	0	0	0	0	2	6	35	20	0	1	0	2
13a	0	2	1	0	0	0	0	0	5	6	22	3	1	1	8	2
13b	0	1	3	1	0	0	0	0	0	10	24	2	0	1	14	6
14	0	2	2	0	0	0	1	0	2	7	20	2	0	0	8	6
15	0	2	6	0	0	0	1	0	3	8	20	2	0	0	2	8
16	0	0	5	0	0	0	0	0	2	1	31	7	0	0	0	9

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
17	3	2	3	0	0	0	0	0	3	7	17	7	0	0	0	7
18	0	1	7	2	0	0	0	2	3	8	15	5	0	0	0	16
19	0	3	4	0	0	0	1	0	0	7	21	1	0	2	2	4
20	0	3	2	1	0	0	0	0	2	10	26	4	0	0	3	6
21	0	3	5	1	0	0	0	0	0	7	16	4	0	0	2	6
22	0	1	1	1	0	0	0	0	0	6	24	0	0	0	0	6
23	0	3	3	0	0	0	0	0	0	8	17	7	0	0	0	20
24	0	1	3	1	0	0	0	0	1	6	13	7	0	0	0	11
25	0	0	4	1	0	0	0	0	2	4	13	6	0	0	0	16
26	0	0	4	1	0	0	0	0	1	5	17	5	0	0	0	8
27	0	0	5	0	0	0	0	0	0	2	15	8	0	0	0	6
28	0	0	4	4	0	0	0	1	0	3	13	3	0	0	0	4
29	0	4	0	0	0	0	1	0	0	6	20	4	0	0	1	9
30	1	0	3	1	0	0	0	1	0	5	16	1	0	0	0	7
31	1	1	6	0	0	0	0	0	1	4	13	1	0	0	2	3
32	0	2	3	0	0	0	0	0	1	3	17	7	0	0	0	6
33	0	4	3	0	0	0	0	0	1	7	21	4	0	0	0	7
34	0	3	1	0	0	0	0	0	3	8	21	3	0	0	0	7
35	0	2	2	1	0	0	0	1	2	7	19	2	0	0	1	9
36	0	1	2	1	0	0	1	0	3	8	24	9	0	0	1	3
37a	1	0	3	1	0	0	0	0	0	7	30	6	0	0	0	4
37b	0	0	2	0	0	0	0	0	1	5	40	3	0	0	5	7
38a	0	0	3	0	0	0	0	3	0	6	12	2	0	0	0	13
38b	0	0	3	0	0	0	0	0	0	3	13	13	0	0	0	8
39	2	1	2	0	0	0	2	2	2	2	15	8	0	0	0	12
40	0	0	1	0	0	0	0	1	1	2	20	2	0	0	0	6
41	0	0	4	0	0	0	0	0	0	5	22	4	0	0	2	11
42	0	1	2	1	0	0	0	0	1	1	23	7	0	0	1	14
43	0	1	5	0	0	0	0	0	0	2	33	3	0	0	2	4

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
44	0	1	3	0	0	0	0	1	1	7	25	2	0	0	1	7
45	2	1	1	2	0	0	0	0	5	0	34	4	0	0	1	9
46	0	0	3	0	0	0	0	0	1	4	14	7	0	0	2	8
47	0	2	3	1	0	0	0	0	1	3	19	3	0	0	2	8
48	0	2	2	1	0	0	0	0	2	4	12	8	0	0	2	15
49a	0	2	2	0	0	0	0	0	0	4	23	2	0	0	1	10
49b	1	0	3	2	0	0	0	0	0	3	22	6	0	0	0	12
50	0	0	4	2	0	0	0	0	5	6	25	2	0	0	0	8
51a	0	0	2	0	0	0	0	0	1	6	27	12	0	0	3	9
51b	0	2	1	1	0	0	0	0	0	5	28	3	0	0	4	10
52	0	0	2	0	0	0	0	1	1	6	25	7	0	0	0	6

1.3.2. **LPA457 count and convergence data**

LPA457 count data

				Total			
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1a	Positive	Positive	48	22	9	35	9
1b	Positive	Positive	67	43	11	49	6
2	Positive	Positive	54	24	2	32	7
3	Positive	Positive	60	37	10	47	8
4	Positive	Positive	42	29	8	33	6
5	Positive	Positive	68	40	10	60	7
6	Positive	Positive	47	31	7	41	6
7	Positive	Positive	49	43	3	40	6
8a	Positive	Positive	46	35	7	39	8
8b	Positive	Positive	47	34	7	40	3
9	Positive	Positive	56	35	8	38	5
10	Positive	Positive	58	40	3	49	2
11	Positive	Positive	54	27	4	41	5
12a	Positive	Positive	31	17	2	23	3
12b	Positive	Positive	45	35	2	34	5
13	Positive	Positive	46	35	6	37	6
14	Positive	Positive	42	22	6	32	6
15	Positive	Positive	38	23	11	32	4
16a	Positive	Positive	55	44	5	45	4
16b	Positive	Positive	42	22	1	32	8
16c	Positive	Positive	44	21	11	28	6
17a	Positive	Positive	41	30	4	36	3
17b	Positive	Positive	39	27	4	29	3
18	Positive	Positive	33	24	7	29	4
19	Positive	Positive	44	26	5	34	4
20	Positive	Positive	41	21	7	30	12
21	Positive	Positive	46	19	6	36	7
22	Positive	Positive	55	34	13	49	4
23	Positive	Positive	31	23	7	28	6
24	Positive	Positive	54	35	5	41	7
25	Positive	Positive	47	39	12	38	5
26a	Positive	Positive	40	30	6	33	3
26b	Positive	Positive	53	29	5	38	7
27	Positive	Positive	46	29	4	37	6
28	Positive	Positive	50	35	4	43	3
29a	Positive	Positive	66	37	7	51	6
29b	Positive	Positive	54	25	3	35	5
30	Positive	Positive	37	17	5	27	2
31a	Positive	Positive	43	23	7	32	4
31b	Positive	Positive	56	37	10	41	3

32a	Positive	Positive	34	27	12	31	4
32b	Positive	Positive	39	15	7	31	5
33	Positive	Positive	42	32	11	37	4
34	Positive	Positive	54	34	7	41	4
35	Positive	Positive	40	26	7	28	6
36a	Positive	Positive	16	7	4	13	4
36b	Positive	Positive	23	13	3	19	6
36c	Positive	Positive	16	9	4	12	4
36d	Positive	Positive	18	5	0	12	3
37	Positive	Positive	44	19	0	27	6
38a	Positive	Positive	45	29	4	31	5
38b	Positive	Positive	39	30	5	30	4
38b	Positive	Positive	39	30	5	30	4
39	Positive	Positive	42	28	2	35	7
40	Positive	Positive	52	26	6	32	5
41a	Positive	Positive	34	20	4	26	2
41b	Positive	Positive	45	38	2	41	7
41c	Positive	Positive	40	21	5	30	3
42	Positive	Positive	49	30	3	40	3
43	Positive	Positive	42	24	7	31	4
44	Positive	Positive	43	31	5	36	3
45	Positive	Positive	48	26	6	44	6
46a	Positive	Positive	42	26	2	34	4
46b	Positive	Positive	46	29	5	41	1
46c	Positive	Positive	57	34	8	47	4
47	Positive	Positive	59	36	8	47	2
48a	Positive	Positive	36	18	3	31	5
48b	Positive	Positive	48	31	6	38	6
49	Positive	Positive	56	38	12	50	6
50	Positive	Positive	37	12	7	26	7
51a	Positive	Positive	50	31	8	42	4
51b	Positive	Positive	46	28	6	41	7
52	Positive	Positive	57	34	6	45	6
53	Positive	Positive	57	35	11	43	8
54a	Positive	Positive	53	34	5	39	4
54b	Positive	Positive	51	33	8	41	1
55	Positive	Positive	42	25	5	31	6
56	Positive	Positive	57	37	5	40	8
57	Positive	Positive	26	17	2	22	3
58	Positive	Positive	54	37	7	47	3
59	Positive	Positive	32	24	11	29	6
60	Positive	Positive	67	46	14	54	3

LPA457 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1a	0	3	6	0	0	0	0	0	4	2	10	10	0	0	1	12
1b	0	1	5	0	0	0	0	0	1	7	28	7	1	1	1	15
2	0	1	2	4	0	0	0	0	0	1	19	5	0	0	1	21
3	1	1	3	3	0	0	0	0	1	7	26	5	0	0	0	13
4	1	3	2	0	0	0	0	0	1	3	21	2	0	0	0	9
5	0	2	4	1	0	0	0	0	0	8	26	19	0	0	0	8
6	0	1	4	1	0	0	0	0	0	5	21	9	1	0	0	5
7	0	0	6	0	0	0	0	0	0	3	27	4	0	0	7	2
8a	0	2	5	1	0	0	0	0	0	5	23	3	0	0	0	7
8b	0	0	3	0	0	0	0	0	2	5	26	4	0	0	0	7
9	1	0	4	0	0	0	0	0	1	6	25	1	0	0	0	18
10	0	0	1	1	0	0	0	0	2	1	37	7	0	0	1	8
11	0	0	1	4	0	0	0	0	2	2	24	8	0	0	0	13
12a	0	0	0	1	0	0	0	2	0	2	15	5	0	0	0	6
12b	0	0	5	0	0	0	0	0	0	2	25	2	0	0	3	8
13	0	2	4	0	0	0	0	0	0	4	24	3	0	0	1	8
14	0	1	5	0	0	0	0	0	4	1	15	6	0	0	0	10
15	1	1	2	0	0	0	0	0	4	5	15	4	0	0	0	6
16a	0	3	1	0	0	0	0	0	0	2	37	2	0	0	1	9
16b	0	1	6	0	0	0	0	1	0	0	15	10	0	0	0	9
16c	1	2	2	1	0	0	0	0	2	6	11	3	0	0	0	16
17a	0	1	2	0	0	0	0	0	0	3	23	7	0	0	1	4
17b	0	0	3	0	0	0	0	0	1	3	20	2	0	0	1	9

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
18	1	3	0	0	0	0	0	0	0	3	18	4	0	0	0	4
19	0	0	3	1	0	0	0	0	1	4	19	6	0	0	0	10
20	3	2	7	0	0	0	0	0	2	0	12	4	0	0	0	11
21	1	0	3	3	0	0	0	0	3	2	14	10	0	0	0	10
22	0	1	1	2	0	0	0	0	5	7	25	8	0	0	0	6
23	0	2	4	0	0	0	0	0	2	3	13	4	0	0	1	2
24	0	2	5	0	0	0	0	0	1	2	26	5	0	0	0	13
25	0	2	3	0	0	0	0	0	1	9	23	0	0	0	2	7
26a	0	0	3	0	0	0	0	0	0	6	21	3	0	0	0	7
26b	0	3	3	1	0	0	0	0	1	1	22	7	0	0	0	15
27	0	0	6	0	0	0	0	0	1	3	20	7	0	0	0	9
28	0	2	1	0	0	0	0	0	0	2	29	9	0	0	1	6
29a	1	0	4	1	0	0	0	0	0	6	26	13	0	0	1	14
29b	1	0	4	0	0	0	0	0	2	0	20	8	0	0	1	18
30	0	0	2	0	0	0	0	0	1	4	11	9	0	0	0	10
31a	0	2	2	0	0	0	0	0	2	3	16	7	0	0	0	11
31b	0	2	1	0	0	0	0	0	0	8	26	4	0	0	0	15
32a	0	0	3	1	0	0	0	0	1	11	13	2	0	0	0	3
32b	0	0	2	3	0	0	0	0	2	5	8	11	0	0	0	8
33	0	2	2	0	0	0	0	0	2	7	21	3	0	0	0	5
34	0	1	2	1	0	0	0	0	1	5	26	5	0	0	0	13
35	1	4	0	1	0	0	0	0	0	2	16	4	0	0	4	8
36a	1	0	1	2	0	0	0	0	1	2	4	2	0	0	0	3
36b	2	0	2	2	0	0	0	0	0	1	9	3	0	0	1	3
36c	0	2	1	1	0	0	0	0	0	2	4	2	0	0	0	4
36d	0	0	1	2	0	0	0	0	0	0	4	5	0	0	0	6
37	0	0	2	3	0	0	1	0	0	0	15	7	0	0	1	15
38a	0	2	3	0	0	0	0	0	0	2	22	2	0	0	0	14
38b	0	1	2	0	0	0	0	1	0	4	22	1	0	0	1	7

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
39	0	1	6	0	0	0	0	0	1	0	21	6	0	0	0	7
40	0	3	2	0	0	0	0	0	0	3	18	6	0	0	0	20
41a	0	1	1	0	0	0	0	0	1	2	16	5	0	0	0	8
41b	0	2	5	0	0	0	0	0	0	0	31	3	0	0	0	4
41c	0	2	1	0	0	0	0	0	3	0	18	6	0	0	0	10
42	0	0	2	1	0	0	0	0	1	2	26	8	0	0	0	9
43	1	0	2	1	0	0	0	0	2	4	18	3	0	0	0	11
44	1	0	2	0	0	0	0	0	1	3	22	7	0	0	4	3
45	1	0	0	5	0	0	0	0	2	3	23	10	0	0	0	4
46a	0	0	4	0	0	0	0	0	1	1	21	7	0	0	0	8
46b	0	0	0	1	0	0	0	0	2	3	26	9	0	0	0	5
46c	0	2	1	1	0	0	0	0	0	6	25	12	0	0	0	10
47	0	0	1	0	0	0	0	1	1	7	28	10	0	0	0	11
48a	0	0	3	2	0	0	0	0	1	2	13	10	0	0	0	5
48b	0	2	2	1	0	0	0	1	0	4	22	7	0	0	1	8
49	0	2	1	3	0	0	0	0	2	8	27	7	0	0	0	6
50	0	0	2	4	0	0	0	1	6	1	9	4	0	0	0	10
51a	2	0	1	1	0	0	0	0	4	2	28	4	0	0	0	8
51b	1	2	4	0	0	0	0	0	1	2	20	11	0	0	0	5
52	0	0	5	1	0	0	0	0	2	4	25	8	0	0	0	12
53	1	1	4	2	0	0	0	0	3	6	24	2	0	0	0	14
54a	0	0	3	1	0	0	0	0	0	5	26	4	0	0	0	14
54b	0	0	1	0	0	0	0	0	4	4	28	4	0	0	0	10
55	0	4	1	1	0	0	0	0	0	1	19	5	0	0	0	11
56	0	3	5	0	0	0	0	0	0	2	26	4	0	0	1	16
57	0	0	2	1	0	0	0	0	0	2	13	4	0	0	0	4
58	0	0	3	0	0	0	0	0	0	7	26	11	0	0	1	6
59	0	0	5	1	0	0	0	0	3	8	10	2	0	0	1	2

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
60	0	1	2	0	0	0	0	0	1	12	31	7	0	0	0	13

1.3.3. **LPA497 count and convergence data**

LPA497 count data

				Total			
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Positive	Positive	44	30	5	32	5
2	Positive	Positive	38	33	2	35	5
3	Positive	Positive	47	39	8	40	5
4a	Positive	Positive	36	30	7	33	6
4b	Positive	Positive	32	22	9	31	3
5	Positive	Positive	56	37	7	50	7
6a	Positive	Positive	56	43	5	51	5
6b	Positive	Positive	64	43	8	53	7
7	Positive	Positive	52	42	5	45	8
8	Positive	Positive	38	25	6	27	5
9	Positive	Positive	54	31	10	35	8
10a	Positive	Positive	58	44	9	48	6
10b	Positive	Positive	51	38	7	44	2
11	Positive	Positive	46	24	3	33	7
12	Positive	Positive	69	42	14	60	6
13	Positive	Positive	58	29	8	46	4
14	Positive	Positive	56	37	10	47	7
15	Positive	Positive	54	39	4	42	2
16	Positive	Positive	52	39	6	46	4
17	Positive	Positive	65	52	16	54	7
18	Positive	Positive	46	25	4	33	5
19	Positive	Positive	46	36	9	42	6
20	Positive	Positive	36	25	7	28	4
21	Positive	Positive	53	30	19	47	4
22a	Positive	Positive	34	19	9	28	5
22b	Positive	Positive	34	20	11	29	1
22c	Positive	Positive	39	22	17	35	4
23	Positive	Positive	65	30	11	47	6
24a	Positive	Positive	41	21	5	32	8
24b	Positive	Positive	46	29	9	40	8
25	Positive	Positive	52	31	9	41	6
26	Positive	Positive	44	23	5	36	8
27	Positive	Positive	55	31	8	43	7
28	Positive	Positive	54	42	5	45	4
29	Positive	Positive	43	36	7	40	3
30	Positive	Positive	49	39	11	48	1
31	Positive	Positive	58	39	4	41	5
32	Positive	Positive	61	35	8	44	6
33	Positive	Positive	49	35	4	41	5
34a	Positive	Positive	49	37	6	41	4

34b	Positive	Positive	36	29	7	32	12
35	Positive	Positive	51	36	12	43	3
36	Positive	Positive	54	41	11	42	3
37a	Positive	Positive	64	50	6	55	5
37b	Positive	Positive	71	51	11	59	4
38	Positive	Positive	57	44	3	49	7
39a	Positive	Positive	52	38	6	41	5
39b	Positive	Positive	54	36	12	42	7
40	Positive	Positive	45	40	9	42	3
41	Positive	Positive	104	68	14	88	6
42	Positive	Positive	37	26	9	30	5
43a	Positive	Positive	53	41	3	42	5
43b	Positive	Positive	43	38	3	37	4
44a	Positive	Positive	78	54	10	62	5
44b	Positive	Positive	48	40	4	42	4
45a	Positive	Positive	42	27	6	30	6
45b	Positive	Positive	41	34	6	36	2
46	Positive	Positive	42	31	5	35	3
47a	Positive	Positive	41	34	3	38	9
47b	Positive	Positive	61	48	9	55	1
48	Positive	Positive	66	42	8	53	5

LPA497 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	1	4	0	0	0	0	0	2	2	23	0	0	0	0	12
2	0	2	3	0	0	0	0	0	0	0	28	2	0	0	0	3
3	0	4	1	0	0	0	0	0	0	4	29	2	0	0	1	6
4a	0	5	0	1	0	0	0	0	0	2	23	2	0	0	0	3
4b	2	0	0	1	0	0	0	0	0	7	15	6	0	0	0	1
5	0	1	6	0	0	0	0	0	3	3	27	10	0	0	0	6
6a	0	0	4	1	0	0	0	0	2	3	36	5	0	0	0	5
6b	0	1	3	1	0	0	0	2	2	5	34	7	0	0	0	9
7	0	2	6	0	0	0	0	0	1	2	31	3	0	0	1	6
8	0	2	3	0	0	0	0	0	0	4	16	2	0	0	0	11
9	1	5	2	0	0	0	0	0	1	3	21	2	0	0	0	19
10a	0	2	4	0	0	0	0	0	1	6	32	3	0	0	0	10
10b	0	0	2	0	0	0	0	0	0	7	29	6	0	0	0	7
11	2	1	1	3	0	0	0	0	0	0	22	4	0	0	0	13
12	0	2	2	2	0	0	0	0	3	9	29	13	0	0	0	9
13	0	0	3	0	0	0	0	1	4	4	22	13	0	0	0	11
14	0	0	6	1	0	0	0	0	7	3	28	2	0	0	0	9
15	0	1	1	0	0	0	0	0	0	3	32	5	0	0	2	10
16	0	0	4	0	0	0	0	0	6	0	35	1	0	0	0	6
17	2	5	0	0	0	0	0	0	0	9	38	0	0	0	0	11
18	0	2	3	0	0	0	0	0	0	2	18	8	0	0	0	13
19	0	0	6	0	0	0	0	0	4	5	25	2	0	0	0	4
20	0	2	2	0	0	0	0	0	1	4	16	3	0	0	1	7

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
21	3	0	1	0	0	0	0	0	10	6	23	4	0	0	0	6
22a	0	1	4	0	0	0	0	0	6	2	12	3	0	0	0	6
22b	0	1	0	0	0	0	0	0	4	6	13	5	0	0	0	5
22c	0	1	3	0	0	0	0	0	8	7	10	6	0	1	0	3
23	1	1	3	1	0	0	0	0	6	3	23	9	0	0	0	18
24a	0	1	5	2	0	0	0	0	1	3	12	8	0	0	0	9
24b	2	1	3	2	0	0	0	0	3	3	22	4	0	0	0	6
25	0	0	3	2	0	0	1	0	2	7	19	8	0	0	1	9
26	0	1	5	2	0	0	0	0	2	2	15	9	0	0	0	8
27	0	1	4	2	0	0	0	0	3	4	21	8	0	0	1	11
28	1	0	3	0	0	0	0	0	0	4	32	5	0	0	3	6
29	0	1	2	0	0	0	0	0	0	6	27	4	0	0	0	3
30	0	1	0	0	0	0	0	0	3	7	31	6	0	0	0	1
31	0	0	3	2	0	0	0	0	0	4	27	5	0	0	5	12
32	0	1	4	1	0	0	0	0	3	4	22	9	0	0	4	13
33	0	0	5	0	0	0	0	0	1	3	26	6	0	0	1	7
34a	0	0	3	0	0	0	1	0	1	5	27	5	0	0	1	6
34b	0	3	7	1	0	0	1	0	1	3	14	3	0	0	1	2
35	0	1	2	0	0	0	0	0	6	5	26	3	0	0	2	6
36	0	0	2	0	0	0	0	1	3	8	27	2	0	0	4	7
37a	0	0	5	0	0	0	0	0	1	5	38	6	0	0	2	7
37b	0	4	0	0	0	0	0	0	3	4	43	5	0	0	0	12
38	0	0	7	0	0	0	0	0	1	2	35	4	0	0	0	8
39a	0	0	3	1	0	0	1	0	2	4	27	4	0	0	3	7
39b	0	0	7	0	0	0	0	0	7	5	21	2	0	0	3	9
40	0	1	2	0	0	0	0	0	1	7	30	1	0	0	0	3
41	1	1	4	0	0	0	0	0	3	9	54	16	0	0	0	16
42	0	0	5	0	0	0	0	0	5	4	16	0	0	0	1	6
43a	0	1	4	0	0	0	0	0	0	2	32	3	0	0	2	9

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
43b	0	1	3	0	0	0	0	0	0	2	31	0	0	0	1	5
44a	0	2	3	0	0	0	0	0	1	7	36	13	0	0	6	10
44b	0	0	4	0	0	0	0	0	2	2	33	1	0	0	1	5
45a	0	2	4	0	0	0	0	0	3	1	18	2	0	0	2	10
45b	0	0	2	0	0	0	0	0	0	6	26	2	0	0	0	5
46	0	2	1	0	0	0	0	0	2	1	25	4	0	0	2	5
47a	1	2	5	1	0	0	0	0	0	0	27	2	0	0	0	3
47b	0	0	1	0	0	0	0	0	4	5	42	3	0	0	0	6
48	0	1	2	1	0	0	0	1	7	0	37	5	0	0	2	10

1.3.4. **LPA499 count and convergence data**

LPA499 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1a	Positive	Positive	48	36	6	38	7
1b	Positive	Positive	22	20	4	21	3
2a	Positive	Positive	46	32	7	39	4
2b	Positive	Positive	22	12	8	17	5
3a	Positive	Positive	54	40	14	43	3
3b	Positive	Positive	30	24	18	28	4
4	Positive	Positive	48	31	6	33	4
5a	Positive	Positive	47	37	5	38	2
5b	Positive	Positive	46	37	9	38	5
6	Positive	Positive	51	36	11	39	3
7a	Positive	Positive	54	34	9	40	3
7b	Positive	Positive	62	51	18	52	5
8	Positive	Positive	44	33	5	37	2
9	Positive	Positive	49	37	8	37	5
10	Positive	Positive	50	36	6	38	3
11a	Positive	Positive	51	44	14	46	5
11b	Positive	Positive	44	37	11	41	5
12a	Positive	Positive	48	31	14	40	6
12b	Positive	Positive	43	29	2	33	6
13	Positive	Positive	53	37	6	40	5
14	Positive	Positive	44	31	4	36	5
15	Positive	Positive	53	45	10	45	4
16a	Positive	Positive	34	19	5	26	5
16b	Positive	Positive	46	28	11	38	10
17	Positive	Positive	51	31	15	41	7
18	Positive	Positive	47	31	10	30	3
19	Positive	Positive	65	31	3	41	3
20	Positive	Positive	41	27	11	28	3
21	Positive	Positive	32	24	10	26	5
22a	Positive	Positive	42	26	5	24	3
22b	Positive	Positive	47	26	5	34	4
23a	Positive	Positive	37	29	3	28	4
23b	Positive	Positive	31	23	7	29	5
24	Positive	Positive	49	32	4	39	3
25a	Positive	Positive	51	39	10	47	6
25b	Positive	Positive	49	40	15	44	6
26	Positive	Positive	59	52	10	49	7
27	Positive	Positive	63	42	17	52	5
28	Positive	Positive	38	33	10	32	4
29	Positive	Positive	50	39	16	44	5

30	Positive	Positive	60	41	17	48	3
31	Positive	Positive	44	37	8	37	3
32	Positive	Positive	46	35	5	39	4
33	Positive	Positive	35	29	1	30	9
34	Positive	Positive	53	39	13	47	6
35	Positive	Positive	45	29	8	34	2
36	Positive	Positive	44	28	10	33	3
37	Positive	Positive	58	41	8	45	5
38	Positive	Positive	50	31	5	35	5
39a	Positive	Positive	47	38	10	41	6
39b	Positive	Positive	35	23	12	28	7
40	Positive	Positive	41	35	4	36	6
41a	Positive	Positive	53	37	10	41	4
41b	Positive	Positive	54	31	13	43	9
42	Positive	Positive	47	35	5	35	8
43a	Positive	Positive	53	49	11	49	5
43b	Positive	Positive	51	35	25	48	2
44a	Positive	Positive	41	38	7	39	4
44b	Positive	Positive	43	26	4	33	3
45	Positive	Positive	54	30	6	40	6
46	Positive	Positive	35	30	6	30	5
47	Positive	Positive	71	37	8	47	7
48	Positive	Positive	39	32	6	33	4
49	Positive	Positive	55	36	10	41	3
50a	Positive	Positive	47	31	15	37	5
50b	Positive	Positive	37	20	2	29	5

LPA499 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1a	1	0	4	1	0	0	1	0	0	5	26	1	0	0	0	9
1b	0	0	3	0	0	0	0	0	0	4	13	1	0	0	0	1
2a	2	0	2	0	0	0	0	0	3	2	28	2	0	0	0	7
2b	1	2	2	0	0	0	0	0	1	4	4	3	0	0	0	5
3a	1	2	0	0	0	0	0	0	1	10	26	3	0	0	2	9
3b	0	2	2	0	0	0	0	0	4	12	7	1	0	0	1	1
4	0	0	1	1	0	0	2	0	1	5	22	3	0	0	1	12
5a	0	0	2	0	0	0	0	0	1	4	30	1	0	0	1	8
5b	2	0	3	0	0	0	0	0	1	6	25	1	0	0	3	5
6	0	2	0	0	0	0	1	0	0	9	24	4	0	0	0	11
7a	0	2	1	0	0	0	0	0	2	5	26	4	0	0	0	14
7b	0	0	4	1	0	0	0	0	0	18	28	1	0	0	1	9
8	0	0	1	1	0	0	0	0	1	4	27	3	0	0	1	6
9	0	2	3	0	0	0	0	0	0	6	25	1	0	0	1	11
10	0	0	2	1	0	0	0	0	1	5	27	2	0	0	2	10
11a	0	1	4	0	0	0	0	0	1	12	26	2	0	0	1	4
11b	0	1	2	2	0	0	0	0	3	7	25	1	0	0	2	1
12a	0	3	3	0	0	0	0	0	2	9	16	7	0	0	0	8
12b	2	0	3	1	0	0	0	0	0	0	25	2	0	0	1	9
13	0	0	2	3	0	0	0	0	0	6	26	3	0	0	3	10
14	2	1	2	0	0	0	0	0	1	0	27	3	0	0	1	7
15	0	1	3	0	0	0	0	0	1	8	31	1	0	0	2	6
16a	0	1	2	2	0	0	0	0	1	3	13	4	0	0	0	8

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
16b	0	2	7	1	0	0	0	0	4	5	14	5	0	0	0	8
17	2	1	3	1	0	0	0	0	7	4	19	4	1	0	4	5
18	0	0	2	0	0	0	0	1	0	10	18	0	0	0	1	15
19	0	0	3	0	0	0	0	0	1	2	25	10	0	0	1	23
20	1	0	2	0	0	0	0	0	3	7	15	0	0	0	3	10
21	0	0	5	0	0	0	0	0	4	5	10	2	1	0	4	1
22a	0	1	2	0	0	0	0	0	0	3	17	1	0	1	2	15
22b	0	0	4	0	0	0	0	0	1	4	18	7	0	0	0	13
23a	0	0	3	0	0	0	1	0	0	3	21	1	0	0	1	7
23b	1	1	3	0	0	0	0	0	1	4	15	4	0	0	0	2
24	1	0	0	2	0	0	0	0	1	2	29	4	0	0	1	9
25a	0	3	1	2	0	0	0	0	2	5	29	5	0	0	1	3
25b	0	4	0	2	0	0	0	0	0	11	24	3	0	0	1	4
26	0	1	5	0	0	1	0	0	2	5	35	1	0	1	4	4
27	0	2	3	0	0	0	0	0	6	9	28	4	0	0	0	11
28	0	2	2	0	0	0	0	0	0	8	18	2	0	0	3	3
29	1	3	1	0	0	0	0	0	2	10	24	3	0	0	1	5
30	1	0	2	0	0	0	0	0	4	12	26	3	0	0	1	11
31	0	2	1	0	0	0	0	0	1	5	27	1	0	0	2	5
32	0	0	4	0	0	0	0	0	2	3	28	2	0	0	0	7
33	0	0	5	2	0	0	2	0	0	1	21	1	0	0	0	3
34	0	2	3	1	0	0	0	0	5	6	28	2	0	0	0	6
35	0	0	0	2	0	0	0	0	1	7	20	4	0	0	2	9
36	3	0	0	0	0	0	0	0	1	6	22	1	0	0	0	11
37	0	2	3	0	0	0	0	0	4	2	32	2	0	0	2	11
38	1	2	0	0	0	0	1	1	1	1	24	6	0	0	3	10
39a	0	1	4	1	0	0	0	0	3	6	26	0	0	0	1	5
39b	2	1	4	0	0	0	0	0	2	7	11	1	0	0	0	7
40	0	1	5	0	0	0	0	0	0	3	25	2	0	0	1	4

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
41a	0	2	2	0	0	0	0	0	3	5	27	2	0	0	1	11
41b	2	4	2	0	0	0	0	1	4	3	22	6	0	0	0	10
42	0	0	7	0	0	0	0	1	0	4	20	4	0	1	3	7
43a	0	2	3	0	0	0	0	0	0	9	34	1	0	0	1	3
43b	0	0	2	0	0	0	0	0	8	17	16	5	0	0	0	3
44a	0	1	3	0	0	0	0	0	1	5	29	0	0	0	0	2
44b	0	0	2	1	0	0	0	0	3	1	23	3	0	0	0	10
45	3	0	1	2	0	0	0	0	1	2	27	4	0	0	0	14
46	0	2	2	0	0	0	1	0	2	2	22	0	0	0	1	3
47	0	2	3	0	0	0	2	0	3	3	26	10	0	0	1	21
48	0	0	3	1	0	0	0	0	1	5	23	0	0	0	1	5
49	0	2	1	0	0	0	0	0	3	5	28	2	0	0	0	14
50a	3	1	1	0	0	0	0	0	2	9	18	3	0	0	2	8
50b	0	0	4	1	0	0	0	0	1	1	14	8	0	0	1	7

1.3.5. LPA187 count and convergence data

LPA187 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1a	Negative	Positive	48	15	7	13	3
1b	Negative	Positive	41	18	10	21	11
2	Negative	Positive	45	22	9	14	9
3	Positive	Negative	42	27	11	31	12
4	Negative	Negative	50	38	6	37	6
5	Positive	Negative	56	28	9	35	2
6	Positive	Negative	50	30	8	36	6
7	Negative	Positive	56	41	14	34	7
8a	Negative	Positive	42	24	12	24	4
8b	Positive	Negative	41	29	12	31	9
9a	Positive	Negative	55	27	10	28	5
9b	Positive	Negative	45	21	6	26	3
10	Positive	Positive	56	47	6	36	4
11	Positive	Positive	43	29	4	24	2
12	Positive	Positive	65	43	3	50	4
13a	Negative	Positive	31	19	8	16	2
13b	Positive	Positive	35	21	12	30	6
13c	Positive	Positive	25	17	5	17	6
14	Positive	Positive	23	18	4	15	7
15	Positive	Positive	39	29	4	28	3
16	Negative	Negative	35	28	8	22	8
17	Positive	Positive	63	36	9	43	10
18	Positive	Negative	69	41	15	49	9
19	Negative	Positive	43	27	12	34	7
20	Negative	Positive	61	40	9	49	5
21	Positive	Positive	69	51	14	51	5
22	Positive	Positive	36	23	6	28	8
23a	Negative	Positive	56	33	4	39	6
23b	Negative	Positive	32	16	8	20	5
24	Positive	Negative	50	37	8	38	9
25	Negative	Positive	65	38	12	50	5
26	Negative	Positive	57	36	15	37	7
27	Negative	Positive	81	61	12	62	5
28a	Negative	Positive	41	28	10	30	2
28b	Positive	Positive	48	38	5	28	2
28c	Positive	Negative	43	29	5	26	2
29	Negative	Positive	53	37	6	38	4
30a	Positive	Positive	29	19	4	22	1
30b	Positive	Negative	57	41	3	38	5
30c	Positive	Positive	18	9	2	12	3

31	Positive	Negative	51	26	6	33	3
32a	Negative	Positive	42	26	2	28	5
32b	Positive	Positive	42	21	2	25	7
33a	Positive	Positive	41	34	10	37	3
33b	Negative	Positive	70	50	8	50	8
34a	Negative	Negative	43	27	2	33	4
34b	Positive	Positive	35	22	7	25	8
34c	Negative	Negative	18	6	3	8	2
34d	Negative	Positive	23	11	5	15	3
35	Negative	Positive	76	49	15	63	5
36	Positive	Positive	71	52	4	55	8
37	Negative	Positive	72	25	4	36	6
38	Negative	Negative	76	54	9	65	5
39a	Negative	Positive	69	54	7	55	6
39b	Negative	Positive	48	35	14	38	6
40	Negative	Negative	43	31	7	36	6
41a	Positive	Negative	61	38	5	44	10
41b	Negative	Positive	50	26	6	31	4
42	Negative	Negative	50	30	17	38	6
43	Negative	Negative	88	64	13	72	6
44	Negative	Positive	54	38	13	44	3
45	Positive	Negative	48	34	8	31	3
46a	Negative	Positive	74	50	1	59	10
46b	Negative	Positive	45	35	7	34	6
47	Positive	Positive	47	39	5	36	7
48	Negative	Negative	48	31	7	32	6
49a	Positive	Positive	44	36	9	36	4
49b	Positive	Positive	26	19	5	23	4
50a	Positive	Positive	49	32	7	32	9
50b	Negative	Positive	65	49	11	43	7
51	Negative	Positive	50	23	10	32	4

LPA187 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
1a	1	1	0	0	0	0	1	0	1	1	8	1	2	1	3	28
1b	0	3	6	0	0	0	0	2	2	5	3	2	0	0	1	17
2	1	2	1	1	0	0	1	3	1	3	4	1	0	2	9	16
3	1	2	8	0	0	0	0	1	2	6	11	1	0	0	0	10
4	0	0	4	1	0	0	1	0	2	4	25	1	0	0	4	8
5	1	0	1	0	0	0	0	0	4	3	19	7	1	0	5	15
6	1	3	2	0	0	0	0	0	1	3	19	7	0	0	3	11
7	0	0	5	2	0	0	0	0	2	10	14	1	1	1	11	9
8a	0	1	2	0	0	0	1	0	1	9	6	5	0	1	4	12
8b	1	2	3	2	0	0	1	0	3	3	16	1	1	2	2	4
9a	1	0	3	1	0	0	0	0	1	8	13	1	0	0	3	24
9b	0	0	3	0	0	0	0	0	4	2	13	4	0	0	3	16
10	0	2	2	0	0	0	0	0	2	2	26	2	0	0	15	5
11	1	0	1	0	0	0	0	0	1	2	16	3	0	0	10	9
12	0	0	4	0	0	0	0	0	2	1	35	8	0	0	3	12
13a	1	1	0	0	0	0	0	0	3	2	7	2	0	1	8	6
13b	3	0	2	0	0	0	1	0	5	4	12	4	0	0	2	2
13c	0	0	4	1	0	0	0	1	1	2	9	0	2	0	2	3
14	0	2	3	0	0	1	1	0	0	1	7	2	0	0	3	3
15	1	0	2	0	0	0	0	0	0	3	21	1	0	0	3	8
16	1	1	6	0	0	0	0	0	0	6	8	0	0	0	7	6
17	0	3	4	3	0	0	0	0	4	1	25	3	1	0	3	16
18	0	3	4	2	0	0	0	0	5	7	20	8	0	0	7	13

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
19	1	2	1	2	1	0	0	0	1	7	15	5	0	0	2	6
20	0	1	4	0	0	0	0	0	4	4	27	9	0	0	4	8
21	1	2	2	0	0	0	0	0	1	9	31	5	0	1	6	11
22	1	1	6	0	0	0	0	0	2	2	14	2	0	0	0	8
23a	1	0	5	0	0	0	0	0	0	3	22	8	0	0	3	14
23b	0	0	4	1	0	0	0	0	3	5	4	3	0	0	3	9
24	0	1	6	2	0	0	0	0	2	5	20	2	0	0	5	7
25	0	1	3	0	0	0	0	1	5	6	28	7	0	0	0	14
26	0	2	4	0	0	0	1	0	1	12	13	5	0	0	4	15
27	0	2	1	1	0	0	0	1	2	8	43	5	0	0	7	11
28a	0	1	1	0	0	0	0	0	3	6	17	2	0	0	3	8
28b	1	0	1	0	0	0	0	0	0	4	20	2	0	0	13	7
28c	0	1	0	1	0	0	0	0	2	2	19	1	0	0	7	10
29	0	0	4	0	0	0	0	0	1	5	26	2	0	0	2	13
30a	0	0	0	0	0	0	0	1	1	3	14	4	0	0	2	4
30b	0	1	2	1	0	0	1	0	0	2	31	1	0	0	4	14
30c	0	1	1	0	0	0	0	1	0	1	6	3	0	0	0	5
31	1	0	1	0	0	0	1	0	4	1	22	4	0	0	1	16
32a	1	1	3	0	0	0	0	0	0	0	21	2	0	0	1	13
32b	0	0	4	2	0	0	0	1	0	2	14	3	0	0	1	15
33a	0	0	3	0	0	0	0	0	2	8	22	2	0	0	1	3
33b	0	1	6	0	0	0	1	0	1	6	32	4	0	0	4	15
34a	0	0	4	0	0	0	0	0	2	0	23	4	0	0	0	10
34b	0	0	4	4	0	0	0	0	1	6	10	0	0	0	2	8
34c	0	0	2	0	0	0	0	0	1	2	1	2	0	0	1	9
34d	1	0	1	0	0	0	0	1	1	3	6	3	0	0	1	6
35	2	0	3	0	0	0	0	0	4	9	37	8	0	0	0	13
36	0	0	6	1	0	0	1	0	2	2	41	3	0	0	2	13
37	0	0	4	2	0	0	0	0	2	2	18	8	0	0	1	35

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
38	0	1	4	0	0	0	0	0	2	6	42	10	0	0	1	10
39a	0	2	3	0	0	0	1	0	0	5	40	5	0	0	3	10
39b	0	2	4	0	0	0	0	0	4	8	15	5	0	0	6	4
40	0	2	4	0	0	0	0	0	1	4	20	5	0	0	1	6
41a	1	3	5	1	0	0	0	0	1	0	28	5	0	0	2	15
41b	0	0	3	1	0	0	0	0	3	3	18	3	0	0	2	17
42	0	3	1	1	0	0	1	0	7	7	15	4	0	0	3	8
43	0	2	4	0	0	0	0	0	4	7	45	10	0	0	6	10
44	1	1	1	0	0	0	0	0	5	6	28	2	0	0	2	8
45	0	2	0	0	0	0	1	0	3	3	21	2	0	0	7	9
46a	0	1	5	2	0	0	1	1	0	0	41	10	0	0	2	11
46b	0	0	4	2	0	0	0	0	3	4	19	2	0	0	8	3
47	0	1	6	0	0	0	0	0	0	4	24	1	0	0	4	7
48	1	1	1	2	0	0	0	1	0	3	20	4	2	0	6	7
49a	0	0	4	0	0	0	0	0	0	9	22	1	0	0	1	7
49b	0	0	2	2	0	0	0	0	1	4	12	2	0	0	1	2
50a	3	1	3	2	0	0	0	0	1	2	16	4	0	0	10	7
50b	0	4	1	2	0	0	0	0	1	5	28	2	1	0	11	10
51	0	1	2	1	0	0	0	0	5	4	15	4	0	0	1	17

1.3.6. **LPA245 count and convergence data**

LPA245 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Positive	Negative	69	49	11	41	4
2	Positive	Negative	57	43	8	35	3
3	Positive	Positive	19	15	2	10	7
4	Negative	Positive	50	27	2	22	9
5	Negative	Negative	39	22	3	28	4
6	Positive	Negative	43	33	4	20	3
7	Negative	Positive	18	16	2	12	2
8	Negative	Negative	33	24	6	14	3
9	Positive	Negative	51	40	10	31	4
10	Positive	Negative	26	25	2	18	3
11	Positive	Positive	38	31	4	14	3
12	Positive	Negative	44	34	4	17	3
13	Negative	Positive	65	44	6	14	2
14	Positive	Positive	22	17	0	8	3
15	Negative	Positive	26	19	5	14	4
16	Positive	Negative	15	12	5	6	3
17	Positive	Positive	33	28	4	14	2
18	Positive	Negative	21	18	2	10	1
19	Positive	Negative	55	40	12	30	4
20	Negative	Negative	30	22	4	15	2
21	Negative	Negative	35	26	2	23	5
22	Negative	Positive	46	32	4	18	5
23	Positive	Negative	20	13	1	8	3
24	Negative	Negative	77	50	15	31	4
25	Negative	Negative	46	26	6	25	2
26	Negative	Negative	48	39	6	24	6
27	Negative	Positive	62	52	17	28	5
28	Negative	Negative	66	48	12	27	6
29	Negative	Negative	51	30	16	21	4
30	Negative	Positive	50	33	5	28	3
31	Negative	Negative	39	29	8	18	5
32	Positive	Positive	30	21	1	17	2
33	Positive	Negative	46	30	5	32	1
34	Positive	Positive	45	28	1	27	3
35	Positive	Negative	32	23	4	22	4
36	Positive	Positive	19	14	2	10	5
37	Positive	Negative	32	26	6	19	3
38	Positive	Negative	41	29	9	23	3
39	Positive	Negative	51	42	14	36	3
40	Negative	Positive	36	30	6	17	3

41	Negative	Positive	78	59	24	35	7
42	Negative	Negative	61	37	15	44	4
43	Negative	Negative	68	40	12	46	3
44	Positive	Positive	32	25	7	19	3
45	Positive	Negative	51	40	17	29	6
46	Positive	Positive	85	39	13	55	10
47	Positive	Negative	27	19	6	19	3

LPA245 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	0	3	0	0	0	0	1	1	10	22	5	0	0	14	13
2	0	0	3	0	0	0	0	0	0	8	24	0	0	0	8	14
3	0	0	4	0	0	0	3	0	2	0	4	0	0	0	4	2
4	0	0	3	0	0	0	1	5	0	2	7	10	0	0	14	8
5	0	0	1	2	0	0	0	1	0	3	18	4	0	0	0	10
6	0	0	1	0	0	0	2	0	0	4	13	2	0	0	13	8
7	0	0	2	0	0	0	0	0	0	2	7	1	0	0	5	1
8	0	0	2	0	0	0	0	1	0	3	9	0	3	0	10	5
9	0	0	2	1	0	0	1	0	1	9	17	1	0	0	11	8
10	0	0	2	0	0	0	1	0	0	2	13	1	0	0	7	0
11	0	1	2	0	0	0	0	0	0	1	9	1	0	2	16	6
12	0	0	0	0	0	0	1	2	1	1	14	1	1	1	17	5
13	1	0	1	0	0	0	0	0	0	2	10	0	0	3	28	20
14	0	0	2	1	0	0	0	0	0	0	3	2	0	0	12	2
15	0	0	2	0	0	0	1	1	0	2	9	1	2	1	4	3
16	0	0	2	0	0	0	1	0	0	0	3	1	0	5	1	2
17	0	1	1	0	0	0	0	0	0	2	9	1	0	1	14	4
18	0	0	0	0	0	0	0	1	0	1	8	1	0	1	8	1
19	0	0	2	0	0	0	2	0	2	10	14	2	0	0	12	11
20	0	0	2	0	0	0	0	0	0	2	10	1	1	1	7	6
21	0	0	2	3	0	0	0	0	0	2	16	0	0	0	6	6
22	0	1	1	0	0	0	1	2	1	2	11	2	0	0	16	9
23	0	0	2	0	0	0	1	0	0	0	5	1	1	0	5	5

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
24	0	0	2	0	0	0	1	1	1	9	14	5	1	4	20	19
25	0	0	0	0	0	0	1	1	0	1	13	11	4	1	10	4
26	0	3	2	0	0	0	1	0	0	2	14	3	0	1	16	6
27	0	2	3	0	0	0	0	0	2	6	14	1	3	4	23	4
28	0	0	4	0	0	0	2	0	1	5	13	4	3	3	21	10
29	1	0	1	1	1	0	0	0	1	4	9	4	3	6	10	10
30	0	0	3	0	0	0	0	0	2	3	17	3	0	0	10	12
31	0	2	2	0	0	0	0	1	0	3	7	4	1	2	13	4
32	0	0	2	0	0	0	0	0	0	1	13	1	0	0	5	8
33	0	0	0	0	0	0	0	1	2	3	21	6	0	0	6	7
34	0	0	2	0	0	0	1	0	0	1	18	6	0	0	6	11
35	0	1	1	0	0	0	1	1	1	2	15	2	0	0	3	5
36	0	1	3	1	0	0	0	0	0	1	4	0	0	0	5	4
37	0	2	1	0	0	0	0	0	1	2	12	1	0	1	8	4
38	0	2	1	0	0	0	0	0	4	2	14	0	1	0	10	7
39	0	3	0	0	0	0	0	0	1	10	18	4	0	0	11	4
40	0	0	2	0	0	0	1	0	1	1	10	3	0	4	12	2
41	0	3	1	1	1	0	1	0	2	6	21	1	8	4	23	6
42	0	0	2	2	0	0	0	0	2	12	20	6	0	1	2	14
43	0	1	0	1	0	0	0	1	3	5	31	5	2	1	2	16
44	0	0	3	0	0	0	0	0	0	5	11	0	0	2	4	7
45	1	1	1	1	1	0	1	0	3	7	13	2	0	4	13	3
46	0	0	4	2	1	0	1	2	3	4	25	17	5	0	5	16
47	0	1	2	0	0	0	0	0	1	4	11	0	0	0	1	7

1.3.7. **LPA281 count and convergence data**

LPA281 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Positive	Positive	38	31	4	35	4
2	Negative	Positive	53	49	11	47	5
3	Negative	Negative	38	30	7	30	6
4	Negative	Positive	51	38	10	44	5
5a	Negative	Negative	52	38	13	46	5
5b	Positive	Positive	40	34	6	35	3
6	Positive	Negative	59	45	18	47	5
7	Positive	Positive	72	51	23	59	7
8	Negative	Positive	54	39	14	49	6
9	Positive	Positive	50	32	6	39	3
10	Positive	Negative	49	35	13	40	7
11	Negative	Positive	49	31	6	33	7
12a	Positive	Positive	41	25	8	29	5
12b	Positive	Negative	64	41	6	44	6
12c	Negative	Positive	37	26	10	28	9
13a	Negative	Positive	44	25	3	30	6
13b	Positive	Positive	35	22	2	26	4
14a	Positive	Positive	37	17	5	20	7
14b	Positive	Positive	29	17	6	24	7
15	Positive	Positive	45	35	8	38	6
16	Positive	Positive	59	49	12	53	5
17	Negative	Positive	73	56	14	60	5
18	Positive	Positive	82	64	8	69	6
19	Negative	Positive	50	34	2	33	6
20a	Positive	Positive	41	31	11	35	2
20b	Positive	Positive	38	35	14	35	2
21	Negative	Negative	32	29	3	30	5
22	Positive	Positive	39	29	11	36	6
23	Negative	Positive	42	32	11	37	3
24	Negative	Positive	68	47	11	55	6
25	Negative	Positive	63	48	8	49	5
26a	Negative	Positive	39	30	7	31	3
26b	Negative	Positive	39	35	7	35	3
27	Negative	Positive	64	34	10	45	8
28	Negative	Positive	45	30	6	31	4
29a	Positive	Positive	28	17	0	21	3
29b	Positive	Positive	45	24	6	34	7
29c	Positive	Positive	40	23	8	30	2
30	Positive	Negative	43	26	3	28	3
31	Negative	Positive	46	35	2	35	8

32	Negative	Positive	64	55	12	53	4
33a	Negative	Positive	57	36	17	46	4
33b	Positive	Positive	48	35	13	40	3
34	Negative	Negative	54	38	22	50	5
35	Negative	Positive	66	50	5	50	1
36	Positive	Positive	43	27	7	33	3
37	Negative	Positive	57	36	21	50	5
38	Negative	Positive	45	32	8	31	3
39a	Negative	Negative	39	31	0	31	2
39b	Positive	Negative	41	31	7	36	4
39c	Negative	Positive	30	22	10	27	2
40	Positive	Positive	53	43	15	45	4
41	Negative	Positive	47	39	5	35	5
42a	Negative	Negative	56	43	10	44	5
42b	Negative	Positive	46	35	6	39	3
43	Negative	Positive	71	52	12	60	4
44a	Positive	Positive	46	31	8	36	5
44b	Negative	Positive	39	35	7	37	4
45	Positive	Positive	27	14	2	22	4
46	Positive	Positive	51	39	14	43	7
47	Positive	Negative	56	40	13	41	7
48	Negative	Positive	60	49	22	52	5
49	Negative	Positive	55	44	23	52	7
50	Negative	Positive	73	68	17	63	9
51	Negative	Positive	67	55	18	60	3
52	Positive	Negative	51	35	10	42	5
53	Negative	Negative	53	39	14	41	4
54a	Negative	Positive	34	22	5	25	6
54b	Negative	Positive	19	13	1	14	6
54c	Positive	Positive	28	16	5	21	5

LPA281 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
1	0	2	2	0	0	0	0	0	1	1	26	3	0	0	0	3
2	0	3	2	0	0	0	0	0	0	8	34	0	0	0	2	4
3	0	3	3	0	0	0	0	0	0	4	20	0	0	0	0	8
4	0	2	2	1	0	0	0	0	4	4	28	3	0	0	2	5
5a	0	1	4	0	0	0	0	0	4	8	25	4	0	0	0	6
5b	0	1	1	1	0	0	0	0	1	4	27	0	0	0	1	4
6	0	0	4	0	0	0	1	0	1	17	21	4	0	0	2	9
7	0	5	2	0	0	0	0	0	7	11	31	3	0	0	2	11
8	0	1	3	2	0	0	0	0	5	8	27	3	0	0	0	5
9	0	1	2	0	0	0	0	0	2	3	25	6	0	0	1	10
10	3	1	2	1	0	0	0	0	0	9	22	2	0	0	1	8
11	0	2	5	0	0	0	0	0	0	4	20	2	0	0	0	16
12a	3	0	2	0	0	0	0	0	0	5	18	1	0	0	0	12
12b	0	0	5	1	0	0	0	0	0	6	28	4	0	0	2	18
12c	0	4	5	0	0	0	0	0	0	6	11	2	0	0	0	9
13a	0	2	4	0	0	0	0	0	1	0	18	5	0	0	1	13
13b	0	0	4	0	0	0	0	0	0	2	16	4	0	0	0	9
14a	1	2	3	1	0	0	0	0	0	2	10	1	0	0	0	17
14b	0	0	2	5	0	0	0	0	0	6	8	3	0	0	1	4
15	3	0	3	0	0	0	0	0	0	5	26	1	0	0	1	6
16	0	0	5	0	0	0	0	0	2	10	34	2	0	0	0	6
17	1	2	2	0	0	0	0	0	4	7	41	3	0	0	4	9
18	0	1	4	1	0	0	0	0	0	7	50	6	0	0	2	11

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
19	0	0	4	2	0	0	0	0	1	1	23	2	0	0	6	11
20a	0	1	1	0	0	0	0	0	3	7	21	2	0	0	1	5
20b	0	0	2	0	0	0	0	0	0	14	18	1	0	0	1	2
21	0	0	5	0	0	0	0	0	0	3	21	1	0	0	0	2
22	0	0	5	1	0	0	0	0	1	10	14	5	0	0	0	3
23	0	1	2	0	0	0	0	0	4	5	23	2	1	0	1	3
24	0	3	3	0	0	0	0	0	3	5	35	6	0	0	1	12
25	0	2	2	1	0	0	0	0	3	3	36	2	0	0	5	9
26a	0	1	2	0	0	0	0	0	1	4	22	1	1	0	1	6
26b	0	0	2	0	0	0	1	0	2	5	24	2	0	0	3	0
27	0	3	4	1	0	0	0	0	3	4	23	7	0	0	0	19
28	0	3	1	0	0	0	0	0	0	3	23	1	0	0	0	14
29a	0	0	3	0	0	0	0	0	0	0	14	4	0	0	0	7
29b	0	2	3	2	0	0	0	0	1	3	16	7	0	0	0	11
29c	0	1	1	0	0	0	0	0	0	7	14	7	0	0	0	10
30	0	1	2	0	0	0	0	0	0	2	21	2	0	0	0	15
31	0	0	7	1	0	0	0	0	0	2	22	3	0	0	4	7
32	0	0	4	0	0	0	0	0	2	10	37	0	0	0	4	7
33a	1	1	1	1	0	0	0	0	5	10	23	4	0	0	1	10
33b	0	0	3	0	0	0	0	0	4	9	22	2	0	0	1	7
34	0	0	5	0	0	0	0	0	12	10	20	3	0	0	3	1
35	0	0	1	0	0	0	0	0	1	4	41	3	0	0	4	12
36	0	0	3	0	0	0	0	0	5	2	19	4	0	0	3	7
37	1	2	0	2	0	0	0	0	9	9	23	4	0	0	2	5
38	0	0	3	0	0	0	0	0	0	7	19	2	0	1	2	11
39a	0	0	2	0	0	0	0	0	0	0	26	3	0	0	3	5
39b	0	1	3	0	0	0	0	0	4	2	24	2	0	0	1	4
39c	1	0	1	0	0	0	0	0	5	4	16	0	0	0	1	2
40	0	1	1	1	0	0	1	0	1	12	24	5	1	0	4	2

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
41	0	2	3	0	0	0	0	0	1	2	26	1	0	0	6	6
42a	0	1	3	1	0	0	0	0	1	8	28	2	0	0	3	9
42b	0	1	2	0	0	0	0	0	2	3	29	2	0	0	0	7
43	1	0	3	0	0	0	0	0	1	10	39	6	0	0	0	11
44a	0	4	1	0	0	0	0	0	0	4	22	5	0	0	0	10
44b	0	1	3	0	0	0	0	0	0	6	25	2	0	0	0	2
45	0	2	1	1	0	0	0	0	0	0	10	8	0	0	1	4
46	0	1	4	2	0	0	0	0	4	9	21	2	0	0	4	4
47	0	3	3	1	0	0	0	0	3	7	21	3	0	0	6	9
48	0	3	2	0	0	0	0	0	2	17	26	2	0	0	1	7
49	1	3	3	0	0	0	0	0	5	14	24	2	0	0	0	3
50	0	6	2	1	0	0	0	0	2	9	42	1	0	0	9	1
51	0	1	2	0	0	0	0	0	5	12	39	1	0	0	1	6
52	0	1	3	1	0	0	0	0	3	6	23	5	0	0	2	7
53	0	1	3	0	0	0	0	0	3	10	24	0	0	0	1	11
54a	2	1	2	1	0	0	0	0	0	2	17	0	0	0	0	9
54b	0	1	5	0	0	0	0	0	0	0	7	1	0	0	0	5
54c	0	1	3	1	0	0	0	0	1	3	9	3	0	0	0	7

1.3.8. **LPA247 count and convergence data**

LPA247 count data

					Total		
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Positive	Negative	56	42	10	43	6
2a	Negative	Positive	43	38	6	38	4
2b	Negative	Positive	43	34	9	38	4
2c	Positive	Positive	49	35	10	41	2
3a	Negative	Positive	37	30	12	29	6
3b	Positive	Positive	36	24	8	29	5
4	Negative	Positive	45	40	17	42	9
5	Negative	Positive	67	53	23	56	6
6	Positive	Positive	64	53	14	58	8
7	Positive	Positive	48	34	8	42	10
8	Negative	Positive	72	57	10	64	8
9a	Negative	Positive	45	34	10	39	6
9b	Negative	Negative	41	27	8	33	6
10a	Negative	Positive	50	39	15	40	5
10b	Negative	Positive	47	23	2	24	4
11	Negative	Negative	45	37	9	42	6
12	Positive	Positive	67	51	16	55	7
13	Positive	Positive	45	35	13	35	4
14	Positive	Positive	45	35	8	40	10
15	Positive	Positive	38	24	11	33	8
16a	Negative	Positive	42	30	9	37	6
16b	Positive	Positive	44	32	10	35	7
17a	Negative	Negative	44	17	6	35	7
17b	Positive	Negative	36	27	5	26	3
18a	Positive	Positive	48	39	4	42	5
18b	Positive	Positive	42	37	10	39	4
19a	Negative	Positive	39	34	7	35	3
19b	Negative	Positive	18	15	3	16	3
20a	Negative	Positive	43	36	7	38	4
20b	Negative	Positive	59	45	17	53	4
21a	Positive	Positive	51	39	13	46	4
21b	Negative	Positive	54	39	10	45	4
22a	Negative	Positive	46	29	10	36	4
22b	Positive	Positive	53	48	10	51	3
23	Negative	Negative	60	55	11	54	5
24a	Positive	Positive	46	34	6	27	6
24b	Negative	Positive	34	16	15	24	4
24c	Positive	Positive	35	19	9	21	4
25	Negative	Positive	46	43	12	26	6
26	Positive	Positive	45	34	5	33	3

27a	Negative	Positive	43	36	12	22	6
27b	Positive	Positive	41	33	4	21	2
27c	Negative	Positive	40	29	8	18	2
28a	Negative	Positive	45	30	4	22	6
28b	Negative	Positive	41	29	4	22	2
29a	Negative	Negative	53	42	10	20	6
29b	Positive	Positive	48	38	9	24	5
30	Positive	Positive	46	42	11	29	6
31a	Positive	Positive	44	30	3	24	3
31b	Positive	Positive	65	41	11	34	4
32	Positive	Positive	42	33	4	30	7
33	Negative	Positive	48	39	8	42	3
34	Negative	Positive	49	38	10	40	4
35	Positive	Positive	47	43	8	30	4
36	Negative	Positive	56	46	9	34	4
37	Positive	Positive	43	34	7	25	2
38	Negative	Negative	55	44	9	39	3
39	Positive	Positive	46	18	11	19	2
40	Positive	Positive	44	33	7	18	4
41	Positive	Positive	43	35	4	22	5
42	Positive	Positive	60	42	17	47	5
43	Positive	Positive	58	42	5	46	6
44a	Negative	Positive	59	46	9	37	7
44b	Negative	Positive	50	35	9	36	4
45	Positive	Positive	54	33	6	21	6
46	Positive	Positive	46	37	8	23	2
47	Negative	Positive	57	41	8	38	6
48	Negative	Positive	73	64	24	36	8
49	Negative	Positive	49	33	10	27	5
50	Negative	Positive	45	37	7	33	6

LPA247 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	3	1	1	0	0	0	1	2	5	28	3	0	0	5	7
2a	0	0	3	1	0	0	0	0	1	4	29	0	1	0	2	2
2b	1	1	2	0	0	0	0	0	1	6	24	3	0	0	1	4
2c	0	1	1	0	0	0	0	0	1	8	24	6	0	0	1	7
3a	1	3	2	0	0	0	0	0	0	8	15	0	0	0	2	6
3b	4	0	1	0	0	0	0	0	0	4	18	2	0	0	1	6
4	0	2	7	0	0	0	0	0	2	13	18	0	0	0	0	3
5	0	2	4	0	0	0	0	0	0	20	26	4	0	1	0	10
6	0	0	8	0	0	0	0	0	3	10	34	3	1	0	1	4
7	0	3	7	0	0	0	0	0	1	4	20	7	0	0	0	6
8	0	1	7	0	0	0	0	0	1	8	40	7	0	0	1	7
9a	0	2	4	0	0	0	0	0	1	7	20	5	0	0	1	5
9b	0	3	3	0	0	0	0	0	1	4	17	5	0	0	0	8
10a	0	2	3	0	0	0	0	0	1	12	22	0	0	0	0	10
10b	0	1	1	2	0	0	0	0	0	1	19	0	0	0	1	22
11	0	2	3	1	0	0	0	0	3	4	28	1	0	0	0	3
12	1	3	3	0	0	0	0	0	3	9	35	1	0	0	1	11
13	0	2	2	0	0	0	0	0	2	9	20	0	0	0	2	8
14	0	1	8	1	0	0	0	0	0	7	18	5	0	0	1	4
15	1	5	2	0	0	0	0	0	4	1	16	4	0	0	0	5
16a	0	2	3	1	0	0	0	0	3	4	20	4	0	0	1	4
16b	0	2	3	2	0	0	0	0	2	6	20	0	0	0	1	8
17a	0	3	3	1	0	0	0	0	0	3	1	24	0	0	7	2

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
17b	0	0	3	0	0	0	0	0	1	4	17	1	0	0	3	7
18a	0	0	5	0	0	0	0	0	1	3	29	4	0	0	2	4
18b	0	0	4	0	0	0	0	0	0	10	23	2	0	0	0	3
19a	0	1	1	1	0	0	0	0	0	6	24	2	0	0	2	2
19b	0	1	2	0	0	0	0	0	0	2	10	1	0	0	0	2
20a	1	0	2	0	0	0	0	1	0	6	28	1	0	0	0	4
20b	1	2	1	0	0	0	0	0	2	12	29	6	0	0	1	5
21a	2	2	0	0	0	0	0	0	2	7	29	4	0	0	1	4
21b	0	2	2	0	0	0	0	0	3	5	29	4	0	0	1	8
22a	2	1	1	0	0	0	0	0	4	3	22	3	0	0	2	8
22b	0	1	2	0	0	0	0	0	0	9	36	3	0	0	0	2
23	0	4	1	0	0	0	0	0	0	7	42	0	0	0	1	5
24a	0	1	5	0	0	0	0	0	1	4	15	1	0	0	9	10
24b	2	2	0	0	0	0	0	0	8	3	6	3	0	0	5	5
24c	0	1	2	0	0	0	1	0	2	6	6	4	0	0	3	10
25	0	2	3	1	0	0	0	0	0	9	9	2	0	1	19	0
26	0	0	3	0	0	0	0	0	1	4	21	4	0	0	6	6
27a	0	3	1	1	0	0	1	0	0	9	8	0	0	0	14	6
27b	0	0	2	0	0	0	0	0	0	3	11	5	0	1	16	3
27c	0	1	1	0	0	0	0	0	1	5	7	3	0	1	14	7
28a	0	1	5	0	0	0	0	0	0	3	10	3	0	0	11	12
28b	0	0	2	0	0	0	0	0	0	4	14	2	0	0	9	10
29a	0	2	3	1	0	0	0	0	1	6	6	1	0	1	24	8
29b	0	2	3	0	0	0	0	0	2	5	9	3	0	0	19	5
30	0	0	6	0	0	0	0	0	0	11	11	1	0	0	14	3
31a	0	0	1	1	0	0	1	0	0	3	17	2	0	0	8	11
31b	1	1	1	1	0	0	0	0	1	8	18	3	0	0	13	18
32	0	1	3	3	0	0	0	0	2	1	18	2	0	0	10	2
33	0	1	2	0	0	0	0	0	3	4	28	4	0	0	4	2

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
34	0	0	2	2	0	0	0	0	3	7	21	5	0	0	8	1
35	0	0	4	0	0	0	0	0	0	8	17	1	0	0	14	3
36	1	0	3	0	0	0	0	0	3	5	21	1	0	0	17	5
37	0	1	0	0	0	0	1	0	0	6	16	2	0	0	10	7
38	0	1	2	0	0	0	0	0	2	6	26	2	0	0	9	7
39	2	0	0	0	0	0	0	0	6	3	4	4	0	0	11	16
40	1	1	2	0	0	0	0	0	0	5	9	0	0	0	16	10
41	0	2	3	0	0	0	0	0	0	2	12	3	0	0	16	5
42	0	2	2	0	0	0	1	0	4	11	20	8	0	0	6	6
43	0	1	5	0	0	0	0	0	0	4	29	7	0	0	3	9
44a	1	1	4	1	0	0	0	0	1	6	21	2	0	0	14	8
44b	0	3	1	0	0	0	0	0	0	6	18	8	0	0	7	7
45	1	2	3	0	0	0	0	0	1	2	9	3	0	0	17	16
46	0	0	2	0	0	0	0	0	1	7	9	4	0	0	19	4
47	0	1	3	2	0	0	0	0	2	5	20	5	0	0	12	7
48	0	7	1	0	0	0	0	0	3	11	13	1	0	3	29	5
49	0	2	2	1	0	0	0	0	1	7	12	2	0	0	10	12
50	0	2	4	0	0	0	0	0	0	4	21	2	0	1	5	6

1.3.9. **LPA280 count and convergence data**

LPA280 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Negative	Positive	29	23	10	27	8
2a	Negative	Positive	61	53	14	55	10
2b	Positive	Positive	49	41	4	42	3
3a	Positive	Negative	65	47	13	54	6
3b	Negative	Positive	53	46	7	48	6
4a	Negative	Positive	47	34	9	36	6
4b	Positive	Positive	53	40	3	42	4
5	Negative	Positive	54	50	6	51	3
6a	Negative	Negative	48	39	11	43	6
6b	Positive	Negative	41	30	1	36	6
7a	Positive	Positive	51	40	4	41	5
7b	Negative	Negative	31	22	12	28	5
8	Positive	Positive	62	47	6	49	6
9a	Negative	Positive	70	54	5	58	5
9b	Negative	Positive	57	28	11	47	7
10	Negative	Positive	60	45	6	48	12
11a	Negative	Positive	74	62	8	63	1
11b	Negative	Negative	55	45	12	44	6
12a	Positive	Positive	48	40	6	42	3
12b	Positive	Positive	40	24	8	36	3
13a	Positive	Positive	24	24	5	24	4
13b	Positive	Positive	26	24	5	24	6
13c	Positive	Positive	41	33	13	38	6
14	Negative	Positive	40	29	5	34	8
15a	Positive	Positive	42	25	4	32	4
15b	Negative	Positive	42	28	3	35	8
16	Negative	Positive	50	32	7	42	5
17	Negative	Positive	61	53	12	53	5
18	Negative	Negative	69	47	17	57	8
19	Negative	Positive	61	48	11	55	6
20	Positive	Positive	43	23	15	38	9
21	Negative	Positive	39	34	4	31	5
22a	Positive	Positive	43	35	1	35	3
22b	Negative	Positive	32	29	2	29	5
23a	Negative	Positive	44	30	3	40	4
23b	Negative	Positive	64	51	8	52	3
24	Negative	Positive	51	46	4	47	4
25a	Negative	Positive	36	30	2	31	4
25b	Positive	Positive	37	30	3	33	6
25c	Positive	Positive	35	27	3	32	5

25d	Negative	Positive	41	34	8	36	4
26a	Positive	Positive	56	44	8	48	5
26b	Negative	Negative	26	23	4	24	2
27a	Negative	Positive	32	27	8	27	3
27b	Positive	Positive	46	38	7	40	2
27c	Negative	Positive	51	34	4	40	9
27d	Positive	Negative	34	28	2	29	3
28	Negative	Positive	55	30	18	40	3
29a	Positive	Positive	62	53	14	57	7
29b	Positive	Negative	45	26	12	35	4
30a	Negative	Positive	57	40	18	47	4
31a	Negative	Positive	31	26	8	26	3
31b	Negative	Positive	24	22	4	19	5
32	Negative	Negative	47	41	9	40	3
33	Positive	Positive	58	51	10	53	2
34a	Positive	Negative	56	43	8	46	7
34b	Positive	Positive	51	32	8	45	9
35	Negative	Positive	47	39	4	42	7
36a	Positive	Positive	57	42	16	52	5
36b	Positive	Positive	49	30	19	39	6
37	Negative	Negative	49	41	15	44	7
38	Positive	Positive	64	44	5	56	10
39	Negative	Positive	76	52	10	61	6
40a	Positive	Positive	41	23	6	29	8
40b	Positive	Positive	52	37	14	43	9
40c	Negative	Positive	41	22	8	27	6

LPA280 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	5	3	0	0	0	0	0	0	5	10	4	0	0	0	2
2a	0	2	8	0	0	0	0	0	2	10	28	5	0	0	5	1
2b	0	1	1	0	0	0	1	0	0	3	33	4	0	0	2	4
3a	2	1	3	0	0	0	0	0	3	7	32	6	0	0	4	7
3b	0	3	3	0	0	0	0	0	0	4	36	2	0	0	0	5
4a	0	0	5	1	0	0	0	0	2	7	21	0	0	0	1	10
4b	0	0	4	0	0	0	0	0	0	3	33	2	0	0	0	11
5	0	1	2	0	0	0	0	0	0	5	42	1	0	0	0	3
6a	0	0	6	0	0	0	0	0	2	9	22	4	0	0	2	3
6b	0	0	5	1	0	0	0	0	0	1	24	5	0	0	0	5
7a	0	1	4	0	0	0	0	0	1	2	31	2	0	0	2	8
7b	0	2	3	0	0	0	0	0	4	6	11	2	0	0	0	3
8	0	1	5	0	0	0	0	0	1	4	31	7	0	0	6	7
9a	0	0	4	1	0	0	0	0	0	5	43	5	0	0	2	10
9b	2	0	4	1	0	0	0	0	8	1	22	9	0	0	1	9
10	2	0	10	0	0	0	0	0	1	3	28	4	0	0	4	8
11a	0	1	0	0	0	0	0	0	4	3	52	3	0	0	6	5
11b	0	3	3	0	0	0	0	0	1	8	29	0	0	0	2	9
12a	0	1	2	0	0	0	0	0	1	4	32	2	0	0	1	5
12b	0	1	2	0	0	0	0	0	3	4	16	10	0	0	1	3
13a	0	0	4	0	0	0	0	0	0	5	15	0	0	0	0	0
13b	0	1	5	0	0	0	0	0	0	4	14	0	0	0	0	2
13c	1	3	1	1	0	0	0	0	1	8	21	2	0	0	0	3

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
14	0	1	6	0	0	0	0	1	2	2	20	3	0	0	0	5
15a	0	0	3	1	0	0	0	0	3	1	19	5	0	0	2	8
15b	0	0	7	1	0	0	0	0	1	2	19	5	0	0	0	7
16	0	0	3	2	0	0	0	0	1	6	23	7	0	0	0	8
17	0	2	2	0	0	0	1	0	3	7	39	0	0	0	2	5
18	0	3	4	1	0	0	0	0	7	7	31	4	0	0	2	10
19	0	2	4	0	0	0	0	0	1	8	33	7	0	0	1	5
20	3	1	3	2	0	0	0	0	8	3	16	2	0	0	0	5
21	0	0	5	0	0	0	0	0	0	3	21	2	0	1	4	3
22a	0	0	1	2	0	0	0	0	0	1	31	0	0	0	2	6
22b	0	0	5	0	0	0	0	0	0	2	20	2	0	0	2	1
23a	1	0	3	0	0	0	0	0	0	2	24	10	0	0	1	3
23b	0	1	2	0	0	0	0	0	0	7	41	1	0	0	0	12
24	0	1	2	1	0	0	0	0	0	3	39	1	0	0	1	3
25a	0	0	4	0	0	0	0	0	0	2	23	2	0	0	1	4
25b	0	0	6	0	0	0	0	0	1	2	22	2	0	0	0	4
25c	0	1	3	1	0	0	0	0	0	2	20	5	0	0	1	2
25d	0	2	2	0	0	0	0	0	3	3	26	0	0	0	1	4
26a	0	1	3	1	0	0	0	0	0	7	31	5	0	0	2	6
26b	0	1	1	0	0	0	0	0	0	3	17	2	0	0	1	1
27a	0	0	3	0	0	0	0	0	0	8	13	3	0	0	3	2
27b	0	0	2	0	0	0	0	0	0	7	27	4	0	0	2	4
27c	0	0	8	1	0	0	0	0	1	2	19	9	0	1	4	6
27d	0	1	2	0	0	0	0	0	1	0	25	0	0	0	0	5
28	1	1	0	0	0	0	1	0	9	7	18	4	0	0	3	11
29a	0	1	5	1	0	0	0	0	3	10	37	0	0	0	0	5
29b	0	1	2	1	0	0	0	0	6	5	15	5	0	0	3	7
30a	0	1	2	1	0	0	0	0	6	10	24	3	1	0	3	6
31a	0	0	2	1	0	0	0	0	1	7	14	1	0	0	3	2

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
31b	0	0	3	1	0	0	1	0	0	3	12	0	0	1	2	1
32	0	0	3	0	0	0	0	0	2	7	28	0	0	0	3	4
33	0	0	2	0	0	0	0	0	0	10	37	4	0	0	2	3
34a	2	0	4	1	0	0	0	0	0	6	30	3	0	0	3	7
34b	2	0	6	1	0	0	0	0	3	3	22	8	0	0	1	5
35	0	0	6	1	0	0	0	0	0	4	29	2	0	0	0	5
36a	0	0	4	1	0	0	0	0	7	8	30	2	1	0	0	4
36b	2	4	0	0	0	0	0	0	2	11	14	6	0	0	1	9
37	1	1	5	0	0	0	0	0	5	8	23	1	0	0	4	1
38	0	1	5	4	0	0	0	0	2	2	33	9	0	0	3	5
39	0	1	5	0	0	0	0	0	3	6	37	9	0	0	3	12
40a	0	4	3	1	0	0	0	0	0	2	13	6	0	0	1	11
40b	1	1	4	3	0	0	0	0	3	9	21	1	0	0	2	7
40c	0	1	3	2	0	0	0	0	1	6	12	2	0	0	0	14

1.3.10. **LPA278 count and convergence data**

LPA278 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Negative	Negative	67	56	22	43	8
2	Negative	Negative	66	47	29	52	6
3	Negative	Positive	112	83	27	83	7
4	Negative	Negative	78	62	21	51	7
5	Negative	Negative	48	35	19	36	8
6	Negative	Negative	81	57	27	65	6
7	Negative	Positive	36	27	11	26	7
8	Negative	Positive	64	43	15	42	6
9	Negative	Negative	52	36	18	40	4
10	Negative	Positive	59	46	16	8	8
11	Negative	Positive	55	36	25	39	4
12	Negative	Positive	75	56	14	47	4
13	Negative	Negative	91	60	27	62	2
14	Negative	Positive	52	36	24	49	4
15	Negative	Positive	64	57	26	60	3
16	Negative	Positive	44	36	20	38	3
17	Negative	Negative	71	57	24	66	10
18	Negative	Positive	70	57	27	62	5
19	Negative	Positive	55	45	23	47	7
20	Negative	Positive	53	43	39	45	7
21	Negative	Positive	75	55	8	49	9

LPA278 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	1	7	0	0	0	0	0	2	17	13	3	0	2	16	6
2	4	1	1	0	0	0	0	0	8	15	22	1	0	1	7	6
3	2	2	0	3	0	0	0	0	3	17	48	8	2	1	15	11
4	0	3	4	0	0	0	0	0	0	18	26	0	0	0	11	16
5	0	5	3	0	0	0	0	0	0	12	11	5	2	0	4	6
6	0	4	2	0	0	0	0	0	4	18	32	5	1	0	1	14
7	0	5	2	0	0	0	0	0	0	6	9	4	0	0	5	5
8	1	1	4	0	0	0	0	0	0	12	20	4	0	1	5	16
9	0	0	4	0	0	0	0	0	2	16	16	2	0	0	0	12
10	1	2	3	2	0	0	0	0	0	0	0	0	2	11	30	8
11	1	3	0	0	0	0	0	0	6	11	15	3	0	4	3	9
12	0	1	0	3	0	0	0	0	3	10	25	5	0	0	20	8
13	0	0	2	0	0	0	0	0	7	16	31	6	2	2	9	16
14	1	0	2	1	0	0	0	0	8	15	17	5	0	0	2	1
15	0	2	1	0	0	0	0	0	3	21	30	3	0	0	3	1
16	1	1	1	0	0	0	0	0	5	13	16	1	0	0	5	1
17	0	6	4	0	0	0	0	0	3	15	31	7	0	0	1	4
18	1	2	2	0	0	0	0	0	4	20	30	3	0	0	3	5
19	1	3	3	0	0	0	0	0	1	18	16	5	0	0	5	3
20	0	5	2	0	0	0	0	0	3	28	6	1	3	0	2	3
21	1	1	6	1	0	0	0	0	1	4	34	1	1	0	10	15

1.3.11. **LPA242 count and convergence data**

LPA242 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Negative	Negative	82	72	5	78	8
2	Negative	Negative	70	58	13	55	7
3	Negative	Positive	55	44	11	47	4
4	Negative	Positive	52	39	3	45	7
5	Negative	Negative	95	75	12	87	1
6	Negative	Positive	119	98	22	109	2
7	Negative	Negative	65	53	14	53	4
8	Negative	Negative	68	57	13	60	4
9	Negative	Positive	60	49	20	51	3
10	Negative	Negative	46	36	12	45	3
11	Negative	Positive	76	64	11	69	2
12	Negative	Positive	71	55	24	60	8
13a	Negative	Negative	72	60	11	65	5
13b	Negative	Positive	61	53	7	59	5
14a	Negative	Positive	66	55	14	61	4
14b	Negative	Positive	59	48	6	51	2
15	Negative	Positive	63	45	7	56	5
16	Negative	Negative	82	67	13	77	1
17	Negative	Positive	47	38	11	45	9
18	Negative	Positive	49	37	16	43	2
19	Negative	Positive	51	39	11	46	6
20	Negative	Positive	54	42	7	43	10
21	Negative	Positive	76	57	18	71	5
22	Negative	Negative	48	34	14	40	1
23	Negative	Positive	67	54	7	60	3
24	Negative	Positive	74	49	12	59	3
25	Negative	Positive	50	42	23	49	4
26	Negative	Negative	69	59	12	61	4
27	Negative	Negative	57	37	6	50	5
28	Negative	Negative	71	56	17	63	9
29	Negative	Positive	48	38	10	40	4
30	Negative	Positive	47	30	11	37	7
31	Negative	Negative	63	42	22	45	4
32	Negative	Positive	86	72	6	74	3
33	Negative	Positive	75	59	29	64	5
34	Negative	Positive	79	62	5	72	4
35	Negative	Negative	44	32	10	34	5
36	Negative	Negative	62	45	16	54	7
37	Negative	Positive	55	44	17	46	7
38	Negative	Positive	69	40	19	57	8
39	Negative	Positive	75	52	19	72	8

40	Negative	Negative	40	30	10	34	8

LPA242 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	0	5	3	0	0	0	0	0	5	61	4	0	0	1	3
2	0	4	3	0	0	0	0	0	0	6	38	4	0	3	4	8
3	0	2	2	0	0	0	0	0	2	7	28	6	0	0	5	3
4	1	0	5	1	0	0	0	0	0	2	31	5	0	0	1	6
5	0	0	1	0	0	0	0	0	0	12	60	14	0	0	2	6
6	0	1	1	0	0	0	0	0	1	19	75	12	0	1	1	8
7	0	3	1	0	0	0	0	0	0	11	35	3	0	0	3	9
8	0	2	2	0	0	0	0	0	0	10	38	8	0	1	4	3
9	0	1	2	0	0	0	0	0	2	15	28	3	1	1	2	5
10	0	0	3	0	0	0	0	0	1	11	22	8	0	0	0	1
11	0	1	1	0	0	0	0	0	0	10	51	6	0	0	1	6
12	3	3	2	0	0	0	0	0	3	12	34	3	1	2	2	6
13a	0	2	3	0	0	0	0	0	2	7	47	4	0	0	1	6
13b	0	1	3	1	0	0	0	0	1	5	44	4	0	0	0	2
14a	0	1	2	1	0	0	0	0	1	12	39	5	0	0	1	4
14b	0	0	2	0	0	0	0	0	0	6	37	6	0	0	3	5
15	0	0	4	1	0	0	0	0	2	5	33	11	0	0	3	4
16	0	1	0	0	0	0	0	0	0	12	50	14	0	0	4	1
17	0	0	8	1	0	0	0	0	3	8	22	3	0	0	0	2
18	0	2	0	0	0	0	0	0	2	12	22	5	0	0	1	5
19	0	1	5	0	0	0	0	0	4	6	25	5	0	0	2	3
20	0	3	7	0	0	0	0	0	1	3	28	1	0	0	1	10
21	3	1	1	0	0	0	0	0	4	10	44	8	0	0	1	4

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
22	0	0	0	0	0	0	0	1	4	10	23	3	0	0	1	6
23	0	1	2	0	0	0	0	0	0	6	42	9	0	0	3	4
24	1	0	2	0	0	0	0	0	5	4	38	9	1	1	4	9
25	0	2	2	0	0	0	0	0	3	18	20	4	0	0	0	1
26	0	0	3	0	0	0	0	1	0	12	39	7	0	0	5	2
27	1	1	1	2	0	0	0	0	0	4	30	11	0	0	1	6
28	1	1	7	0	0	0	0	0	2	13	33	6	0	0	2	6
29	0	3	1	0	0	0	0	0	4	3	29	0	0	0	2	6
30	1	1	1	4	0	0	0	0	2	7	18	3	0	0	3	7
31	0	3	0	1	0	0	0	0	7	10	21	3	2	0	8	8
32	0	0	3	0	0	0	0	0	0	6	57	8	0	0	6	6
33	0	5	0	0	0	0	0	0	3	21	31	4	0	0	2	9
34	0	2	2	0	0	0	0	0	0	3	53	12	0	0	2	5
35	1	1	2	0	0	0	1	0	2	6	19	3	0	0	3	6
36	1	2	2	2	0	0	0	0	3	10	29	5	0	0	2	6
37	0	4	3	0	0	0	0	0	0	13	21	5	0	0	3	6
38	0	2	4	2	0	0	0	0	10	7	24	8	0	0	3	9
39	0	0	4	4	0	0	0	0	6	13	34	11	0	0	1	2
40	0	2	2	3	0	0	0	1	0	8	14	5	0	0	4	1

1.3.12. **LPA213 count and convergence data**

LPA213 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1	Negative	Positive	73	69	11	71	6
2a	Negative	Positive	68	57	4	59	3
2b	Negative	Positive	73	59	19	65	8
3	Negative	Positive	73	55	16	64	4
4	Negative	Positive	73	61	13	69	6
5	Negative	Positive	53	41	5	47	2
6	Negative	Positive	82	66	21	75	5
7a	Negative	Positive	56	45	9	49	5
7b	Negative	Positive	41	29	6	35	7
8a	Negative	Positive	48	22	2	29	6
8b	Negative	Positive	39	24	10	30	7
9	Negative	Negative	83	70	24	72	7
10	Negative	Negative	107	81	26	96	6
11	Negative	Positive	31	24	10	30	7
12	Negative	Positive	53	39	10	46	5
13	Negative	Positive	61	51	30	58	8
14	Negative	Negative	75	53	15	68	7
15a	Negative	Positive	49	37	14	46	4
15b	Negative	Negative	81	65	16	75	8
16	Negative	Positive	56	41	13	47	4
17	Negative	Positive	62	54	15	54	6
18	Negative	Positive	74	57	15	58	5
19	Negative	Negative	65	50	10	54	11
20	Negative	Positive	67	57	7	62	6
21a	Negative	Positive	51	43	16	49	6
21b	Negative	Positive	45	34	3	40	6
22	Negative	Negative	96	89	29	85	4
23	Negative	Positive	47	41	15	38	5
24a	Negative	Positive	49	39	10	41	4
24b	Negative	Negative	95	64	12	76	1
25	Negative	Negative	60	50	16	39	4
26	Negative	Positive	66	51	19	48	11
27	Negative	Negative	52	37	11	41	6
28	Negative	Positive	63	43	17	39	10
29a	Negative	Positive	59	45	4	36	8
29b	Negative	Positive	55	46	17	48	5
30	Negative	Negative	92	80	21	42	4
31	Negative	Positive	76	64	13	29	3
32	Negative	Negative	68	63	19	56	4
33	Negative	Positive	69	64	16	39	5

34a	Negative	Negative	73	54	17	56	3
34b	Negative	Negative	63	36	10	45	3
35	Negative	Negative	78	63	15	66	8
36	Negative	Positive	59	34	11	38	12
37	Negative	Positive	60	46	10	47	4
38	Negative	Negative	62	56	12	42	10
39	Negative	Positive	54	40	11	32	7
40a	Negative	Positive	37	33	5	31	8
40b	Negative	Positive	70	61	15	51	9

LPA213 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
1	0	1	5	0	0	0	0	0	1	9	54	1	0	0	0	2
2a	0	0	3	0	0	0	0	0	0	4	49	3	0	0	1	8
2b	1	3	4	0	0	0	0	0	1	14	38	4	0	0	0	8
3	1	0	3	0	0	0	0	0	6	9	40	5	0	0	3	6
4	1	0	3	2	0	0	0	0	3	9	47	4	0	0	2	2
5	0	0	2	0	0	0	0	0	1	4	35	5	0	0	0	6
6	0	2	3	0	0	0	0	0	3	16	44	7	0	0	1	6
7a	0	3	2	0	0	0	0	0	0	6	34	4	0	0	0	7
7b	0	2	5	0	0	0	0	0	0	4	18	6	0	0	0	6
8a	0	0	4	1	0	0	0	1	0	2	16	6	0	0	0	18
8b	1	2	4	0	0	0	0	0	0	7	11	5	0	0	0	9
9	2	1	4	0	0	0	0	0	2	17	45	1	2	0	3	6
10	0	2	3	1	0	0	0	0	1	22	52	15	1	0	2	8
11	0	0	7	0	0	0	0	0	3	7	9	4	0	0	1	0
12	0	3	2	0	0	0	0	0	2	5	29	5	0	0	0	7
13	0	6	1	1	0	0	0	0	4	20	22	4	0	0	2	1
14	3	1	2	1	0	0	0	0	2	9	38	12	0	0	3	4
15a	0	0	3	1	0	0	0	0	2	12	21	7	0	0	1	2
15b	2	0	3	3	0	0	0	0	1	13	49	4	0	0	0	6
16	0	0	2	1	0	0	0	1	3	10	28	3	0	0	1	7
17	0	0	6	0	0	0	0	0	1	14	32	1	0	0	2	6
18	0	0	4	1	0	0	0	0	0	15	32	6	0	0	6	10
19	0	2	8	1	0	0	0	0	0	8	32	3	0	0	0	11

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	Cld- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-	CldU- IdU+	CldU+ IdU+	CldU+ IdU-	CldU- IdU-
20	0	0	6	0	0	0	0	0	0	7	44	5	0	0	0	5
21a	0	5	1	0	0	0	0	0	3	8	29	3	0	0	0	2
21b	0	1	3	2	0	0	0	0	1	1	29	3	0	0	0	5
22	1	1	2	0	0	0	0	0	2	25	52	2	0	0	9	2
23	0	2	3	0	0	0	0	0	0	12	21	0	0	1	2	6
24a	0	0	1	3	0	0	0	0	2	8	22	5	0	0	8	0
24b	0	1	0	0	0	0	0	0	0	10	46	19	0	1	6	12
25	0	1	3	0	0	0	0	0	0	11	20	4	1	3	12	5
26	0	5	5	1	0	0	0	0	1	12	17	7	0	1	11	6
27	0	2	2	2	0	0	0	0	0	8	22	5	0	1	2	8
28	2	5	3	0	0	0	0	0	3	7	13	6	0	0	15	9
29a	0	0	7	0	0	0	1	0	0	4	20	5	0	0	13	9
29b	0	2	2	1	0	0	0	0	2	13	26	2	0	0	3	4
30	0	3	1	0	0	0	0	0	2	14	17	5	0	2	43	5
31	0	1	2	0	0	0	0	0	0	7	17	2	2	3	34	8
32	0	4	0	0	0	0	0	0	0	15	36	1	0	0	8	4
33	0	0	5	0	0	0	0	0	0	14	20	0	0	2	23	5
34a	0	0	3	0	0	0	0	0	3	14	28	8	0	0	9	8
34b	0	0	1	1	0	0	1	0	5	5	24	9	0	0	5	12
35	0	3	4	0	0	0	1	0	0	12	39	8	0	0	4	7
36	0	2	10	0	0	0	0	0	3	5	16	2	1	0	1	19
37	0	0	4	0	0	0	0	0	2	8	29	4	0	0	5	8
38	0	7	3	0	0	0	0	0	0	4	26	2	0	1	15	4
39	0	1	1	4	0	0	1	0	1	8	13	4	0	1	15	5
40a	0	2	6	0	0	0	0	0	0	3	17	3	0	0	5	1
40b	0	2	7	0	0	0	0	0	0	13	25	4	0	0	14	5

1.3.13. **LPA219 count and convergence data**

LPA219 count data

			Total				
Crypt Number	**C1-20**	**COX-1**	**DAPI**	**CldU**	**IdU**	**Ki67**	**Lgr5**
1a	Negative	Positive	74	53	16	65	2
1b	Negative	Positive	33	25	7	29	2
2a	Negative	Positive	71	53	5	63	1
2b	Negative	Positive	67	56	13	60	1
3a	Negative	Positive	39	29	6	33	1
3b	Negative	Positive	52	44	12	45	3
4	Negative	Positive	105	78	37	100	8
5	Negative	Positive	61	55	14	57	9
6	Negative	Positive	62	48	19	53	9
7	Negative	Negative	65	51	18	57	3
8	Negative	Positive	58	46	9	50	8
9	Negative	Positive	69	66	16	61	4
10	Negative	Positive	51	43	12	43	4
11	Negative	Positive	39	32	15	38	7
12	Negative	Negative	56	50	10	51	5
13	Negative	Negative	63	48	17	53	9
14	Negative	Negative	56	35	10	45	6
15	Negative	Positive	48	38	16	41	13
16	Negative	Positive	65	53	5	58	11
17	Negative	Negative	59	55	10	53	5
18	Negative	Negative	56	42	13	48	1
19	Negative	Positive	23	19	7	21	5
20	Negative	Negative	82	61	13	71	5
21	Negative	Positive	67	57	17	57	3
22	Negative	Positive	70	66	20	62	8

LPA219 convergence data

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
1a	0	0	2	0	0	0	0	0	6	9	41	7	1	0	1	7
1b	0	0	2	0	0	0	0	0	1	6	17	3	0	0	0	4
2a	0	0	1	0	0	0	0	0	1	4	46	11	0	0	2	6
2b	0	1	0	0	0	0	0	0	1	11	42	5	0	0	2	5
3a	0	1	0	0	0	0	0	0	0	5	22	5	0	0	1	5
3b	0	0	2	0	0	0	0	1	2	9	32	0	1	0	1	4
4	0	3	2	3	0	0	0	0	8	22	49	13	2	2	0	1
5	0	3	6	0	0	0	0	0	0	9	36	3	1	1	0	2
6	0	2	7	0	0	0	0	0	3	12	24	5	1	1	2	5
7	0	0	3	0	0	0	0	0	1	13	32	8	3	1	2	2
8	0	2	3	3	0	0	0	0	0	6	32	4	0	1	2	5
9	0	0	3	0	0	0	0	1	0	13	45	0	2	1	4	0
10	0	1	3	0	0	0	0	0	1	8	27	3	2	0	4	2
11	1	5	0	1	0	0	0	0	2	7	20	2	0	0	0	1
12	1	2	1	1	0	0	0	0	0	6	37	3	0	1	3	1
13	0	3	5	1	0	0	0	0	0	12	26	6	2	0	2	6
14	0	0	5	1	0	0	0	0	2	8	22	7	0	0	0	11
15	0	11	1	1	0	0	0	0	1	4	21	2	0	0	1	6
16	0	3	8	0	0	0	0	0	0	2	36	9	0	0	4	3
17	0	0	4	1	0	0	0	0	1	8	37	2	0	1	5	0
18	0	0	1	0	0	0	0	0	3	7	31	6	2	1	2	3
19	0	2	3	0	0	0	0	0	1	3	10	2	1	0	1	0
20	0	0	5	0	0	0	0	0	5	8	45	8	0	0	3	8

	EGFP+								EGFP-							
	Ki67+				Ki67-				Ki67+				Ki67-			
Crypt Number	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**Cld- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**	**CldU- IdU+**	**CldU+ IdU+**	**CldU+ IdU-**	**CldU- IdU-**
21	0	3	0	0	0	0	0	0	1	13	34	6	0	0	7	3
22	0	2	5	1	0	0	0	0	0	18	35	1	0	0	6	2

1.4. MATLAB SCRIPTS

1.4.1. Stem cell population model code

```
%% Mutated mtDNA propogation within a population of stem cells
%
% This script simulates the number of mutated mtDNA molecules that are
% replicated before cell division and how many of those mutated mtDNA
% segregate to one of two daughter cells. This script also includes a birth
% death cycle of mutated mtDNA molecules to simulate a quiescence state in
% which mtDNA molecules are being degraded and mtDNA molecules are being
% replicated to maintain the same number of mtDNA molecules.
%
% All Mutated mtDNA are those that will contribute towards COX Deficiency.

tic
%% Parameter values

rng('shuffle')

% Total number of mtDNA molecules within a cell

mtDNATot = 200;

% MtDNA mutation rate

WTRate = 1e-2;

% Number of cells to be simulated

Sim = 1000;

% Maximum number of cell divisions per cell
% Mouse stem cells divide approx once per day
% 36 months x 30 days

MaxDiv = 1080;

% Initial Number of mutations per stem cell

InitialMut = 0;

% Birth-Death cycles (i.e. the amount of time the cell spends in a quiecent
% state)

BirthDeath = 0;

% Record results

MutResult = zeros(MaxDiv,Sim);
```

```
%% Simulate the experiment

for qq = 1:Sim

   %%%% Random Number Generator SPEED BOOST %%%%

   clearvars RandomNumbers rngcount
   RandomNumbers = rand(10000000,1);
   rngcount = 1;

   mtDNAMut = InitialMut; % Reset mtDNA mutations to their initial values

   for ii = 1 : MaxDiv

      % Quiescence stage - number of birth death cycles of mtDNA
      % molecules (more birth death cycles means longer quiescence state)

      for ee = 1 : BirthDeath % Number of mtDNA birth-death cycles

         %%%% Birth cycle %%%%

         % The probability of a mutated mtDNA molecule replicating is
         % dependent on the number of mutated mtDNA molcules present
         % and the total number of mtDNA molecules within the cell.

         if RandomNumbers(rngcount,1) < mtDNAMut/mtDNATot
            mtDNAMut = mtDNAMut + 1;
         end

         % If it is not a mutated molecule being replicated what is the
         % chance of this molecule replicating and producing a mutated
         % molecule due to errors in replication?

         if RandomNumbers(rngcount,1) > mtDNAMut/mtDNATot
            rngcount = rngcount + 1;
            if RandomNumbers(rngcount,1) < WTRate
               mtDNAMut = mtDNAMut + 1;
            end
            rngcount = rngcount + 1;
         end
         rngcount = rngcount + 1;

         %%%% Death cycle %%%%

         % The probability of a mutated mtDNA molecule being killed is
         % dependent on the number of mutated mtDNA molcules present
         % and the total number of mtDNA molecules within the cell.

         if RandomNumbers(rngcount,1) < mtDNAMut/(mtDNATot+1)
```

```
        % An extra one has been born in the birth phase
        % (mtDNATot + 1)
        mtDNAMut = mtDNAMut - 1;
    end

    rngcount = rngcount + 1;

    % If the birth death cycles produce mtDNA mutation values below
    % zero or above the maximum number of mtDNA molecules then limit
    % them.

    if mtDNAMut < 0
        mtDNAMut = 0;
    end

    if mtDNAMut > mtDNATot
        mtDNAMut = mtDNATot;
    end

end

%%%% mtDNA replication stage %%%%

NewCell = 0; % For every division the NewCell value needs to be
% reset to 0

% Relaxed replication takes place to bring the number of mtDNA
% molecule to mtDNATot*2

for nn = 1 : mtDNATot

    % For each mtDNA replication the probability that a mutated
    % mtDNA molecule is replicated is dependent on the number of
    % mutated mtDNA molecules and the number of mtDNA molecules
    % that are present in the cell. The number of mtDNA molecules
    % present increases every time a mtDNA molecule is replicated
    % therefore the probability denominator increases by one each
    % time. When a normal mtDNA molecule is replicated there is a
    % chance a new mutation is introduced into the daughter mtDNA
    % molecule

    RepProb = mtDNAMut / (mtDNATot + (nn - 1));

    if RandomNumbers(rngcount,1) < RepProb
        mtDNAMut = mtDNAMut + 1;
    end

    % The same random number has to be used to determine whether it
    % is a normal or mutated mtDNA molecule that is being
    % replicated.
```

```
            if RandomNumbers(rngcount,1) > RepProb
                rngcount = rngcount + 1;
                if RandomNumbers(rngcount,1) < WTRate
                    mtDNAMut = mtDNAMut + 1;
                end
                rngcount = rngcount + 1;
            end
            rngcount = rngcount + 1;
        end

        %%%% mtDNA segregation stage %%%%

        % Random segregation of mtDNA molecules into daughter cells.

        for tt = 1 : mtDNATot

            % For each mtDNA segregation the probability that a mutated
            % mtDNA molecule is segregated is dependent on the number of
            % mutated mtDNA molecules and the number of mtDNA molecules
            % that are left in the mother cell. The number of mtDNA
            % molecules left decreases every time a mtDNA molecule is
            % segregated therefore the probability denominator decreases
            % by one each time. The numerator is dependent on the number of
            % mutated mtDNA that were present in the mother cell minus the
            % number of those that have been segregated into the daughter
            % cell.

            DivProb = (mtDNAMut - NewCell) / ((2*mtDNATot) - (tt-1));

            if RandomNumbers(rngcount,1) < DivProb
                NewCell = NewCell + 1;
                MutResult(ii,qq) = NewCell;
            end
            rngcount = rngcount + 1;
        end

        % After segregation the number of mutated mtDNA molecules gets
        % updated to the number that are now in the new cell before the
        % cycle runs again.

        mtDNAMut = NewCell;

    end

end

clearvars RandomNumbers

%% Cell Accumulation Analysis
```

```
% Find which cells have a mtDNA fixation event

MaxMut = zeros(1,Sim);

FixResult = 0;

for ii = 1 : Sim
    a = max(MutResult(:,ii));
    MaxMut(ii) = a;
end

CellSim = find(MaxMut == mtDNATot);

for tt = 1 : numel(CellSim);
    FixPos = find(MutResult(:,CellSim(1,tt)) == mtDNATot);
    FixAge = FixPos(1,1);
    FixResult(end+1) = FixAge;
end

FixResult(1) = [];

% Display the average fixation time in the command window

AverageFixTime = mean(FixResult)

%% How many cells become fixed at a particular age?
% 1 month intervals

for ff = 1 : 36

   COXPos = find(MutResult(((ff-1)*30+1),:) > mtDNATot*0.75);
   COXdefSC = numel(COXPos);
   FractionMutated = COXdefSC/Sim;
   COXDefAge(ff,:) = FractionMutated;

end

%% Graphing the results

plot(MutResult);

toc
```

1.4.2. **Niche succession model code**

1.4.2.1. *Part 1*

```
%% Niche Succession Model - FINAL
% Stem cell dynamics and mutated mtDNA clonal expansion
%
% The script is an amalgamation of the previous crypt model. It identifies
% that there are a certain number of mtDNA molecules residing within each
% stem cell of the crypt. With the evolution of stem cell divisions, the
% number of mutated mtDNA molecules evolves stochastically according to
% pre-determined probabilities. Also, with each additional mutated mtDNA
% molecule, the model determines which kind of mutation has developed
% according to probability data previously acquired. Therefore, this model
% is a more accurate representation of the processes that take place within
% the crypt and at the tissue level.

% v8 v7.3 compression of the saved variables. Time bar for each crypt
% simulation
% v9 Integrates a user interface box which asks all of the required
% parameters
% v10 enables the user to open and close a parameter list file while the
% simulation is running to add new simulations to the list
% v11 Crypt fission fix which allows the continuation of the script for
% large numbers of runs
% v12 The way in which the new crypt data from fission is integrated into
% the final results table is drawn using the randperm function therefore
% unique numbers are now selected.
% v13 Every new crypt fission event does not overwrite the resultant data
% from a previous crypt result with the addition of a cryptReplaceCount
% Counter for every crypt that is replace in the final data set.
% v14 Solving memory issues which arise after prolonged model simulation,
% solved by executing clear command before every model of a different
% parameter set.
% v26 Fully COX deficient stem cells are subject to random removal for some
% relevant biological reason -- at the mtDNA molecular level. SpeciesID
% identification and removal.
% v27 Additional COX deficient stem cell division and ParameterNames
% variable has been transposed.
% v28 Mitochondrial degradation incorporated into the model
% v29 Integration of the new transition matrices that take into account
% the possible asymmetric segregation of mutated mtDNA molecules.

% Load the parameter list file
load ParameterListFittingScan

% Determine how many simulations are to be carried out using 'cycle'

ParameterNames = ParameterNames';
a = size(ParameterNames);
```

```
b = a(1);
cycle = b - 1;

% Set 'qq' to 1 for the first simulation
qq = 1;

% For every simulation increase 'qq' by 1 until total number of simulations
% 'cycle' has been reached

while qq <= cycle

   % Clear memory after every simulation so that the memory doesn’t become too
   % fragmented when many simulations are to be carried out
   save SystemMemoryClearUp qq
   save SystemMemoryClearUp cycle -append
   clear

   % Set global variable structure 'gg' where all parameters are stored for the
   % model simulation and where all metrics are stored once model is completed
   global gg
   global dd

   % Reload all critical variables after the memory purge
   load SystemMemoryClearUp
   load ParameterListFittingScan

   % Transpose parameter variables so that they're in the correct format
   ParameterNames = ParameterNames';

   % What is the filename for the overall results?
   gg.finalFilename = datestr(clock,30);
   ParameterNames(qq+1, 14) = {gg.finalFilename};

   % Shuffle random number generator before every simulation so that the model
   % is truly stochastic in nature
   rng('shuffle');

   %% Load all the variables into the 'gg' global variable

   % Number of crypts generated per simulation
   gg.numRuns = cell2mat(ParameterNames(qq+1,1));

   % The percentage threshold that characterises a stem cell as COX deficient
   gg.mutThreshold = cell2mat(ParameterNames(qq+1,2));

   % The number of asynchronous stem cell divisions that portrays the human
   % lifespan (1 stem cell division per week)
   gg.numDiv = cell2mat(ParameterNames(qq+1,3));

   % Number of mtDNA molecules contained within each stem cell
```

```
gg.mtDNA = cell2mat(ParameterNames(qq+1,4));

% Number of stem cells contained within crypts
gg.initS = cell2mat(ParameterNames(qq+1,5));

% Stem cell division types 'Pa' Asymmetric probability 'Ps' Symmetric
% probability
gg.Pa = cell2mat(ParameterNames(qq+1,6));
gg.Ps = (1 - gg.Pa)/2;

% Advantage to COX deficient stem cell to divide more often according to
% 'adv' which increases Ps and reduces Pa1
gg.adv = cell2mat(ParameterNames(qq+1,7));

% Which method will be used for the mutation rate?
gg.mutMethod = char(ParameterNames(qq+1,8));

% What is the base mutation rate?
gg.mutationRate = cell2mat(ParameterNames(qq+1,9));

% Is there a mutation rate fold change from 0 to 80 years of age? if so
% what is it? If there isn't this should be set to 1.
gg.mutationRateFold = cell2mat(ParameterNames(qq+1,15));

% Calculating the mutation rate vector for each division step in the    model
c = gg.mutationRate;
m = (gg.mutationRateFold *gg.mutationRate - gg.mutationRate) / 4171;
for zz = 1 : 5211
   gg.mutationRate1(zz) = m*zz + c;
end

%% COX Correction Factors

gg.COXCorrectionFactor = cell2mat(ParameterNames(qq+1,16));

gg.COXCorrectionFactor2 = cell2mat(ParameterNames(qq+1,17));

gg.COXSCTimePoint = char(ParameterNames(qq+1,18));

gg.COXSCTimePointInterval = cell2mat(ParameterNames(qq+1,19));

gg.COXDefCycleRepeats = cell2mat(ParameterNames(qq+1,20));

gg.MitoDegradation = cell2mat(ParameterNames(qq+1,21));

%% Crypt Fission

gg.cryptFission = char(ParameterNames(qq+1,11));

if strcmp('yes',gg.cryptFission)
```

```
        gg.cryptNormalPercentage  = cell2mat(ParameterNames(qq+1,12));
        gg.cryptFissionFactor  = cell2mat(ParameterNames(qq+1,13));
        gg.cryptFissionProb  = (1/gg.numDiv)*gg.cryptNormalPercentage;
        gg.cryptFisSave  = 1;

    end

    % Parameters  to prime  the  metrics  to be recorded

    gg.FailedCE  = [0;0;0];
    gg.SuccessCE  = [0;0;0];

    gg.NicheFailedSC  = [0;0;0];
    gg.NicheSuccessSC  = [0;0;0];

    % Load probability  tables  and mutation  probabilities

    switch  gg.mtDNA

        case 5

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb5.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb5.mat');

        case 10

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb10.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb10.mat');

        case 25

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb25.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb25.mat');

        case 50

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb50.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb50.mat');

        case 100

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb100.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb100.mat');

            % Load the  advantage  DivProbs  for 100 mtDNA  SCs

            % load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb10010.mat');
            % load('D:\Niche  Succession  Model
Transfer\dividingMutations\DivProb100100.mat');
```

```
            % load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb1002.mat');
            % load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb10011.mat');
            % load('D:\Niche  Succession  Model
Transfer\dividingMutations\DivProb100101.mat');
            % load('D:\Niche  Succession  Model
Transfer\dividingMutations\DivProb100102.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb100103.mat');
            % load('D:\Niche  Succession  Model
Transfer\dividingMutations\DivProb100108.mat');
            % load('D:\Niche  Succession  Model
Transfer\dividingMutations\DivProb1001001.mat');

        case 200

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb200.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb200.mat');

        case 400

            load('D:\Niche  Succession  Model Transfer\replicatingMutations\RepProb400.mat');
            load('D:\Niche  Succession  Model Transfer\dividingMutations\DivProb400.mat');

        otherwise
            warning('Please  enter a valid  mtDNA number  for which  a transition  matrix  has been
created.');

    end

    gg.RepProb  = RepProb; clearvars  RepProb
    gg.DivProb  = DivProb;  clearvars  DivProb

    % gg.DivProb10  = DivProb10010;  clearvars  DivProb10010
    % gg.DivProb100  = DivProb100100;  clearvars  DivProb100100
    % gg.DivProb2  = DivProb1002;  clearvars  DivProb1002
    % gg.DivProb11  = DivProb10011;  clearvars  DivProb10011

    % gg.DivProb101  = DivProb100101;  clearvars  DivProb100101
    % gg.DivProb102  = DivProb100102;  clearvars  DivProb100102
    gg.DivProb103  = DivProb100103;  clearvars  DivProb100103
    % gg.DivProb108  = DivProb100108;  clearvars  DivProb100108
    % gg.DivProb1001  = DivProb100108;  clearvars  DivProb100108

    % Least  sqaures  determination

    gg.LeastSqauresRunInterval  = gg.numRuns  / 100;

    % Save parameters  name  with  new information

    ParameterNames  = ParameterNames';
```

```
save ParameterListFittingScan ParameterNames

clearvars ParameterNames

tic

if strcmp('yes',gg.cryptFission)

    % Which simulation type is going to be performed

    switch gg.mutMethod

        case 'constant'

            [MutatedSCAgeFinal, MutatedSCAgeCorrFinal,...
                MutatedSCAgeCorr2Final] = mtDNACrypt_ConstantV11FCN_COXAd_CF();

        case 'exponential'

            MutatedSCAgeFinal = mtDNACrypt_ExponentialV2FCN_CF();
    end

    save MutatedSCAgeFission MutatedSCAgeFinal -v7.3
    save MutatedSCAgeCorrected MutatedSCAgeCorrFinal -v7.3
    save MutatedSCAgeCorrected2 MutatedSCAgeCorr2Final -v7.3

    % Load the crypt data where crypt fission has occurred

    rr = load('cryptFissionResult.mat');
    rr = rmfield(rr,'Kickstart1');
    cryptNames = fieldnames(rr);

    % The loaded crypt data is loaded into a cell in order to find out the
    % number of crypts that underwent fission

    s = numel(cryptNames);

    % Record how many doublets have formed

    gg.colonys.doublets = s;

    % Bring out all the crypts data before crypt fission occurred from the
    % structured array

    struct2var(rr);

    % Create a new matrix that contains the continued data from the crypt
    % fission event

    MutatedSCAgeFission = zeros(gg.numDiv,s);
```

```
% Create a new vector which records the point at which each crypt fission
% event took place so this can be used only import new crypt fission data
% into the original results matrix

CryptFisTime = zeros(1,s);

% Reset the filename so that the new crypt fission data is saved to a
% different location

gg.filename = 'cryptFissionResult2';

h = waitbar(0,'Simulating model with crypt fission - part 2, please wait...');

for kk = 1 : s

    s1 = cryptNames(kk,1);    % Identify a single crypt fission event
    s2 = char(s1);   % Convert the crypt name into a string
    s3 = eval(s2);   % Assign the matrix to the string


    if strcmp('constant',gg.mutMethod)
        [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
            mtDNACrypt_ConstantV2FCN_CF_Input(s3);
    else
        [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
            mtDNACrypt_ExponentialV2FCN_CF_Input(s3);
    end

    waitbar(kk/s,h)

end

delete(h)

% Save the crypts that have undergone a second round of fission

save(gg.filename,'dd','-v7.3');

gg.FissionTime = CryptFisTime;

load MutatedSCAgeFission MutatedSCAgeFinal

% Replace random crypts

cryptReplaceCount = 0;

a = numel(CryptFisTime);
b = size(MutatedSCAgeFinal);
cryptReplaceNo = [randperm(b(2)) randperm(b(2)) randperm(b(2))];
```

```
    %    c = ceil(rand(1,a)*b(2));

    for tt = 1 : a
        MutatedSCAgeFinal(CryptFisTime(tt):gg.numDiv,cryptReplaceNo(tt +
cryptReplaceCount)) =...
            MutatedSCAgeFission(CryptFisTime(tt):gg.numDiv,tt);
    end

    cryptReplaceCount = cryptReplaceCount + a;

    save MutatedSCAgeFission MutatedSCAgeFinal -v7.3

    clearvars -except gg cycle qq cryptReplaceCount cryptReplaceNo

    gg.analysisComplete = 0;

    %% Analysis OR Crypt Fission Round 2

    load('cryptFissionResult2.mat');
    rr = dd;
    clearvars dd

    if isstruct(rr)
        cryptNames = fieldnames(rr);
    else
        cryptNames = [];
    end

    if isempty(cryptNames)
        moreCryptFission = 'no';
    else
        moreCryptFission = 'yes';
    end

    if strcmp('no',moreCryptFission)

        gg.analysisComplete = 1;

    else

        % The loaded crypt data is loaded into a cell in order to find out the
        % number of crypts that underwent fission

        s = numel(cryptNames);

        % Record how many doublets have formed

        gg.colonys.triplets = s;
```

```
% Bring out all the crypts data before crypt fission occurred from the
% structured array

struct2var(rr);

% Create a new matrix that contains the continued data from the crypt
% fission event

MutatedSCAgeFission = zeros(gg.numDiv,s);

% Create a new vector which records the point at which each crypt fission
% event took place so this can be used only import new crypt fission data
% into the original results matrix

CryptFisTime = zeros(1,s);

% Reset the filename so that the new crypt fission data is saved to a
% different location

gg.filename = 'cryptFissionResult3';

clearvars -global dd
global dd

h = waitbar(0,'Simulating model with crypt fission - part 3, please wait...');

for kk = 1 : s

    s1 = cryptNames(kk,1);    % Identify a single crypt fission event
    s2 = char(s1);  % Convert the crypt name into a string
    s3 = eval(s2);  % Assign the matrix to the string


    if strcmp('constant',gg.mutMethod)
        [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
            mtDNACrypt_ConstantV2FCN_CF_Input(s3);
    else
        [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
            mtDNACrypt_ExponentialV2FCN_CF_Input(s3);
    end

    waitbar(kk/s,h)

end

delete(h)

save(gg.filename,'dd','-v7.3')

gg.FissionTime = [gg.FissionTime CryptFisTime];
```

```
            load MutatedSCAgeFission MutatedSCAgeFinal

            % Replace random crypts

            a = numel(CryptFisTime);

            for tt = 1 : a
                MutatedSCAgeFinal(CryptFisTime(tt):gg.numDiv,cryptReplaceNo(tt +
cryptReplaceCount)) =...
                    MutatedSCAgeFission(CryptFisTime(tt):gg.numDiv,tt);
            end

            cryptReplaceCount = cryptReplaceCount + a;

            save MutatedSCAgeFission MutatedSCAgeFinal -v7.3

            clearvars -except gg cycle qq cryptReplaceCount cryptReplaceNo

        end

        %% Analysis OR Crypt Fission Round 3

        if gg.analysisComplete ~= 1;

            load('cryptFissionResult3.mat');
            rr = dd;
            clearvars dd

            if isstruct(rr)
                cryptNames = fieldnames(rr);
            else
                cryptNames = [];
            end

            if isempty(cryptNames)
                moreCryptFission = 'no';
            else
                moreCryptFission = 'yes';
            end

            if strcmp('no',moreCryptFission)

                gg.analysisComplete = 1;

            else

                % The loaded crypt data is loaded into a cell in order to find out
                % the number of crypts that underwent fission
```

```
s = numel(cryptNames);

% Bring out all the crypts data before crypt fission occurred from
% the structured array

struct2var(rr);

% Record how many doublets have formed

gg.colonys.quadruplets = s;

% Create a new matrix that contains the continued data from the
% crypt fission event

MutatedSCAgeFission = zeros(gg.numDiv,s);

% Create a new vector which records the point at which each crypt
% fission event took place so this can be used only import new
% crypt fission data into the original results matrix

CryptFisTime = zeros(1,s);

% Reset the filename so that the new crypt fission data is savd to
% a different location

gg.filename = 'cryptFissionResult4';

clearvars -global dd
global dd

h = waitbar(0,'Simulating model with crypt fission - part 4, please wait...');

for kk = 1 : s

   s1 = cryptNames(kk,1); % Identify a single crypt fission event
   s2 = char(s1);  % Convert the crypt name into a string
   s3 = eval(s2);  % Assign the matrix to the string

   if strcmp('constant',gg.mutMethod)
      [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
         mtDNACrypt_ConstantV2FCN_CF_Input(s3);
   else
      [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
         mtDNACrypt_ExponentialV2FCN_CF_Input(s3);
   end

   waitbar(kk/s,h)

end
```

```
            delete(h)

            save(gg.filename,'dd','-v7.3');

            gg.FissionTime  = [gg.FissionTime  CryptFisTime];

            load MutatedSCAgeFission  MutatedSCAgeFinal

            % Replace  random  crypts

            a = numel(CryptFisTime);

            for tt = 1 : a
               MutatedSCAgeFinal(CryptFisTime(tt):5210,cryptReplaceNo(tt  +
cryptReplaceCount))  =...
                  MutatedSCAgeFission(CryptFisTime(tt):5210,tt);
            end

            cryptReplaceCount  = cryptReplaceCount  + a;

            save MutatedSCAgeFission  MutatedSCAgeFinal  -v7.3

            clearvars  -except gg cycle  qq cryptReplaceCount  cryptReplaceNo

         end

      end

      %% Analysis  OR Crypt Fission  Round  4

      if gg.analysisComplete  ~= 1;

         load('cryptFissionResult4.mat');
         rr = dd;
         clearvars  dd

         if isstruct(rr)
            cryptNames  = fieldnames(rr);
         else
            cryptNames  = [];
         end

         if isempty(cryptNames)
            moreCryptFission  = 'no';
         else
            moreCryptFission  = 'yes';
         end

         if strcmp('no',moreCryptFission)
```

```
        gg.analysisComplete  = 1;

    else

        % The loaded crypt data is loaded into a cell in order to find out
        % the number of crypts that underwent fission

        s = numel(cryptNames);

        % Record how many doublets have formed

        gg.colonys.quintuplets  = s;

        % Bring out all the crypts data before crypt fission occurred from
        % the structured array

        struct2var(rr);

        % Create a new matrix that contains the continued data from the
        % crypt fission event

        MutatedSCAgeFission  = zeros(gg.numDiv,s);

        % Create a new vector which records the point at which each crypt
        % fission event took place so this can be used only import new
        % crypt fission data into the original results matrix

        CryptFisTime  = zeros(1,s);

        % Reset the filename so that the new crypt fission data is savd to
        % a different location

        gg.filename  = 'cryptFissionResult5';

        clearvars -global dd
        global dd

        h = waitbar(0,'Simulating model with crypt fission - part 5, please wait...');

        for kk = 1 : s

            s1 = cryptNames(kk,1);   % Identify a single crypt fission event
            s2 = char(s1);  % Convert the crypt name into a string
            s3 = eval(s2);  % Assign the matrix to the string

            if strcmp('constant',gg.mutMethod)
                [MutatedSCAgeFission(:,kk),CryptFisTime(kk)]  =...
                    mtDNACrypt_ConstantV2FCN_CF_Input(s3);
            else
                [MutatedSCAgeFission(:,kk),CryptFisTime(kk)]  =...
                mtDNACrypt_ExponentialV2FCN_CF_Input(s3);
            end

            waitbar(kk/s,h)

        end

        delete(h)

        save(gg.filename,'dd','-v7.3');

        gg.FissionTime  = [gg.FissionTime  CryptFisTime];

        load MutatedSCAgeFission  MutatedSCAgeFinal

        % Replace  random  crypts

        a = numel(CryptFisTime);
        %            b = size(MutatedSCAgeFinal);
        %            c = randperm(b(2),a);

        %            c = ceil(rand(1,a)*b(2));

        for tt = 1 : a
            MutatedSCAgeFinal(CryptFisTime(tt):5210,cryptReplaceNo(tt  +
cryptReplaceCount))  =...
                MutatedSCAgeFission(CryptFisTime(tt):5210,tt);
        end

        cryptReplaceCount  = cryptReplaceCount  + a;

        save MutatedSCAgeFission  MutatedSCAgeFinal  -v7.3

        clearvars  -except cycle  qq gg  cryptReplaceCount  cryptReplaceNo

    end

end

%% Analysis  OR Crypt Fission  Round  5 (Last  Round)

if gg.analysisComplete  ~= 1;

    load('cryptFissionResult5.mat');
    rr = dd;
    clearvars  dd

    if isstruct(rr)
        cryptNames  = fieldnames(rr);
    else
        cryptNames = [];
    end

    if isempty(cryptNames)
        moreCryptFission = 'no';
    else
        moreCryptFission = 'yes';
    end

    if strcmp('no',moreCryptFission)

        gg.analysisComplete = 1;

    else

        % The loaded crypt data is loaded into a cell in order to find out
        % the number of crypts that underwent fission

        s = numel(cryptNames);

        % Record how many doublets have formed

        gg.colonys.sextuplets = s;

        % Bring out all the crypts data before crypt fission occurred from
        % the structured array

        struct2var(rr);

        % Create a new matrix that contains the continued data from the
        % crypt fission event

        MutatedSCAgeFission = zeros(gg.numDiv,s);

        % Create a new vector which records the point at which each crypt
        % fission event took place so this can be used only import new
        % crypt fission data into the original results matrix

        CryptFisTime = zeros(1,s);

        % Reset the filename so that the new crypt fission data is saved to
        % a different location

        gg.filename = 'cryptFissionResult6';

        clearvars -global dd
        global dd

        h = waitbar(0,'Simulating model with crypt fission - part 6, please wait...');
```

```
            for kk = 1 : s

                s1 = cryptNames(kk,1);   % Identify a single crypt fission event
                s2 = char(s1);  % Convert the crypt name into a string
                s3 = eval(s2);   % Assign the matrix to the string

                if strcmp('constant',gg.mutMethod)
                    [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
                        mtDNACrypt_ConstantV2FCN_CF_Input(s3);
                else
                    [MutatedSCAgeFission(:,kk),CryptFisTime(kk)] =...
                        mtDNACrypt_ExponentialV2FCN_CF_Input(s3);
                end

                waitbar(kk/s,h)

            end

            delete(h)

            save(gg.filename,'dd','-v7.3');

            gg.FissionTime = [gg.FissionTime CryptFisTime];

            load MutatedSCAgeFission MutatedSCAgeFinal

            % Replace random crypts

            a = numel(CryptFisTime);

            for tt = 1 : a
                MutatedSCAgeFinal(CryptFisTime(tt):5210,cryptReplaceNo(tt +
cryptReplaceCount)) =...
                    MutatedSCAgeFission(CryptFisTime(tt):5210,tt);
            end

            cryptReplaceCount = cryptReplaceCount + a;

            save MutatedSCAgeFission MutatedSCAgeFinal -v7.3

            clearvars -except cycle qq gg cryptReplaceCount cryptReplaceNo

        end

    end

    %% What happens if there is no crypt fission that occurs??

  else
```

```
        switch gg.mutMethod

            case 'constant'

                [MutatedSCAgeFinal, MutatedSCAgeCorrFinal, MutatedSCAgeCorr2Final] =
mtDNACrypt_ConstantV11FCN_COXAd();

            case 'exponential'

                MutatedSCAgeFinal = mtDNACrypt_ExponentialV2FCN();
        end

        save MutatedSCAgeFission MutatedSCAgeFinal -v7.3
        save MutatedSCAgeCorrected MutatedSCAgeCorrFinal -v7.3
        save MutatedSCAgeCorrected2 MutatedSCAgeCorr2Final -v7.3


        clearvars -except gg cycle qq

    end

    %% Least squares to determine the optimum number of runs

    load MutatedSCAgeFission MutatedSCAgeFinal
    load MutatedSCAgeCorrected MutatedSCAgeCorrFinal
    load MutatedSCAgeCorrected2 MutatedSCAgeCorr2Final

    for jj = 1 : gg.numRuns / gg.LeastSqauresRunInterval
        for ii = 1 : gg.numDiv
            a(ii,jj) = mean(MutatedSCAgeFinal(ii,1:(jj*gg.LeastSqauresRunInterval)));
        end
    end

    b = size(a);

    for tt = 1 : b(2)
        c(tt) = sum(a(:,tt));
    end

    for rr = 1 : length(c)-1
        d(rr) = sqrt((c(rr+1) - c(rr))^2);
    end


    %% Save Simulation Results

    % remove fields that are no longer required

    if strcmp('no',gg.cryptFission)
        fields = {'RepProb','DivProb'};
```

```
else
    fields  = {'filename','analysisComplete','RepProb','DivProb','cryptFisSave'};
end

gg = rmfield(gg,fields);

% Make the final results and the least square values global gg

gg.MutatedSCAgeFinal  = MutatedSCAgeFinal;
gg.MutatedSCAgeFinalCorr  = MutatedSCAgeCorrFinal;
gg.MutatedSCAgeFinalCorr2  = MutatedSCAgeCorr2Final;

gg.LeastSqaures  = d;

% save the whole global gg variable

gg.SimulationTime  = toc;

save(gg.finalFilename,'gg','-v7.3')

% Delete files that are no longer required

if strcmp('no',gg.cryptFission)
    delete('MutatedSCAgeFission.mat')
    delete('MutatedSCAgeCorrected.mat')
    delete('MutatedSCAgeCorrected2.mat')
    delete('SystemMemoryClearUp.mat')
else
    delete('MutatedSCAgeFission.mat')
    delete('cryptFissionResult*')
    delete('MutatedSCAgeCorrected.mat')
    delete('MutatedSCAgeCorrected2.mat')
    delete('SystemMemoryClearUp.mat')
end

% Clear up the desktop and workspace and declare the model is finished

qqstr = num2str(qq);
disp(['Model ' qqstr ' Completed Successfully']);

% Assess the number of parameters that

load ParameterListFittingScan
ParameterNames  = ParameterNames';
a = size(ParameterNames);
cycle  = a(1) - 1;
clearvars  -except qq cycle

qq = qq + 1;
```

```
end

disp('Simulations  Complete');
h = msgbox('Simulation  Complete','Success');
clear  all;
```

1.4.2.2. *Part 2*

```
function [MutatedSCAge, MutatedSCAgeCorr, MutatedSCAgeCorr2] =
mtDNACrypt_ConstantV11FCN_COXAd_CF()

% mtDNACrypt Function for Constant and Increasing Mutation Rate - FINAL

% The script is an amalgamation of the previous crypt model. It identifies
% that there are a certain number of mtDNA molecules residing within each
% stem cell of the crypt. With the evolution of stem cell divisions, the
% number of mutated mtDNA molecules evolves stochastically according to
% pre-determined probabilities. Also, with each additional mutated mtDNA
% molecule, the model determines which kind of mutation has developed
% according to probability data previously acquired. Therefore, this model
% is a more accurate representation of the processes that take place within
% the crypt and at the tissue level.

% Tracks multiple mutations on single mtDNA species for clonal expansion
% comparison of multiple mutations within individual cells with biological
% data

% Bring in the global variable 'gg' that has already been set up and make a
% global 'dd' variable
global gg
global dd

% Set up the results matrices
MutatedSCAge = zeros(gg.numDiv,gg.numRuns);
MutatedSCAgeCorr = zeros(gg.numDiv,gg.numRuns);
MutatedSCAgeCorr2 = zeros(gg.numDiv,gg.numRuns);

% Set up a progress tracking bar
h = waitbar(0,'Simulating model w/o crypt fission, please wait...');

% Set up the multiple mutations record matrices for stem cells at age 70
% years of simulated time
SingleMutRecord = zeros(gg.numRuns,gg.initS);
MultipleMutRecord = zeros(gg.numRuns,gg.initS);

% Probability for each age (numDiv) getting a mutation
mutProbAge = zeros(2,gg.numDiv);

% Crypt Fission Recording
Kickstart1 = [];
gg.filename = 'cryptFissionResult';
save(gg.filename,'Kickstart1');

for pp = 1 : gg.numRuns

    % set up the multiple mutations species ID result structure
```

```
mtDNAmutations = zeros(gg.numDiv, gg.initS*gg.mtDNA);

% we start with all cells/mtDNA mutation free
MutatedAll = zeros(gg.numDiv,gg.initS);

% Set up the first value of the original mutation
origMut = 1;

% Set up the mtDNA species records
speciesIDRecord = [];
speciesIDMultRecord = [];

% initiate time, time+1 means divTime has passed
time = 1;

%% Pre-determined random numbers for crypt simulation

% Mutation Rate random numbers
aaaa = DiscSampVec3((0:1),[gg.mutationRate1],(gg.mtDNA*gg.initS));

% Stem cell division type random numbers
bbbb = DiscSampVec2((1:3),[gg.Pa,gg.Ps,gg.Ps],gg.numDiv*gg.initS*2);
count2 = 1;

% Stem cell division type with advantage random numbers
cccc = DiscSampVec2((1:3),[gg.Pa-((gg.Ps*gg.adv)-
gg.Ps),gg.Ps*gg.adv,gg.Ps],gg.numDiv*gg.initS*2);
count3 = 1;

% Segregation event random numbers
dddd = DiscSampVec2((1:2),[0.5,0.5],gg.numDiv*gg.initS*2);
count4 = 1;

% Stem cell replacement random numbers
eeee = rand(1,(gg.numDiv*gg.initS*2));
count5 = 1;

% Species ID checking
ffff = ceil(rand(1,(gg.mtDNA*gg.numDiv))*gg.mtDNA);
count6 = 1;

% Crypt Fission Crypt Numbering
gggg = rand(1,2*gg.numDiv);
count7 = 1;

%% Simulate only for certain time
while time < gg.numDiv

    if time == 1
        b = 0;
    else
        a = sum(MutatedAll(time,:));
        b = a > 0;
    end

    %% mutations occuring
    % random numbers generated for each mtDNA molecule
    % within all stem cells of the crypt to determine how many are
    % mutated

    if b == 0

        Mutated = [];

        for iii = 1:gg.initS

            Mutated(iii) = sum(aaaa(time,((iii*gg.mtDNA)-(gg.mtDNA-1)):...
                ((iii*gg.mtDNA)-(gg.mtDNA-1)) + (gg.mtDNA-1)));

        end

        % For each number of new mutations, determine if it is
        % replacing any of the current mutated mtDNA molecules. If it
        % is replacing any, "Mutated" is decreased by the same amount
        % for that stem cell.

        % Proceed if there are mutations present

        MutatedAll(time+1,:) = MutatedAll(time,:) + Mutated;

        % This is the point at which the first mutation will emerge
        % First mutation needs to be inserted and recorded
        % This is just mutation insertion only where they appear

        if max(Mutated) > 0

            for iii = 1 : gg.initS

                % Records how many mtDNA acquire second mutation per
                % stem cell

                Multiple = 0;

                if Mutated(iii) > 0

                    % Isolate the current stem cells mutational species

                    tempA = mtDNAmutations(time, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA)));
```

```
% Find all the WT mtDNA molecules

tempC = find(tempA == 0);

% For each new mutation, determine whether it is
% affecting a WT mtDNA or an already mutated mtDNA
% molecule

ttt = 1;
mutPos = zeros(1,Mutated(iii));

while ttt <= Mutated(iii)

   tempZ = ffff(count6);

   if isempty(find(mutPos == tempZ))

      mutPos(ttt) = tempZ;
      count6 = count6 + 1;
      ttt = ttt + 1;

   else

      count6 = count6 + 1;

   end

end

% Which values of mutPos are not present in tempC

for ttt = 1 : numel(mutPos)
   occuMut = find(tempC == mutPos(ttt));
   if isempty(occuMut)
      Multiple = Multiple + 1;
   end
end

if Multiple > 0

   % Find all current mutations

   currMut = find(tempA > 0);

   % Produce a random permutation of the indexed
   % mutated mtDNA molecules

   currMutRandom = currMut(randperm(numel(currMut)));

   % The overwritten molecules species ID's will be
```

```
overSpeciesID = tempA(currMutRandom(1:Multiple));

% Determine the new species IDs for the mutated mtDNA molecules

tempE = origMut : (origMut + (Multiple-1));

% Take away the multiple mutations from the species
% ID generator vector

tempEMult = tempE(1 : Multiple); % Contains the multiple species ID

tempEWT = tempE((Multiple+1) : end); % Contains the normal species ID

% Insert the new species ID into the current list
% of species IDs in the WT molecule positions

tempA(tempC(1:numel(tempEWT))) = tempEWT;

% Replace the selected mtDNA species to be
% overwritten for multiple mutations on same
% mtDNA species.

for ttt =1 : Multiple
    a = find(tempA == overSpeciesID(ttt));
    tempA(a(end)) = tempEMult(ttt);
end

% Insert the modified species ID vector into the
% master matrix

mtDNAmutations(time+1, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA))) = tempA;

% This needs to be reflected in the MutatedAll
% array as well

MutatedAll(time+1,iii) = MutatedAll(time+1,iii) - Multiple;

% increase the species ID tracker by one

origMut = origMut + numel(tempE);

% Recording the multiple mutation information
% For each mutation, need to check whether it
% has been muliplied before.

for ttt = 1 : Multiple

    % determine whether the speciesID that is about
```

```
                    % to be overwritten has any multiple mutations
                    % already

                    a = find(speciesIDRecord == overSpeciesID(ttt));

                    if isempty(a)
                        speciesIDRecord(end+1) = tempEMult(ttt);
                        speciesIDMultRecord(end+1) = 2;
                    else
                        speciesIDRecord(end+1) = tempEMult(ttt);
                        speciesIDMultRecord(end+1) = speciesIDMultRecord(a) + 1;
                    end
                end

            else

                % Determine the new species IDs for the mutated mtDNA molecules

                tempE = origMut : (origMut + Mutated(iii)-1);

                % Insert the new species ID into the current list
                % of species IDs in the WT molecule positions

                tempA(tempC(1:numel(tempE))) = tempE;

                % Insert the modified species ID vector into the
                % master matrix

                mtDNAmutations(time+1, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA))) = tempA;

                % increase the species ID tracker by one

                origMut = origMut + numel(tempE);

            end

        end

    end

end

time = time + 1;

end

if b > 0

    %% mutations occuring
```

```
        % random numbers generated for each mtDNA molecule
        % within all stem cells of the crypt to determine how many are
        % mutated

        Mutated = [];

        for iii = 1:gg.initS

            Mutated(iii) = sum(aaaa(time,((iii*gg.mtDNA)-(gg.mtDNA-1)):...
                ((iii*gg.mtDNA)-(gg.mtDNA-1)) + (gg.mtDNA-1)));

        end

        MutatedAll(time,:) = MutatedAll(time,:) + Mutated;

        % This is the point at which additional mutations will arise
        % and where multiple mutations will be tracked and recorded

        if max(Mutated) > 0

            for iii = 1 : gg.initS

                % Records how many mtDNA acquire second mutation per
                % stem cell

                Multiple = 0;

                if Mutated(iii) > 0

                    % Isolate the current stem cells mutational species

                    tempA = mtDNAmutations(time, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA)));

                    % Find all the WT mtDNA molecules

                    tempC = find(tempA == 0);

                    % For each new mutation, determine whether it is
                    % affecting a WT mtDNA or an already mutated mtDNA
                    % molecule

                    ttt = 1;
                    mutPos = zeros(1,Mutated(iii));

                    while ttt <= Mutated(iii)

                        tempZ = ffff(count6);
                        if isempty(find(mutPos == tempZ)) % For same number sequence in ffff
                check

                            mutPos(ttt) = tempZ;
                            count6 = count6 + 1;
                            ttt = ttt + 1;

                        else

                            count6 = count6 + 1;

                        end

                    end

                    % Which values of mutPos are not present in tempC

                    for ttt = 1 : numel(mutPos)
                        occuMut = find(tempC == mutPos(ttt));
                        if isempty(occuMut)
                            Multiple = Multiple + 1;
                        end
                    end

                    if Multiple > 0

                        % Find all current mutations

                        currMut = find(tempA > 0);

                        % Produce a random permutation of the indexed
                        % mutated mtDNA molecules

                        currMutRandom = currMut(randperm(numel(currMut)));

                        % The overwritten molecules species ID's will be

                        overSpeciesID = tempA(currMutRandom(1:Multiple));

                        % Determine the new species IDs for the mutated mtDNA molecules

                        tempE = origMut : (origMut + (Multiple-1));

                        % Take away the multiple mutations from the species
                        % ID generator vector

                        tempEMult = tempE(1 : Multiple); % Contains the multiple species ID

                        tempEWT = tempE((Multiple+1) : end); % Contains the normal species ID
```

```
            % Insert the new species ID into the current list
            % of species IDs in the WT molecule positions

            tempA(tempC(1:numel(tempEWT))) = tempEWT;

            % Replace the selected mtDNA species to be
            % overwritten for multiple mutations on same
            % mtDNA species.

            for ttt = 1 : Multiple
                a = find(tempA == overSpeciesID(ttt));
                tempA(a(end)) = tempEMult(ttt);
            end

            % insert the modified species ID vector into the
            % master matrix

            mtDNAmutations(time, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA))) = tempA;

            % This needs to be reflected in the MutatedAll
            % Array as well

            MutatedAll(time,iii) = MutatedAll(time,iii) - Multiple;

            % Increase the species ID tracker by one

            origMut = origMut + numel(tempE);

            % Recording the multiple mutation information
            % For each mutation, need to check whether it
            % has been muliplied before.

            for ttt = 1 : Multiple
                % determine whether the speciesID that is about
                % to be overwritten has any multiple mutations
                % already

                a = find(speciesIDRecord == overSpeciesID(ttt));

                if isempty(a)
                    speciesIDRecord(end+1) = tempEMult(ttt);
                    speciesIDMultRecord(end+1) = 2;
                else
                    speciesIDRecord(end+1) = tempEMult(ttt);
                    speciesIDMultRecord(end+1) = speciesIDMultRecord(a) + 1;
                end
            end

        else
```

```
                    % Determine the new species IDs for the mutated mtDNA molecules

                    tempE = origMut : (origMut + Mutated(iii)-1);

                    % Insert the new species ID into the current list
                    % of species IDs in the WT molecule positions

                    tempA(tempC(1:numel(tempE))) = tempE;

                    % insert the modified species ID vector into the
                    % master matrix

                    mtDNAmutations(time, (((iii*gg.mtDNA)-(gg.mtDNA-
1)):(iii*gg.mtDNA))) = tempA;

                    % increase the species ID tracker by one

                    origMut = origMut + numel(tempE);

                 end

              end

           end

        end

        % if there are some mutations in our system, then we see how they
        % propagate

        RelevantMutations = find(MutatedAll(time,:)>0);

        % if there are mutations present

        if sum(MutatedAll(time,:))>0

           % for each cell with a mutation present

           for jj = 1 : numel(RelevantMutations)

              % stem cell dividing (1 - asymmetric, 2 - symmetric 2 stem cells, 3 - symmetric
2 TA cells)

              if MutatedAll(time,RelevantMutations(jj)) >= gg.mutThreshold*gg.mtDNA
                 divisionType = bbbb(count2);
                 count2 = count2 + 1;
              else
                 divisionType = cccc(count3);
                 count3 = count3 + 1;
```

```
end

% mutated  mtDNA  loss and  gain  before  stem cell  division
% mutratedRep  - how  many  new mutated  mtDNAs  you get  in  the stem
% cell  after  doubling  the number  of mtDNA  molecules.

mutatedRep  = DiscSampVec2...
    ((0:gg.mtDNA),gg.RepProb...
    (MutatedAll(time,RelevantMutations(jj)),:),1);

% add new mtDNA  mutation  to old  ones

numMutated  = mutatedRep  + MutatedAll(time,RelevantMutations(jj));

% At this  point the  multiple  mutations  in
% mtDNAmutations  need to be increased  to
% the  numbers  that are in numMutated

tempA  = numMutated;
tempB  = mtDNAmutations(time,...
    (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
    (RelevantMutations(jj)*gg.mtDNA)));
tempC  = tempB(tempB>0);
tempD  = tempC(randi(numel(tempC),1,tempA));

% Store this  matrix  to a seperate variable

numMutatedMutations  = tempD;

% division  into  two cells,  each with  n mtDNA
% mutatedDiv  - how many  of the mutations  will  one cell
% get (the  other by proxy gets  all  the rest)

% Altered  for  DivProb  with  advantage...

if divisionType  == 1

    mutatedDiv  = DiscSampVec2...
        ((0:gg.mtDNA),gg.DivProb103...
        (numMutated,:),1);

else

    mutatedDiv  = DiscSampVec2...
        ((0:gg.mtDNA),gg.DivProb...
        (numMutated,:),1);

end

% Depeding  on the number  of mtDNA  molecules  go into  one
% cell, the other gets the other lot this is based on
% numMutatedMutations in the master cell before
% segregation

tempA = mutatedDiv;
tempB = numMutatedMutations(randperm(numel(numMutatedMutations)));

Cell1 = tempB(1:tempA);
Cell2 = tempB(tempA+1 : end);

if isempty(Cell1)
   Cell1 = 0;
end

if isempty(Cell2)
   Cell2 = 0;
end

% how many does the other cell have

vectorDiv = [mutatedDiv, numMutated - mutatedDiv];

% depending on the type of division, cells get kept or lost
% asymmetric division occurs, one cell gets lost, one remains

if divisionType == 1

   remainingCell = 2; % Advantage forces the mutatedDiv result to be the stem
cell

   count4 = count4 + 1;
   remained = vectorDiv(remainingCell);
   MutatedAll(time+1,RelevantMutations(jj)) = remained;

   if remainingCell == 1

      % insert the new cell multiple mutation data
      % depending on which cell is chosen for an
      % aysmmetric division fate outcome

      tempA = zeros(1,gg.mtDNA);
      tempA(1:numel(Cell1)) = Cell1;
      mtDNAmutations(time+1,...
         (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
         (RelevantMutations(jj)*gg.mtDNA))) = tempA;

   else

      tempA = zeros(1,gg.mtDNA);
      tempA(1:numel(Cell2)) = Cell2;
      mtDNAmutations(time+1,...
         (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
         (RelevantMutations(jj)*gg.mtDNA))) = tempA;

   end

   % symmetric division into 2 stem cells, both are kept

elseif divisionType == 2

   remained1 = vectorDiv(1);
   remained2 = vectorDiv(2);
   MutatedAll(time+1,RelevantMutations(jj)) = remained1;

   % insert the new cell multiple mutation data for
   % the stem cell that stays for the symmetric fate
   % outcome 1

   tempA = zeros(1,gg.mtDNA);
   tempA(1:numel(Cell1)) = Cell1;
   mtDNAmutations(time+1,...
      (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
      (RelevantMutations(jj)*gg.mtDNA))) = tempA;

   % which one of the other cells will it replace?

   a = 1:gg.initS;
   possibleReplacements = a(a ~= RelevantMutations(jj));
   b = ceil((gg.initS-1)*eeee(count5));
   count5 = count5+1;
   c = possibleReplacements(b);
   MutatedAll(time+1,c) = remained2;

   % insert the new cell multiple mutation data for
   % the stem cell that stays for the symmetric fate
   % outcome 2

   tempA = zeros(1,gg.mtDNA);
   tempA(1:numel(Cell2)) = Cell2;
   mtDNAmutations(time+1,...
      (((c*gg.mtDNA)-(gg.mtDNA-1)):...
      (c*gg.mtDNA))) = tempA;

   % symmetric division into 2 TA cells, none are kept

elseif divisionType == 3

   % which of the other ones gets doubled?

   a = 1:gg.initS;
   possibleReplacements = a(a ~= RelevantMutations(jj));
```

```
                b = ceil((gg.initS-1)*eeee(count5));
                count5 = count5+1;
                c = possibleReplacements(b);
                MutatedAll(time+1,RelevantMutations(jj)) = MutatedAll(time,c);

                % insert the new cell multiple mutation data for
                % the stem cell that stays for the symmetric fate
                % outcome 2

                mtDNAmutations(time+1,(((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-
1)):...
                    (RelevantMutations(jj)*gg.mtDNA))) = ...
                    mtDNAmutations(time,(((c*gg.mtDNA)-(gg.mtDNA-1)):...
                    (c*gg.mtDNA)));

            end

        end

    end

    %%%%%%%%%%%%%% COX DEF SC RATE
INCREASE %%%%%%%%%%%%%%%%%%%%%%

    if strcmp(gg.COXSCTimePoint, 'Yes') == 1

        % Run just the DIVISION CODE again for COX neg stem cells at
        % specific time points

        % Run code every n timepoints

        if mod(time,gg.COXSCTimePointInterval) == 0

            COXDefCycle = 0;

            while COXDefCycle < gg.COXDefCycleRepeats

                blueSCPres = find(MutatedAll(time+1,:)>=(gg.mtDNA*gg.mutThreshold));

                if ~isempty(blueSCPres)

                    RelevantMutations = blueSCPres;

                    % if there are mutations present
                    if sum(MutatedAll(time+1,:))>0

                        % for each cell with a mutation present
                        for jj = 1 : numel(RelevantMutations)
```

```
                  % stem cell dividing (1 - asymmetric, 2 - symmetric 2 stem cells, 3 -
symmetric 2 TA cells)

                  if MutatedAll(time+1,RelevantMutations(jj)) >=
gg.mutThreshold*gg.mtDNA
                    divisionType = bbbb(count2);
                    count2 = count2 + 1;
                  else
                    divisionType = cccc(count3);
                    count3 = count3 + 1;
                  end

                  % mutated mtDNA loss and gain before stem cell division
                  % mutratedRep - how many new mutated mtDNAs you get in the stem
                  % cell after doubling the number of mtDNA molecules.

                  mutatedRep = DiscSampVec2...
                    ((0:gg.mtDNA),gg.RepProb...
                    (MutatedAll(time+1,RelevantMutations(jj)),:),1);

                  % add new mtDNA mutation to old ones

                  numMutated = mutatedRep +
MutatedAll(time+1,RelevantMutations(jj));

                  % At this point the multiple mutations in
                  % mtDNAmutations need to be increased to
                  % the numbers that are in numMutated

                  tempA = numMutated;
                  tempB = mtDNAmutations(time+1,...
                    (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
                    (RelevantMutations(jj)*gg.mtDNA)));
                  tempC = tempB(tempB>0);
                  tempD = tempC(randi(numel(tempC),1,tempA));

                  % Store this matrix to a seperate variable

                  numMutatedMutations = tempD;

                  % division into two cells, each with n mtDNA
                  % mutatedDiv - how many of the mutations will one cell get (the
                  % other by proxy gets all the rest)

                  % Altered for DivProb with advantage...

                  if divisionType == 1

                    mutatedDiv = DiscSampVec2...
                      ((0:gg.mtDNA),gg.DivProb103...
                      (numMutated,:),1);

                  else

                    mutatedDiv = DiscSampVec2...
                      ((0:gg.mtDNA),gg.DivProb...
                      (numMutated,:),1);

                  end

                  % Depeding on the number of mtDNA molecules go into one
                  % cell, the other gets the other lot this is based on
                  % numMutatedMutations in the master cell before
                  % segregation

                  tempA = mutatedDiv;
                  tempB =
numMutatedMutations(randperm(numel(numMutatedMutations)));

                  Cell1 = tempB(1:tempA);
                  Cell2 = tempB(tempA+1 : end);

                  if isempty(Cell1)
                    Cell1 = 0;
                  end

                  if isempty(Cell2)
                    Cell2 = 0;
                  end

                  % how many does the other cell have

                  vectorDiv = [mutatedDiv, numMutated - mutatedDiv];

                  % depending on the type of division, cells get kept or lost
                  % asymmetric division occurs, one cell gets lost, one remains

                  if divisionType == 1

                    remainingCell = 2; % Advantage forces the mutatedDiv result to be
the stem cell
                    count4 = count4 + 1;
                    remained = vectorDiv(remainingCell);
                    MutatedAll(time+1,RelevantMutations(jj)) = remained;

                    if remainingCell == 1

                      % insert the new cell multiple mutation data
                      % depending on which cell is chosen for an
                      % aysmmetric division fate outcome
                      tempA = zeros(1,gg.mtDNA);
                      tempA(1:numel(Cell1)) = Cell1;
                      mtDNAmutations(time+1,...
                          (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
                          (RelevantMutations(jj)*gg.mtDNA))) = tempA;

                    else

                      tempA = zeros(1,gg.mtDNA);
                      tempA(1:numel(Cell2)) = Cell2;
                      mtDNAmutations(time+1,...
                          (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
                          (RelevantMutations(jj)*gg.mtDNA))) = tempA;

                    end

                    % symmetric division into 2 stem cells, both are kept

                  elseif divisionType == 2

                    remained1 = vectorDiv(1);
                    remained2 = vectorDiv(2);
                    MutatedAll(time+1,RelevantMutations(jj)) = remained1;

                    % insert the new cell multiple mutation data for
                    % the stem cell that stays for the symmetric fate
                    % outcome 1


                    tempA = zeros(1,gg.mtDNA);
                    tempA(1:numel(Cell1)) = Cell1;
                    mtDNAmutations(time+1,...
                      (((RelevantMutations(jj)*gg.mtDNA)-(gg.mtDNA-1)):...
                      (RelevantMutations(jj)*gg.mtDNA))) = tempA;

                    % which one of the other cells will it replace?

                    a = 1:gg.initS;
                    possibleReplacements = a(a ~= RelevantMutations(jj));
                    b = ceil((gg.initS-1)*eeee(count5));
                    count5 = count5+1;
                    c = possibleReplacements(b);
                    MutatedAll(time+1,c) = remained2;

                    % insert the new cell multiple mutation data for
                    % the stem cell that stays for the symmetric fate
                    % outcome 2

                    tempA = zeros(1,gg.mtDNA);
```

```
                    tempA(1:numel(Cell2)) = Cell2;
                    mtDNAmutations(time+1,...
                        (((c*gg.mtDNA)-(gg.mtDNA-1)):...
                        (c*gg.mtDNA))) = tempA;

                    % symmetric division into 2 TA cells, none are kept

                elseif divisionType == 3

                    % which of the other ones gets doubled?

                    a = 1:gg.initS;
                    possibleReplacements = a(a ~= RelevantMutations(jj));
                    b = ceil((gg.initS-1)*eeee(count5));
                    count5 = count5+1;
                    c = possibleReplacements(b);
                    MutatedAll(time+1,RelevantMutations(jj)) = MutatedAll(time+1,c);

                    % insert the new cell multiple mutation data for
                    % the stem cell that stays for the symmetric fate
                    % outcome 2

                    mtDNAmutations(time+1,(((RelevantMutations(jj)*gg.mtDNA)-
(gg.mtDNA-1)):...
                        (RelevantMutations(jj)*gg.mtDNA))) = ...
                        mtDNAmutations(time+1,(((c*gg.mtDNA)-(gg.mtDNA-1)):...
                        (c*gg.mtDNA)));

                end

            end

        end


    end

    COXDefCycle = COXDefCycle + 1;

  end

 end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

time = time + 1;
```

```
    end

end

%% METRICS mtDNA clonal expansion

% Identifying successul and failed mtDNA clonal expansion events.

% Insert an extra column into MutatedAll so if a mutation appears at the
% first time point then the difference is captured

MutBuffer = zeros(1,gg.initS);

MutatedAll2 = [MutBuffer; MutatedAll];

% Find the difference between mutation events and clonal expansion

Difference = zeros(gg.numDiv+1,gg.initS);

for ii = 1 : gg.initS  % InitS
    for tt = 1 : gg.numDiv-1
        Difference(tt+1,ii) = MutatedAll2(tt+1,ii) - MutatedAll2(tt,ii);
    end
end

% Find where the 1 mtDNA values are for all stem cells in the niche

for ll = 1 : gg.initS

    PrimaryName = ['PrimaryMut' num2str(ll)];

    PrimaryMut = find(MutatedAll2(:,ll) > 0)';

    str = [PrimaryName, '=PrimaryMut;'];

    eval(str)

    a = eval(['PrimaryMut' num2str(ll)]);

    for uu = 1 : numel(a);

        if Difference(a(uu),ll) == MutatedAll2(a(uu),ll);
            a(uu) = a(uu);
        else
            a(uu) = 0;
        end

    end

    a(a == 0) = [];
```

```
        str = [PrimaryName,  '=a;'];
        eval(str)

    end

    % Find where the end of the clonal expansion is if it does have an end

    for ll = 1 : gg.initS

        PrimaryName  = ['PrimaryEndMut'  num2str(ll)];

        PrimaryEndMut  = find(MutatedAll2(:,ll)  == 0)';

        str = [PrimaryName,  '=PrimaryEndMut;'];

        eval(str)

        a = eval(['PrimaryEndMut'  num2str(ll)]);

        for uu = 2 : numel(a);

            if Difference(a(uu),ll)  == MutatedAll2(a(uu)-1,ll)*-1  && Difference(a(uu),ll)  ~= 0
                a(uu)  = a(uu);
            else
                a(uu)  = 0;
            end

        end

        a(a == 0) = [];
        a(a == 1) = [];

        str = [PrimaryName,  '=a;'];
        eval(str)

    end

    % Find the failed clonal expansions and the times

    for ll = 1:gg.initS

        a = eval(['PrimaryMut'  num2str(ll)]);
        b = eval(['PrimaryEndMut'  num2str(ll)]);

        start = numel(a);
        finish  = numel(b);

        if start == finish
            for tt = 1 : start
                vv = b(tt) - a(tt);
                if vv >= 0
                    gg.FailedCE(1,end+1) = vv;
                    gg.FailedCE(2,end) = a(tt);
                    gg.FailedCE(3,end) = b(tt);
                end
            end
        end

        if start > finish

            for jj = 1 : finish
                gg.FailedCE(1,end+1) = b(jj) - a(jj);
                gg.FailedCE(2,end) = a(jj);
                gg.FailedCE(3,end) = b(jj);
            end

            for jj = start
                c = find(MutatedAll2(a(jj):gg.numDiv,ll) == gg.mtDNA);
                d = min(c) + a(jj) - 1;

                if isempty(c) % For those that are still transient
                else
                    vv = d - a(jj);
                    if vv >= 0
                        gg.SuccessCE(1,end+1) = vv + 1;
                        gg.SuccessCE(2,end) = a(jj);
                        gg.SuccessCE(3,end) = d;
                    end
                end

            end
        end
    end

    %% METRICS SC Niche Succession

    stemCellAll = zeros(gg.numDiv+1,1);

    for ii = 1 : gg.numDiv
        stemCellAll(ii+1,1) = numel(find(MutatedAll(ii,:) >= gg.mutThreshold*gg.mtDNA));
    end

    % Find the difference between mutation SC and niche succession

    SCDifference = zeros(gg.numDiv+1,1);

    for tt = 1 : gg.numDiv - 1
        SCDifference(tt+1,1) = stemCellAll(tt+1,1) - stemCellAll(tt,1);
    end
```

```
 % Make both the Difference and the MutatedAll the same size

 SCPrimaryMut = find(stemCellAll > 0)';

 for uu = 1 : numel(SCPrimaryMut);

     if SCDifference(SCPrimaryMut(uu),1) == stemCellAll(SCPrimaryMut(uu),1);
         SCPrimaryMut(uu) = SCPrimaryMut(uu);
     else
         SCPrimaryMut(uu) = 0;
     end

 end

 SCPrimaryMut(SCPrimaryMut == 0) = [];

 % Find where the end of the clonal expansion is if it does have an end

 SCPrimaryEndMut = find(stemCellAll == 0)';

 for uu = 2 : numel(SCPrimaryEndMut);

     if SCDifference(SCPrimaryEndMut(uu),1) == stemCellAll(SCPrimaryEndMut(uu)-
1,1)*-1 && SCDifference(SCPrimaryEndMut(uu),1) ~= 0
         SCPrimaryEndMut(uu) = SCPrimaryEndMut(uu);
     else
         SCPrimaryEndMut(uu) = 0;
     end

 end

 SCPrimaryEndMut(SCPrimaryEndMut == 0) = [];
 SCPrimaryEndMut(SCPrimaryEndMut == 1) = [];


 % Find the failed clonal expansions and the times

 start = numel(SCPrimaryMut);
 finish = numel(SCPrimaryEndMut);

 if start == finish
     for tt = 1 : start
         gg.NicheFailedSC(1,end+1) = SCPrimaryEndMut(tt) - SCPrimaryMut(tt);
         gg.NicheFailedSC(2,end) = SCPrimaryMut(tt);
         gg.NicheFailedSC(3,end) = SCPrimaryEndMut(tt);
     end
 end

 if start > finish
```

```
    for jj = 1 : finish
        gg.NicheFailedSC(1,end+1) = SCPrimaryEndMut(jj) - SCPrimaryMut(jj);
        gg.NicheFailedSC(2,end) = SCPrimaryMut(jj);
        gg.NicheFailedSC(3,end) = SCPrimaryEndMut(jj);
    end
    for jj = start
        c = find(stemCellAll(SCPrimaryMut(jj):gg.numDiv,1) == gg.initS);
        d = min(c) + SCPrimaryMut(jj) - 1;

        if isempty(c)
        else
            vv = d - SCPrimaryMut(jj);
            if vv >= 0
                gg.NicheSuccessSC(1,end+1) = d - SCPrimaryMut(jj) + 1;
                gg.NicheSuccessSC(2,end) = SCPrimaryMut(jj) + 1;
                gg.NicheSuccessSC(3,end) = d;
            end
        end
    end
end

%% Correction factor for mtDNAmutations and mutatedAll

% This part of the code affects both mtDNAmutations and mutatedAll in
% order to affect MutatedSCAge to determine how much COX deficiency
% will be present after the correction factor has been implemented.
% This will run alongside the current code so there is a measure of the
% affect the correction factor has

% Determine the max number of mutations present

maxMut = max(max(mtDNAmutations));

% Generate each speciesID

maxSpecies = 1 : maxMut;

% For every species ID thats present in specesIDRecord, delete from
% maxSpecies

for vv = 1 : numel(speciesIDRecord)
    maxSpecies(maxSpecies == speciesIDRecord(vv)) = [];
end

% Determine which numbers need to be excluded from the list present,
% need to use a random number generator

maxRand = rand(1,numel(maxSpecies));

corrPos = maxRand <= gg.COXCorrectionFactor;
```

```
exclSpecies1 = maxSpecies(corrPos);

% Now for the species that have multiple mutations present. Each
% mutation has to be assessed individually

% Generate the number of random numbers required for each mutation

multRand = rand(1,sum(speciesIDMultRecord));

% Go through each species with multiple mutations and see if any
% dont contain any COX deficiency mutation

exclSpecies2 = [];

for vv = 1 : numel(speciesIDRecord)

    if min(multRand(1:speciesIDMultRecord(vv))) <= (gg.COXCorrectionFactor)

        exclSpecies2(end+1) = speciesIDRecord(vv);

    end

    multRand(1:speciesIDMultRecord(vv)) = [];

end

% Combine both exclSpecies and exclSpecies2 which contain the species
% IDs that are to be excluded from mtDNAmutations.

exclSpecies = [exclSpecies1 exclSpecies2];

% Delete the numbers that are present in corrPos from mtDNAmutations

mtDNAmutationsCorr = mtDNAmutations;

for vv = 1 : numel(exclSpecies)
    mtDNAmutationsCorr(mtDNAmutationsCorr == exclSpecies(vv)) = 0;
end

%% Correction factor for mtDNAmutations and mutatedAll - Adjusted

% This part of the code affects both mtDNAmutations and mutatedAll in
% order to affect MutatedSCAge to determine how much COX deficiency
% will be present after the correction factor has been implemented.
% This will run alongside the current code so there is a measure of the
% affect the correction factor has.

% Set up the vector that is going to record the mtDNAspecies that are
% homoplasmic within the cell.
```

```
  homoplas_mtDNASpecies = [];

  % For each age and for each stem cell determine where the homoplasmic
  % mutations are.

  for vv = 1 : gg.numDiv

    for bb = 1 : gg.initS

      % Determine the vector to be assessed (stem cell at timepoint)

      vectorCorr = mtDNAmutationsCorr(vv, (((bb*gg.mtDNA)-(gg.mtDNA-
1)):(bb*gg.mtDNA)));

      % What are the unique values present within this

      vectorCorr_unique = unique(vectorCorr);

      vectorCorr_unique(vectorCorr_unique == 0) = [];

      % For each unique speciesID, what is the %

      for jj = 1 : numel(vectorCorr_unique)

        vectorCorr_number = numel(find(vectorCorr == vectorCorr_unique(jj)));
        vectorCorr_percentage = vectorCorr_number / gg.mtDNA * 100;

        if vectorCorr_percentage == 100

          % Set what happens when there is a homoplasmic mtDNA
          % species present -- It gets recorded into a new vector

          homoplas_mtDNASpecies(end+1) = vectorCorr_unique(jj);

        end

      end

    end

  end

  % Need to get rid of repeated values in order

  homoplas_mtDNASpecies = unique(homoplas_mtDNASpecies);

  %% Need to remove the species IDs that dont satisfy the inclusion criteria

  homoplas_mtDNASpecies_post = homoplas_mtDNASpecies(rand(1,...
        numel(homoplas_mtDNASpecies)) <= gg.COXCorrectionFactor2);

    % Delete the numbers that are present in homoplas_mtDNASpecies_post from
mtDNAmutations

    mtDNAmutationsCorr2 = mtDNAmutationsCorr;

    for vv = 1 : numel(homoplas_mtDNASpecies_post)
        mtDNAmutationsCorr2(mtDNAmutationsCorr2 ==
homoplas_mtDNASpecies_post(vv)) = 0;
    end


    %% Correction Factor Integration

    % Now that the correction factor has been implemented, we need to
    % determine, for each age, for each stem cell, the new number of mtDNA

    % mtDNAmutationsCorr summed up in MutatedAllCorr

    MutatedAllCorr = zeros(gg.numDiv,gg.initS);

    for vv = 1 : gg.numDiv

        for uu = 1 : gg.initS

            section = mtDNAmutationsCorr(vv,((gg.mtDNA*uu) - (gg.mtDNA-1)) :
(gg.mtDNA*uu));
            speciesPres = find(section > 0);
            numSpeciesPresent = numel(speciesPres);
            MutatedAllCorr(vv,uu) = numSpeciesPresent;

        end

    end

    % Main Output for correctionFactorResult

    for uu = 1 : gg.numDiv
        Mut = find(MutatedAllCorr(uu,:) >= (gg.mtDNA*gg.mutThreshold));
        MutNo = numel(Mut);
        MutatedSCAgeCorr(uu,pp) = MutNo;
    end

    %% Correction Factor 2 Integration

    % Now that the correction factor has been implemented, we need to
    % determine, for each age, for each stem cell, the new number of mtDNA

    % mtDNAmutationsCorr2 summed up in MutatedAllCorr2
```

```
MutatedAllCorr2 = zeros(gg.numDiv,gg.initS);

for vv = 1 : gg.numDiv

    for uu = 1 : gg.initS

        section = mtDNAmutationsCorr2(vv,((gg.mtDNA*uu) - (gg.mtDNA-1)) :
(gg.mtDNA*uu));
        speciesPres = find(section > 0);
        numSpeciesPresent = numel(speciesPres);
        MutatedAllCorr2(vv,uu) = numSpeciesPresent;

    end

end

% Main Output for correctionFactorResult

for uu = 1 : gg.numDiv
    Mut = find(MutatedAllCorr2(uu,:) >= (gg.mtDNA*gg.mutThreshold));
    MutNo = numel(Mut);
    MutatedSCAgeCorr2(uu,pp) = MutNo;
end

%% Main Output

% How many stem cells at each age have a pathogenic mutation present.

for uu = 1 : gg.numDiv
    Mut = find(MutatedAll(uu,:) >= (gg.mtDNA*gg.mutThreshold));
    MutNo = numel(Mut);
    MutatedSCAge(uu,pp) = MutNo;
end

%% Crypt Fission Events?

% The number of stem cells that are mutated in this crypt during its
% lifetime

SCMutatedNo = MutatedSCAge(:,pp);

% Primed scalar vector to record when and where a crypt fission event
% occurs

CryptFissionEvent = zeros(gg.numDiv,1);

% For each division that has occured, determine what the crypt
% fission probability is dependent on the number of stem cells that
% are mutated. Determine if fission does occur and record it in the
```

```
  % CryptFissionEvent vector.

  for hh = 1 : gg.numDiv

    SCMut = SCMutatedNo(hh,1);

    if SCMut == 0 && gggg(count7) < gg.cryptFissionProb
      CryptFissionEvent(hh,1) = 1;
    end

    count7 = count7 + 1;

    if SCMut > 0 && gggg(count7) < gg.cryptFissionProb*gg.cryptFissionFactor*SCMut
      CryptFissionEvent(hh,1) = 1;
    end

    count7 = count7 + 1;

  end

  if sum(CryptFissionEvent) > 0

    FissionAge = find(CryptFissionEvent == 1);

    for rr = 1 : numel(FissionAge)

      MutatedAllData = MutatedAll(1:FissionAge(rr),:);
      cryptFisSaveNo = ['MutatedAllData' num2str(gg.cryptFisSave)];
      str = [cryptFisSaveNo, '= MutatedAllData;'];
      eval(str)
      save(gg.filename,...
        (['MutatedAllData' num2str(gg.cryptFisSave)]),'-append');
      gg.cryptFisSave = gg.cryptFisSave + 1;
    end

  end

  %% To match the biological data I need to identify the clonally
  % expanded mutations (>25% heteroplasmy) at 70 years of age
  % equivalent to 3647 numDivs.

  for cc = 1 : gg.initS

    speciesPresent(:,cc) = mtDNAmutations(3647,((gg.mtDNA*cc)-(gg.mtDNA-
1)):(gg.mtDNA*cc))';

  end

  % speciesPresent now gives the mutation
  % For each stem cell,find unique values and see if any of them are over 25%
```

```
SingleMut  = zeros(1,gg.initS);
MultipleMut  = zeros(1,gg.initS);

for cc = 1 : gg.initS

    % Clonally  expanded  point  mutation  present?

    temp  = unique(speciesPresent(:,cc));
    temp(temp==0)  = [];

    % temp contains  all the mtDNA  mutations  that are present at the age
    % of 70 years. Need to know if any of these are present in
    % speciesIDRecord.

    if isempty(temp)

    else

        for xx = 1 : numel(temp)

            temp2  = numel(find(speciesPresent(:,cc)  == temp(xx)));
            temp3  = temp2  / gg.mtDNA  * 100;

            findDouble  = find(speciesIDRecord  == temp(xx));

            if temp3 > 25 && isempty(findDouble)
                SingleMut(cc)  = SingleMut(cc)  + 1;
                MultipleMut(cc)  = MultipleMut(cc)  + 1;

            elseif  temp3 > 25 && ~isempty(findDouble)
                MultipleMut(cc)  = MultipleMut(cc)  + speciesIDMultRecord(findDouble);

            end

        end

    end

end

SingleMutRecord(pp,:)  = SingleMut;
MultipleMutRecord(pp,:)  = MultipleMut;

% Work out the probability  for each age (numDivs)  that there will  be a
% mutation  present

for tt = 1 : gg.initS
    for ii = 1 : gg.numDiv
        mutProbAge(2,ii)  = mutProbAge(2,ii)  + 1;
```

```
            if MutatedAll(ii,tt) >= gg.mtDNA*gg.mutThreshold
                mutProbAge(1,ii) = mutProbAge(1,ii) + 1;
            end
        end
    end

    % Update waitbar

    waitbar(pp/gg.numRuns,h)

end

delete(h)

gg.FailedCE(:,1) = [];
gg.SuccessCE(:,1) = [];

FailedClonal = gg.FailedCE;
SuccessClonal = gg.SuccessCE;


gg.NicheFailedSC(:,1) = [];
gg.NicheSuccessSC(:,1) = [];

NicheFailed = gg.NicheFailedSC;
NicheSuccess = gg.NicheSuccessSC;

% Make single and multiple mutation records spit out similar data to
% Biological Data

a = find(SingleMutRecord > 0);
a1 = find(MultipleMutRecord > 0);

b = numel(a);
b1 = numel(a1);

for kk = 1 : 20

    c = find(SingleMutRecord == kk);
    c1 = find(MultipleMutRecord == kk);

    SingleMutRecordResult(kk) = (numel(c)) / b * 100;
    MultipleMutRecordResult(kk) = (numel(c1)) / b1 * 100;

end

% Save the single and multiple mtDNA mutation data

gg.SingleMutRecordResult = SingleMutRecordResult;
gg.MultipleMutRecordResult = MultipleMutRecordResult;
```

```
% Save the mutation probabilities by age

mutProbAgeFinal = mutProbAge(1,:) ./ mutProbAge(2,:) * 100;

gg.mutProbAgeFinal = mutProbAgeFinal;

end
```

1.4.2.3. *Part 3*

```
function [MutatedSCAgeFission,CryptFisTime2] =
mtDNACrypt_ConstantV2FCN_CF_Input(s3)

% mtDNACrypt Function for Constant and Increasing Mutation Rate
% - Crypt Fission - Single Crypts - FINAL

% The script is an amalgamation of the previous crypt model. It identifies
% that there are a certain number of mtDNA molecules residing within each
% stem cell of the crypt. With the evolution of stem cell divisions, the
% number of mutated mtDNA molecules evolves stochastically according to
% pre-determined probabilities. Also, with each additional mutated mtDNA
% molecule, the model determines which kind of mutation has developed
% according to probability data previously acquired. Therefore, this model
% is a more accurate representation of the processes that take place within
% the crypt and at the tissue level.

global gg
global dd

MutatedSCAgeFission = zeros(gg.numDiv,1);

% we start with all cells/mtDNA mutation free
MutatedAll = zeros(gg.numDiv,gg.initS);

% Find out the time at which the crypt fission event arose
CryptFisTime = size(s3);
CryptFisTime2 = CryptFisTime(1);

% Insert the data about the old crypt into the
MutatedAll(1:CryptFisTime(1),1:gg.initS) = s3;

% initiate time, time+1 means divTime has passed
time = CryptFisTime2;

%% Pre-determined random numbers for crypt simulation

% Mutation Rate random numbers
aaaa = DiscSampVec3((0:1),[gg.mutationRate1],(gg.mtDNA*gg.initS));

% Stem cell division type random numbers
bbbb = DiscSampVec2((1:3),[gg.Pa,gg.Ps,gg.Ps],gg.numDiv*gg.initS*2);
count2 = 1;

% Stem cell division type with advantage random numbers
cccc = DiscSampVec2((1:3),[gg.Pa-((gg.Ps*gg.adv)-
gg.Ps),gg.Ps*gg.adv,gg.Ps],gg.numDiv*gg.initS*2);
count3 = 1;
```

```
% Segregation event random numbers
dddd = DiscSampVec2((1:2),[0.5,0.5],gg.numDiv*gg.initS*2);
count4 = 1;

% Stem cell replacement random numbers
eeee = rand(1,(gg.numDiv*gg.initS*2));
count5 = 1;

% Crypt Fission Crypt Numbering
ffff = rand(1,2*gg.numDiv);
count6 = 1;

%% Simulate only for certain time
while time < gg.numDiv

    if time == 1
        b = 0;
    else
        a = sum(MutatedAll(time,:));
        b = a > 0;
    end

    %% mutations occuring
    % random numbers generated for each mtDNA molecule
    % within all stem cells of the crypt to determine how many are
    % mutated

    if b == 0

        Mutated = [];

        for iii = 1:gg.initS

            Mutated(iii) = sum(aaaa(time,((iii*gg.mtDNA)-(gg.mtDNA-1)):...
                    ((iii*gg.mtDNA)-(gg.mtDNA-1)) + (gg.mtDNA-1)));

        end

        MutatedAll(time+1,:) = MutatedAll(time,:) + Mutated;

        time = time + 1;

    end

    if b > 0

        %% mutations occuring
        % random numbers generated for each mtDNA molecule
        % within all stem cells of the crypt to determine how many are
        % mutated
        Mutated = [];

        for iii = 1:gg.initS

          Mutated(iii) = sum(aaaa(time,((iii*gg.mtDNA)-(gg.mtDNA-1)):...
                    ((iii*gg.mtDNA)-(gg.mtDNA-1)) + (gg.mtDNA-1)));

        end

        MutatedAll(time+1,:) = MutatedAll(time,:) + Mutated;

        % if there are some mutations in our system, then we see how they
        % propagate
        RelevantMutations = find(MutatedAll(time,:)>0);

        % if there are mutations present
        if sum(MutatedAll(time,:))>0

            % for each cell with a mutation present
            for jj = 1 : numel(RelevantMutations)

               %%stem cell dividing (1 - asymmetric, 2 - symmetric 2 stem cells, 3 - symmetric 2
TA cells)

               if MutatedAll(time,RelevantMutations(jj)) > gg.mutThreshold*gg.mtDNA
                    divisionType = bbbb(count2);
                    count2 = count2 + 1;
               else
                    divisionType = cccc(count3);
                    count3 = count3 + 1;
               end

               % mutated mtDNA loss and gain before stem cell division
               % mutatedRep - how many new mutated mtDNAs you get in the stem
               % cell after doubling the number of mtDNA molecules.

               mutatedRep = DiscSampVec2...
                    ((0:gg.mtDNA),gg.RepProb...
                    (MutatedAll(time,RelevantMutations(jj)),:),1);

               % add new mtDNA mutation to old ones

               numMutated = mutatedRep + MutatedAll(time,RelevantMutations(jj));

               % division into two cells, each with n mtDNA
               % mutatedDiv - how many of the mutations will one cell get (the
               % other by proxy gets all the rest)

               mutatedDiv = DiscSampVec2...
```

```
                ((0:gg.mtDNA),gg.DivProb...
                (numMutated,:),1);

            % how many does the other cell have

            vectorDiv = [mutatedDiv, numMutated - mutatedDiv];

            % depending on the type of division, cells get kept or lost
            % asymmetric division occurs, one cell gets lost, one remains

            if divisionType == 1
                remainingCell = dddd(count4);
                count4 = count4 + 1;
                remained = vectorDiv(remainingCell);
                MutatedAll(time+1,RelevantMutations(jj)) = remained;

                % symmetric division into 2 stem cells, both are kept
            elseif divisionType == 2
                remained1 = vectorDiv(1);
                remained2 = vectorDiv(2);
                MutatedAll(time+1,RelevantMutations(jj)) = remained1;

                % which one of the other cells will it replace?
                a = 1:gg.initS;
                possibleReplacements = a(a ~= RelevantMutations(jj));
                b = ceil((gg.initS-1)*eeee(count5));
                count5 = count5+1;
                c = possibleReplacements(b);
                MutatedAll(time+1,c) = remained2;

                % symmetric division into 2 TA cells, none are kept
            elseif divisionType == 3

                % which of the other ones gets doubled?
                a = 1:gg.initS;
                possibleReplacements = a(a ~= RelevantMutations(jj));
                b = ceil((gg.initS-1)*eeee(count5));
                count5 = count5+1;
                c = possibleReplacements(b);
                MutatedAll(time+1,RelevantMutations(jj)) = MutatedAll(time,c);
            end
        end
    end
    time = time + 1;
  end
end

%% Main Output

% How many stem cells at each age have a pathogenic mutation present.
```

```
for uu = 1 : gg.numDiv
    Mut = find(MutatedAll(uu,:) > (gg.mtDNA*gg.mutThreshold));
    MutNo = numel(Mut);
    MutatedSCAgeFission(uu,1) = MutNo;
end

%% Crypt Fission Events?

% The number of stem cells that are mutated in this crypt during its
% lifetime

SCMutatedNo = MutatedSCAgeFission(:,1);

% Primed scalar vector to record when and where a crypt fission event
% occurs

CryptFissionEvent = zeros(gg.numDiv,1);

% For each division that has occured, determine what the crypt
% fission probability is dependent on the number of stem cells that
% are mutated. Determine if fission does occur and record it in the
% CryptFissionEvent vector.

for hh = CryptFisTime2 : gg.numDiv

    SCMut = SCMutatedNo(hh,1);

    if SCMut == 0 && ffff(count6) < gg.cryptFissionProb
        CryptFissionEvent(hh,1) = 1;
    end

    count6 = count6 + 1;

    if SCMut > 0 && ffff(count6) < gg.cryptFissionProb*gg.cryptFissionFactor*SCMut
        CryptFissionEvent(hh,1) = 1;
    end

    count6 = count6 + 1;

end

if sum(CryptFissionEvent) > 0

    FissionAge = find(CryptFissionEvent == 1);

    for rr = 1 : numel(FissionAge)

        MutatedAllData = MutatedAll(1:FissionAge(rr),:);
        cryptFisSaveNo = ['dd.MutatedAllData' num2str(gg.cryptFisSave)];
        str = [cryptFisSaveNo,  '= MutatedAllData;'];
        eval(str)
        gg.cryptFisSave  = gg.cryptFisSave  + 1;

    end

end

end
```

1.4.3. **Essential functions for niche succession model**

1.4.3.1. *Discrete probability generation*

```
%% Random number generator from a user defined discrete probability
% distribution

% x - vector of outcomes
% p - vector of outcome probabilities
% ns - how many random numbers you need

function S = DiscSampVec2(x,p,ns)

[~,idx] = histc(rand(1,ns),[0,cumsum(p)]);
S = x(idx);
```

1.4.3.2. *Discrete probability generation for increasing mutation rate*

```
%% Random number generator from a user defined discrete probability
% distribution

% x - vector of outcomes
% p - vector of outcome probabilities
% ns - how many random numbers you need


function S = DiscSampVec3(x,p,ns)

global gg

S = zeros(gg.numDiv,gg.mtDNA*gg.initS);

for ii = 1 : gg.numDiv

    pVec = [1-p(ii),p(ii)];

    [~,idx] = histc(rand(1,ns),[0,cumsum(pVec)]);
    S(ii,:) = x(idx);

end

end
```

1.4.3.3. *Un-nesting structured field names*

```
function struct2var(s)

%STRUCT2VAR Convert structure array to workspace variables.
%   STRUCT2VAR(S) converts the M-by-N structure S (with P fields)
%   into P variables defined by fieldnames with dimensions M-by-N. P
%   variables are placed in the calling workspace.

if nargin < 1
    error('struct2var:invalid','No input structure')
elseif nargin > 1
    error('struct2var:invalidt','Too many inputs')
elseif ~isstruct(s)
    error('struct2var:invalid','Input needs to be a structure data type')
end

[r,c] = size(s);
names = fieldnames(s);

for i=1:length(names)
    assignin('caller',names{i},s.(names{i}))
end
```

1.4.3.4. *Graphing niche succession model results*

```
%% GraphNicheSuccessionResults
% Graphs the results of the niche succession simulations

% Need to open the Model results file first for the gg global variable

PercentageSCAge = zeros(gg.numDiv,(gg.initS+1));

h = waitbar(0,'Analysing results within parameter file...please wait');

for jj = 1 : gg.numDiv

    for mm = 1 : gg.initS+1

        Pera = find(gg.MutatedSCAgeFinal(jj,:) == (mm-1));
        % Pera = find(gg.MutatedSCAgeFinalCorr(jj,:) == (mm-1));
        % Pera = find(gg.MutatedSCAgeFinalCorr2(jj,:) == (mm-1));
        Perb = (numel(Pera) / gg.numRuns)*100;
        PercentageSCAge(jj,mm) = Perb;

    end

    waitbar(jj/gg.numDiv,h)

end

delete(h)

% mean the results to attain a results table comparible to experimental
% results

tenYears = mean(PercentageSCAge(1:521,:));
twentyYears = mean(PercentageSCAge(522:1042,:));
thrityYears = mean(PercentageSCAge(1043:1563,:));
fortyYears = mean(PercentageSCAge(1564:2084,:));
fiftyYears = mean(PercentageSCAge(2085:2605,:));
sixtyYears = mean(PercentageSCAge(2606:3126,:));
seventyYears = mean(PercentageSCAge(3127:3647,:));
eightyYears = mean(PercentageSCAge(3648:4168,:));
ninetyYears = mean(PercentageSCAge(4169:4689,:));
hundredYears = mean(PercentageSCAge(4690:5210,:));


meanAgeBrackets = [twentyYears; thrityYears;...
    fortyYears; fiftyYears; sixtyYears; seventyYears ;...
    eightyYears]';

meanAgeBrackets2 = meanAgeBrackets / 100;
```

```
%% Graph the results

figure('position'  , [200 400 1000 700])
subplot(2,1,1)
bar(meanAgeBrackets(2:end,:))
set(gca, ...
    'Box'         , 'off'      ,...
    'TickDir'     , 'out'      ,...
    'TickLength'  , [.01 .01] ,...
    'XColor'      , 'k'        ,...
    'YColor'      , 'k'        ,...
    'XTick'       , 0:1:5      ,...
    'LineWidth'   , 2          ,...
    'FontSize'    , 8          ,...
    'XTick'       , 1:5        ,...
    'XTickLabel',{'20','40','60','80','100'});
xlabel('Percentage  COX deficiency  of individual  crypts',...
    'FontWeight','Bold','FontSize',12);
ylabel('Percentage  of total crypts',...
    'FontWeight','Bold','FontSize',12);
legend('10-20years','20-30years','30-40years',...
    '40-50years','50-60years','60-70years','70-80years',...
    'Location','NorthEastOutside');

set(gca, ...
    'Box'         , 'off'      ,...
    'TickDir'     , 'out'      ,...
    'TickLength'  , [.01 .01] ,...
    'XColor'      , 'k'        ,...
    'YColor'      , 'k'        ,...
    'XTick'       , 0:1:5      ,...
    'LineWidth'   , 2          ,...
    'FontSize'    , 8          ,...
    'XTick'       , 1:5        ,...
    'XTickLabel',{'20','40','60','80','100'});
xlabel('Percentage  COX deficiency  of individual  crypts',...
    'FontWeight','Bold','FontSize',12);
ylabel('Percentage  of total crypts',...
    'FontWeight','Bold','FontSize',12);
legend('10-20years','20-30years','30-40years',...
    '40-50years','50-60years','60-70years','70-80years',...
    'Location','NorthEastOutside');
title(gg.finalFilename,  'FontSize',  16, 'FontWeight',  'Bold');

subplot(2,1,2)
bar(gg.MultipleMutRecordResult(1:5))
axis([0,  6, 0, 100]);
set(gca, ...
    'YTick'          ,            0:10:100  ,...
    'TickDir'        ,          'out'  ,...
    'TickLength'  ,          [.01 .01] ,...
    'Box'              ,          'off'  ,...
    'LineWidth'     ,           2);

xlabel('Number of mutations within individual stem cells',...
    'FontWeight','Bold','FontSize',12);
ylabel('Percentage of cells with mutations',...
    'FontWeight','Bold','FontSize',12);
```

1.4.3.5. *Relaxed replication transition matrices generation*

```
function [ ReplicativeProbabilities ] = RepProbScript( mtDNATot )

% Replicative Probability Distribution
% Calculates the probability distribution of any number of mutated mtDNA
% molecules undergoing replication before division.

r = mtDNATot + 1;
c = 1:mtDNATot;
row = zeros(1,mtDNATot + 1);
row(1) = 1;

for ii = 1:mtDNATot
  row(ii+1)=row(ii)*(r-c(ii))/c(ii);
end

ReplicativeProbabilities = zeros(mtDNATot-1,mtDNATot+1);

for ii = 1:mtDNATot-1
    for jj = 1:mtDNATot+1
        ReplicativeProbabilities(ii,jj) = (ii/mtDNATot)^(jj-1)...
            *((mtDNATot-ii)/mtDNATot)^(mtDNATot+1-jj)*row(jj);
    end
end

ReplicativeProbabilities = [ReplicativeProbabilities; zeros(1,mtDNATot + 1);];
ReplicativeProbabilities(mtDNATot,mtDNATot+1)=1;

end
```

1.4.3.6. *Random segregation transition matrices generation*

```
function [ DivisionProbabilities ] = DivProbScript( mtDNATot )

% Dividing Probability Distribution
% Calculates the probability distribution of any number of mutated mtDNA
% molecules undergoing division after replication

r = mtDNATot+1;
c = 1:mtDNATot;
row = zeros(1,mtDNATot+1);
row(1) = 1;
for ii = 1:mtDNATot
  row(ii+1)=row(ii)*(r-c(ii))/c(ii);
end

% the calculation is only done for mutation between 1 and 2*mtDNA-1, if
% there are no mutations present or all mtDNA is mutated, the solution
% is obvious

DivisionProbabilities = zeros(mtDNATot*2-1,mtDNATot+1);

% the denominator is always the same:
denominator = 1/prod(mtDNATot+1:mtDNATot*2);

for ii = 1:mtDNATot*2-1
    for jj = 1:mtDNATot+1
        if jj - 1 <= ii
            nonMutNumerator = prod((mtDNATot+1)-ii+jj-1:((mtDNATot*2)-ii));
            MutNumerator = prod(ii-jj+2:ii);
            DivisionProbabilities(ii,jj) = nonMutNumerator*...
                MutNumerator*row(jj)*denominator;
            % you can't have more mutated mtDNA in the daughter cells than
            % you have in the mother cell
        else
           DivisionProbabilities(ii,jj) = 0;
        end
    end
end

DivisionProbabilities = [DivisionProbabilities; zeros(1,mtDNATot + 1);];
DivisionProbabilities(mtDNATot*2,mtDNATot+1)=1;

end
```

1.4.3.7. *Random segregation with advantage transition matrices generation*

```
% division_montecarlo.m

% Gives you the probabilities that a certain number of mutated mtDNA
% molecules segregate into one of the daughter cells (depending on the
% total number of mtDNA molecules in the cell, and the total number of
% mutated mtDNA molecules in the cell, and the advantage that is given for
% segregation of mutated mtDNA molecules

% INPUTS:
% numMTDNA    - number of mtDNA molecules in the daughter cell
% adv         - a number between 0 and infinity (if adv == 1, it's neutral)
%               that describes how many times more likely a mutated mtDNA
%               molecules is to get segregated
% montecarlo  - number of samples taken

% OUTPUT:
% divisionTransitionMatrix  - the transition matrix with segregation
%                             advantage

function [divisionTransitionMatrix] = division_montecarlo(numMTDNA,...
   adv, montecarlo)
tic
% initiate the transition matrix, according to number of mtDNA molecules
divisionTransitionMatrix = zeros(numMTDNA*2, numMTDNA+1);

% the first row is always the same 0.5, 0.5 all zeros
% the second to last and the last rows are also always the same
divisionTransitionMatrix(1,:)=[0.5, 0.5, zeros(1,numMTDNA-1)];
divisionTransitionMatrix(end,:)=[zeros(1,numMTDNA),1];
divisionTransitionMatrix(end - 1,:)=[zeros(1,numMTDNA-1), 0.5 , 0.5];

% Do the calculation of probability for each number of mutated mtDNA
% molecules in the mother cell [2 to 2*numMTDNA-2] - outer for loop
% for each number of mutated mtDNA molecules in the daughter cell [0 to
% numMTDNA] - inner loop

for ii = 2:numMTDNA*2-2
   % the matrix is always symmetric so when the first half is calculated,
   % the second half is a mirror images p - probability that a healthy
   % mtDNA molecules is picked for segregation into a daughter cell (as
   % the first cell) p*adv - probability that a mutated mtDNA molecules
   % gets segregated (as the first cell)

   p= 1/((numMTDNA*2-ii)+ii*adv);

   % initialize row for transition matrix (counts of how many mutated
   % mtDNA are chosen)
```

```
    countMutations = zeros(1,numMTDNA+1);

    % monte carlo samplings ,100 for now
    for jj = 1:montecarlo
        probDistribution = [p*ones(1,2*numMTDNA-ii),p*adv*ones(1,ii)];
        mutatedVector = [zeros(1,2*numMTDNA-ii),ones(1,ii)];

        % initialize count of how many mutated mtDNA molecules are chosen
        countMut = 0;

        % take exactly half of the mtDNA molecules from the mother cell
        for pp = 1:numMTDNA
            % take a single mtDNA from the cell
            [n,x] = histc(rand(1,1),[0;cumsum(probDistribution(:))...
                /sum(probDistribution)]);
            % if a mutated was taken the probDistribution and
            % mutatedVector change
            if mutatedVector(x)>0.5
                countMut = countMut +1;
            end
            probDistribution(x) = [];
            mutatedVector(x) = [];
        end
        countMutations(countMut+1) = countMutations(countMut+1) +1;
    end
    divisionTransitionMatrix(ii,:)=countMutations/montecarlo;
end
toc
```

1.4.4. **Stem cell relationship to cells observed in transverse crypts**

1.4.4.1. *Stem cell lineage tracing simulation and distribution generation*

```
%% Stem cell lineage tracing relationship

% Stochastically calculating the relationship between the number of mutated
% stem cells at the base of the crypt and the percentage COX deficiency of
% a crypt when viewed in transverse cross sections.

rng('shuffle')

for ll = 4 : 16 % Generate probability tables for these number of stem cells

% Number of stem cells at the base of the crypt

X = ll;

% Number of stem cells at the base of the crypt

Y = X;

% Number of steps to reach the point of observation

steps = 4;

% Activate random number generator

RandomNumbers = rand(1000000000,1);
rngcount = 1;

% Number of Runs

numRuns = 100000;

% Record Final Results

FinalResults = zeros(numRuns,X+1);

% Iteration for each number of COX deficient stem cells at the base of the
% crypt

for yy = 1 : numRuns

  for ii = 1 : X-1 % For each number of mutations

    % For each cell replication the probability that a mutated cell is
    % replicated is dependent on the number of mutated cells and the number
    % of cells that are present at level it is at. The number of cells
    % present increases every time a cell is replicated therefore the
```

```
        % probability  denominator  increases  by one each time.

        cellMut  = ii;

        for tt = 1 : steps % For the number  of steps

            Y = X*(2^(tt-1));

            for nn = 1 : Y % For the  number  of cells  to be replicated  i.e 5 to 10

                RepProb = cellMut  / (Y + (nn  - 1));

                if RandomNumbers(rngcount,1)  < RepProb
                    cellMut  = cellMut  + 1;
                end
                rngcount  = rngcount  + 1;
            end
            Result(tt,ii)  = cellMut;
        end
    end

    % refine  results

    a = zeros(1,steps)';
    b = 0:1:X;

    % Generate  the maximum  number  of mutated  cells  at each level

    c = zeros(steps,1);

    for uu = 1 : steps
        c(uu)  = X*2^uu;
    end

    Result  = [a Result  c];
    Result  = [b; Result];

    FinalResults(yy,1:end)  = Result(steps+1,1:end);

    clearvars  Result

end

% Save the  Final  Results  with  unique  name  i.e the  number  of stem cells  that
% the  simulation  is  calculating  probability  distibutions  for

saveNameSC  = num2str(ll);

filename  = ['FinalResults',saveNameSC];
```

```
save(filename,'FinalResults');

display(filename)

clearvars  -except ll
end
```

1.4.4.2. *Simulated distribution conversion from percentage observed to stem cell number*

```
%% Convert the generated data into the format that is required
% How % observed relates to SC number

% Parameters required for script continuation

numRuns = 100000;

% Which file needs to be brought into the script
% FinalResultsSC#

uiopen('load')

% Number of stem cells

% Convert the matrix into percentage terms

FinalResults = (FinalResults./FinalResults(1,end))*100;

% Attain the values to make the original histogram

for ii = 2 : numel(FinalResults(1,1:end)) - 1

   [x1(ii-1,:),c1(ii-1,:)] = hist(FinalResults(:,ii));

end

% Make the number of elements within the histogram a percentage of the
% total number

for ii = 2 : numel(FinalResults(1,1:end)) - 1

   x1(ii-1,:) = (x1(ii-1,:) ./ numRuns) * 100;

end

% Modify percentage frequency distributions to include the initial and end
% zero value.

a(1:numel(FinalResults(1,1:end))-2,1) = 0;
b(1:numel(FinalResults(1,1:end))-2,1) = 100;

x1 = [a x1 a]; % nelements

c1 = [a c1 b]; % centers

% Continuous frequency function to be applied to the frequency distribution
% using pchip.
```

```
for ii = 2 : numel(FinalResults(1,1:end)) - 1

    y1(ii-1,:) = pchip(c1(ii-1,:),x1(ii-1,:),0:1:100);

end

% Smooth the functions

for ii = 2 : numel(FinalResults(1,1:end)) - 1

    yy1(:,ii-1) = smooth(y1(ii-1,:))'; % invert for plot function below

end

%% Nomalisation for each stem cell

% Make the area under the curve equal to 1 so that it is converted into a
% probability distribution

sumYy1 = sum(yy1);

for ii = 2 : numel(FinalResults(1,1:end)) - 1

    yy1(:,ii-1) = yy1(:,ii-1)/sumYy1(ii-1);

end

% Remove rows that are not required anymore

yy1(1,:) = [];
yy1(100,:) = [];

%% Normalisation for individual percentage probability

% Sum values across the percentage probability

for ii = 1 : 99

    sumYy2(ii) = sum(yy1(ii,:));

end

for ii = 1 : 99

    yy1(ii,:) = yy1(ii,:)/sumYy2(ii);

end

%% Save the final probability table for each number of stem cells
```

```
yyAllFinal = yy1;
% Save yyAllFinal under the name DistributionSC#
```

1.4.4.3. *Simulated distribution conversion from stem cell number to percentage observed*

```
%% Convert the generated data into the format that is required
% How SC number relates to % observed

% Parameters required for script continuation

numRuns = 100000;

% Which file needs to be brought into the script 'FinalResults4-16'

uiopen('load')

% Number of stem cells

% Convert the matrix into percentage terms

FinalResults = (FinalResults./FinalResults(1,end))*100;

% Attain the values to make the original histogram

for ii = 2 : numel(FinalResults(1,1:end)) - 1

   [x1(ii-1,:),c1(ii-1,:)] = hist(FinalResults(:,ii));

end

% Make the number of elements within the histogram a percentage of the
% total number

for ii = 2 : numel(FinalResults(1,1:end)) - 1

   x1(ii-1,:) = (x1(ii-1,:) ./ numRuns) * 100;

end

% Modify percentage frequency distributions to include the initial and end
% zero value.

a(1:numel(FinalResults(1,1:end))-2,1) = 0;
b(1:numel(FinalResults(1,1:end))-2,1) = 100;

x1 = [a x1 a]; % nelements

c1 = [a c1 b]; % centers

% Continuous frequency function to be applied to the frequency distribution
% using pchip.
```

```
for ii = 2 : numel(FinalResults(1,1:end)) - 1

    y1(ii-1,:) = pchip(c1(ii-1,:),x1(ii-1,:),0:1:100);

end

% Smooth the functions

for ii = 2 : numel(FinalResults(1,1:end)) - 1

    yy1(:,ii-1) = smooth(y1(ii-1,:))'; % invert for plot function below

end

%% Nomalisation for each stem cell

% Make the area under the curve equal to 1 so that it is converted into a
% probability distribution

sumYy1 = sum(yy1);

for ii = 2 : numel(FinalResults(1,1:end)) - 1

    yy1(:,ii-1) = yy1(:,ii-1)/sumYy1(ii-1);

end

% Remove rows that are not required anymore

yy1(1,:) = [];
yy1(100,:) = [];

%% Save the final probability table for each number of stem cells

yy1 = yy1';
yyAllFinal = yy1;

% Save yyAllFinal under the name DistributionPerSC#
```

1.4.4.4. *Convert biological data into specified stem cell fractions*

```
%% biologicalDataSCs.m
% This script takes all the biological data and then splits it up into the
% number of stem cells that it represents based on specific fraction
% boundaries

% Please enter number of stem cells between 4 and 16...

clc

clear all

X = 9;

FinalResultsMatrixIter = zeros(X+1,7);

FinalSEMIter = zeros(X+1,7);

FinalSDIter = zeros(X+1,7);

%% load the excel data table into MATLAB

expDataWhole = xlsread('allData2.xlsx');

% Take out row numbers from the data table

expDataWhole(:,1) = [];

% Take out the ages from the data table and assign to a new variable

dataRowAges = expDataWhole(:,1)'; expDataWhole(:,1) = [];

%% For each age bracket determine the distributional binning number

dimensions = size(expDataWhole);

Counts = zeros(dimensions(1),X+1);

for ii = 1 : dimensions(1)

    % Fully normal or partial crypts

    Counts(ii,1) = numel(find(expDataWhole(ii,:) == 0));
    Counts(ii,X+1) = numel(find(expDataWhole(ii,:) == 1));

    for jj = 1 : X-1

        if jj == 1
```

```
            Counts(ii,jj+1) = numel(find(expDataWhole(ii,:) >0 & expDataWhole(ii,:) <= jj/(X-
1)));

        end

        if jj > 1 && jj < X-1

            Counts(ii,jj+1) = numel(find(expDataWhole(ii,:) > (jj-1)/(X-1) & expDataWhole(ii,:)
<= (jj)/(X-1)));

        end

        if jj == X-1

            Counts(ii,jj+1) = numel(find(expDataWhole(ii,:) > (jj-1)/(X-1) & expDataWhole(ii,:)
< 1));

        end

    end

end

% Need to split data up into age brackets use the data row ages

Bracket20 = find(dataRowAges > 10 & dataRowAges <= 20);
Bracket30 = find(dataRowAges > 20 & dataRowAges <= 30);
Bracket40 = find(dataRowAges > 30 & dataRowAges <= 40);
Bracket50 = find(dataRowAges > 40 & dataRowAges <= 50);
Bracket60 = find(dataRowAges > 50 & dataRowAges <= 60);
Bracket70 = find(dataRowAges > 60 & dataRowAges <= 70);
Bracket80 = find(dataRowAges > 70 & dataRowAges <= 80);

% Use the row numbers for block seperation

Block20 = Counts(Bracket20,:);
Block30 = Counts(Bracket30,:);
Block40 = Counts(Bracket40,:);
Block50 = Counts(Bracket50,:);
Block60 = Counts(Bracket60,:);
Block70 = Counts(Bracket70,:);
Block80 = Counts(Bracket80,:);

%% Determine the standard deviation and standard error of the mean

% Convert to percentage terms

% Number of samples

dimension20 = size(Block20);
```

```
dimension30  = size(Block30);
dimension40  = size(Block40);
dimension50  = size(Block50);
dimension60  = size(Block60);
dimension70  = size(Block70);
dimension80  = size(Block80);

% Mean All

for ii = 1 : dimension20(1)
   a = sum(Block20(ii,:));
   Block20Per(ii,:)  = Block20(ii,:)  / a*100;
end

for ii = 1 : dimension30(1)
   a = sum(Block30(ii,:));
   Block30Per(ii,:)  = Block30(ii,:)  / a*100;
end

for ii = 1 : dimension40(1)
   a = sum(Block40(ii,:));
   Block40Per(ii,:)  = Block40(ii,:)  / a*100;
end

for ii = 1 : dimension50(1)
   a = sum(Block50(ii,:));
   Block50Per(ii,:)  = Block50(ii,:)  / a*100;
end

for ii = 1 : dimension60(1)
   a = sum(Block60(ii,:));
   Block60Per(ii,:)  = Block60(ii,:)  / a*100;
end

for ii = 1 : dimension70(1)
   a = sum(Block70(ii,:));
   Block70Per(ii,:)  = Block70(ii,:)  / a*100;
end

for ii = 1 : dimension80(1)
   a = sum(Block80(ii,:));
   Block80Per(ii,:)  = Block80(ii,:)  / a*100;
end

% Determine  mean

Block20Mean  = mean(Block20Per);
Block30Mean  = mean(Block30Per);
Block40Mean  = mean(Block40Per);
Block50Mean  = mean(Block50Per);
```

```
Block60Mean = mean(Block60Per);
Block70Mean = mean(Block70Per);
Block80Mean = mean(Block80Per);

MeanAll = [Block20Mean; Block30Mean; Block40Mean;...
   Block50Mean; Block60Mean; Block70Mean; Block80Mean];

% Determine standard deviation

Block20SD = std(Block20Per);
Block30SD = std(Block30Per);
Block40SD = std(Block40Per);
Block50SD = std(Block50Per);
Block60SD = std(Block60Per);
Block70SD = std(Block70Per);
Block80SD = std(Block80Per);

SDAll = [Block20SD; Block30SD; Block40SD;...
   Block50SD; Block60SD; Block70SD; Block80SD];

% Determine standard error of the mean

% Number of samples

Block20SEM = Block20SD / sqrt(dimension20(1));
Block30SEM = Block30SD / sqrt(dimension30(1));
Block40SEM = Block40SD / sqrt(dimension40(1));
Block50SEM = Block50SD / sqrt(dimension50(1));
Block60SEM = Block60SD / sqrt(dimension60(1));
Block70SEM = Block70SD / sqrt(dimension70(1));
Block80SEM = Block80SD / sqrt(dimension80(1));

SEMAll = [Block20SEM; Block30SEM; Block40SEM;...
   Block50SEM; Block60SEM; Block70SEM; Block80SEM];


% Correct format of matrices

MeanAll = MeanAll';
SDAll = SDAll';
SEMAll = SEMAll';


%% Graph the main result

% We know what X is

% Set up 2 string arrays that have 1 to 16 generated

str = (0:1:X);
```

```
str1 = num2cell(str);


figHandle  = figure(1);
set(gcf,'color','w');
set(gcf,'units','normalized','outerposition',[0  0 1 1]);

subplot(2,1,1)

barweb(MeanAll,SEMAll,[],  [], [], [], [], [], [], [], [], []);
set(gca, ...
        'Box'          , 'off'      ,...
        'TickDir'      , 'out'      ,...
        'TickLength'   , [.01 .01] ,...
        'XColor'       , 'k'       ,...
        'YColor'       , 'k'       ,...
        'XTick'        , 1:1:X+1    ,...
        'LineWidth'    , 2         ,...
        'FontSize'     , 10        ,...
        'XLim'         , [0 X+2]     ,...
        'YLim'         , [0 100]   ,...
        'XTickLabel',str1);
        xlabel('Number  of stem cells  COX deficient',...
            'FontSize',15);
        ylabel('Percentage  of age bracket',...
            'FontSize',15);
        legend('10-20  years','20-30 years','30-40 years',...
            '40-50 years','50-60 years','60-70 years',...
            '70-80 years','Location','NorthEastOutside');
        title('Human  colon  respiratory  deficiency  data - 16SCs',...
            'FontWeight','Bold','FontSize',20)



subplot(2,1,2)

barweb(MeanAll(2:end,:),SEMAll(2:end,:),[],  [], [], [], [], [], [], [], [], []);
set(gca, ...
        'Box'          , 'off'      ,...
        'TickDir'      , 'out'      ,...
        'TickLength'   , [.01 .01] ,...
        'XColor'       , 'k'       ,...
        'YColor'       , 'k'       ,...
        'XTick'        , 0:1:X      ,...
        'LineWidth'    , 2         ,...
        'FontSize'     , 10        ,...
        'XLim'         , [0 X+1]     ,...
        'YLim'         , [0 11]   ,...
        'XTickLabel',str1);
        xlabel('Number  of stem cells  COX deficient',...
            'FontSize',15);
        ylabel('Percentage  of age bracket',...
            'FontSize',15);
        legend('10-20  years','20-30 years','30-40 years',...
            '40-50 years','50-60 years','60-70 years',...
            '70-80 years','Location','NorthEastOutside');
```

1.4.4.5. *Convert biological data into number of stem cells using generated distributions*

```
%% AllExperimentalDataManipulation.m
% This script will take all the experimentally obtained data for use
% when using a distribution of probabilities for assigning how many stem
% cells those partially deficient percentages relate to how many stem cells
% are contained at the base of the crypt.
% v2 - This version calculates the standard error and standard deviation
% from each patients sample.

% What type of binning is to be performed, 5SC, 8SC or 16SC

% Please enter number of stem cells between 4 and 16...

for X = 4:16;

strName = ['Distribution', num2str(X)];

load(strName);

FinalResultsMatrixIter = zeros(X+1,7);

FinalSEMIter = zeros(X+1,7);

FinalSDIter = zeros(X+1,7);

% Number of runs

numRuns = 20;

IterSample = numRuns/20:numRuns/20:numRuns;

IterSampleRecord = zeros(1,numRuns/(numRuns/20));

Sample = 1;

% Begin for loop that will iteratively sum the final results matrix

for cc = 1 : numRuns

%% load the excel data table into MATLAB

expDataWhole = xlsread('allData2.xlsx');

% Take out row numbers from the data table

expDataWhole(:,1) = [];

% Take out the ages from the data table and assign to a new variable
```

```
dataRowAges = expDataWhole(:,1)'; expDataWhole(:,1) = [];

% Split data up into all age brackets
% 10-20yr
Bracket20 = find(dataRowAges > 10 & dataRowAges <= 20);
Bracket20Data = expDataWhole(Bracket20(1):Bracket20(end),:);

% 20-30yr
Bracket30 = find(dataRowAges > 20 & dataRowAges <= 30);
Bracket30Data = expDataWhole(Bracket30(1):Bracket30(end),:);

% 30-40yr
Bracket40 = find(dataRowAges > 30 & dataRowAges <= 40);
Bracket40Data = expDataWhole(Bracket40(1):Bracket40(end),:);


% 40-50yr
Bracket50 = find(dataRowAges > 40 & dataRowAges <= 50);
Bracket50Data = expDataWhole(Bracket50(1):Bracket50(end),:);

% 50-60yr
Bracket60 = find(dataRowAges > 50 & dataRowAges <= 60);
Bracket60Data = expDataWhole(Bracket60(1):Bracket60(end),:);

% 60-70yr
Bracket70 = find(dataRowAges > 60 & dataRowAges <= 70);
Bracket70Data = expDataWhole(Bracket70(1):Bracket70(end),:);

% 70-80yr
Bracket80 = find(dataRowAges > 70 & dataRowAges <= 80);
Bracket80Data = expDataWhole(Bracket80(1):Bracket80(end),:);

%% For each age bracket determine the distributional binning number

% 10-20yr

dimensions20 = size(Bracket20Data);

Bracket20SCNo = zeros(dimensions20(1),dimensions20(2));

for ii = 1 : dimensions20(1)
    for jj = 1 : dimensions20(2)
        if Bracket20Data(ii,jj) == 1;
            Bracket20SCNo(ii,jj) = X;
        elseif Bracket20Data(ii,jj) == 0;
            Bracket20SCNo(ii,jj) = 0;
        elseif isnan(Bracket20Data(ii,jj));
            Bracket20SCNo(ii,jj) = Inf;
        else Bracket20SCNo(ii,jj) = DiscSampVec2((1:X-1),...
            yyAllFinal(ceil(Bracket20Data(ii,jj)*100),:),1);
        end
    end
end

% 20-30yr

dimensions30  = size(Bracket30Data);

Bracket30SCNo = zeros(dimensions30(1),dimensions30(2));

for ii = 1 : dimensions30(1)
    for jj = 1 : dimensions30(2)
        if Bracket30Data(ii,jj)  == 1;
            Bracket30SCNo(ii,jj)  = X;
        elseif  Bracket30Data(ii,jj)  == 0;
            Bracket30SCNo(ii,jj)  = 0;
        elseif  isnan(Bracket30Data(ii,jj));
            Bracket30SCNo(ii,jj)  = Inf;
        else Bracket30SCNo(ii,jj)  = DiscSampVec2((1:X-1),...
            yyAllFinal(ceil(Bracket30Data(ii,jj)*100),:),1);
        end
    end
end

% 30-40yr

dimensions40  = size(Bracket40Data);

Bracket40SCNo = zeros(dimensions40(1),dimensions40(2));

for ii = 1 : dimensions40(1)
    for jj = 1 : dimensions40(2)
        if Bracket40Data(ii,jj)  == 1;
            Bracket40SCNo(ii,jj)  = X;
        elseif  Bracket40Data(ii,jj)  == 0;
            Bracket40SCNo(ii,jj)  = 0;
        elseif  isnan(Bracket40Data(ii,jj));
            Bracket40SCNo(ii,jj)  = Inf;
        else Bracket40SCNo(ii,jj)  = DiscSampVec2((1:X-1),...
            yyAllFinal(ceil(Bracket40Data(ii,jj)*100),:),1);
        end
    end
end

% 40-50yr

dimensions50  = size(Bracket50Data);

Bracket50SCNo = zeros(dimensions50(1),dimensions50(2));
```

```
for ii = 1 : dimensions50(1)
    for jj = 1 : dimensions50(2)
        if Bracket50Data(ii,jj) == 1;
            Bracket50SCNo(ii,jj) = X;
        elseif Bracket50Data(ii,jj) == 0;
            Bracket50SCNo(ii,jj) = 0;
        elseif isnan(Bracket50Data(ii,jj));
            Bracket50SCNo(ii,jj) = Inf;
        else Bracket50SCNo(ii,jj) = DiscSampVec2((1:X-1),...
              yyAllFinal(ceil(Bracket50Data(ii,jj)*100),:),1);
        end
    end
end

% 50-60yr

dimensions60 = size(Bracket60Data);

Bracket60SCNo = zeros(dimensions60(1),dimensions60(2));

for ii = 1 : dimensions60(1)
    for jj = 1 : dimensions60(2)
        if Bracket60Data(ii,jj) == 1;
            Bracket60SCNo(ii,jj) = X;
        elseif Bracket60Data(ii,jj) == 0;
            Bracket60SCNo(ii,jj) = 0;
        elseif isnan(Bracket60Data(ii,jj));
            Bracket60SCNo(ii,jj) = Inf;
        else Bracket60SCNo(ii,jj) = DiscSampVec2((1:X-1),...
              yyAllFinal(ceil(Bracket60Data(ii,jj)*100),:),1);
        end
    end
end

% 60-70yr

dimensions70 = size(Bracket70Data);

Bracket70SCNo = zeros(dimensions70(1),dimensions70(2));

for ii = 1 : dimensions70(1)
    for jj = 1 : dimensions70(2)
        if Bracket70Data(ii,jj) == 1;
            Bracket70SCNo(ii,jj) = X;
        elseif Bracket70Data(ii,jj) == 0;
            Bracket70SCNo(ii,jj) = 0;
        elseif isnan(Bracket70Data(ii,jj));
            Bracket70SCNo(ii,jj) = Inf;
        else Bracket70SCNo(ii,jj) = DiscSampVec2((1:X-1),...
            yyAllFinal(ceil(Bracket70Data(ii,jj)*100),:),1);
        end
    end
end

% 70-80yr

dimensions80  = size(Bracket80Data);

Bracket80SCNo = zeros(dimensions80(1),dimensions80(2));

for ii = 1 : dimensions80(1)
    for jj = 1 : dimensions80(2)
        if Bracket80Data(ii,jj)  == 1;
            Bracket80SCNo(ii,jj)  = X;
        elseif  Bracket80Data(ii,jj)  == 0;
            Bracket80SCNo(ii,jj)  = 0;
        elseif  isnan(Bracket80Data(ii,jj));
            Bracket80SCNo(ii,jj)  = Inf;
        else Bracket80SCNo(ii,jj)  = DiscSampVec2((1:X-1),...
            yyAllFinal(ceil(Bracket80Data(ii,jj)*100),:),1);
        end
    end
end

%% Age Bracket Count Up

% All  Age Groups  Total  for  each number  of stem cells

for pp = 0 : X

Year20Result(1,pp+1)  = numel(find(Bracket20SCNo  == pp));
Year30Result(1,pp+1)  = numel(find(Bracket30SCNo  == pp));
Year40Result(1,pp+1)  = numel(find(Bracket40SCNo  == pp));
Year50Result(1,pp+1)  = numel(find(Bracket50SCNo  == pp));
Year60Result(1,pp+1)  = numel(find(Bracket60SCNo  == pp));
Year70Result(1,pp+1)  = numel(find(Bracket70SCNo  == pp));
Year80Result(1,pp+1)  = numel(find(Bracket80SCNo  == pp));

end

ResultsMatrix  = [Year20Result;Year30Result;...
    Year40Result;Year50Result;Year60Result;...
    Year70Result;Year80Result]';

FinalResultsMatrix  = zeros(X+1,7);

for kk = 1 : 7
    for ll = 1 : X + 1
        a = sum(ResultsMatrix(:,kk));
        FinalResultsMatrix(ll,kk)  = (ResultsMatrix(ll,kk)  / a)*100;
    end
end

%% Incorporate standard error and standard error of the mean

% % Modify this so that it does it for each patient sample

Year20ResultIndiv  = zeros(X+1,dimensions20(1));

for hh = 1 : dimensions20(1)
    for pp = 0 : X
        Year20ResultIndiv(pp+1,hh)  = numel(find(Bracket20SCNo(hh,:)  == pp));
    end
end

Year30ResultIndiv  = zeros(X+1,dimensions30(1));

for hh = 1 : dimensions30(1)
    for pp = 0 : X
        Year30ResultIndiv(pp+1,hh)  = numel(find(Bracket30SCNo(hh,:)  == pp));
    end
end

Year40ResultIndiv  = zeros(X+1,dimensions40(1));

for hh = 1 : dimensions40(1)
    for pp = 0 : X
        Year40ResultIndiv(pp+1,hh)  = numel(find(Bracket40SCNo(hh,:)  == pp));
    end
end

Year50ResultIndiv  = zeros(X+1,dimensions50(1));

for hh = 1 : dimensions50(1)
    for pp = 0 : X
        Year50ResultIndiv(pp+1,hh)  = numel(find(Bracket50SCNo(hh,:)  == pp));
    end
end

Year60ResultIndiv  = zeros(X+1,dimensions60(1));

for hh = 1 : dimensions60(1)
    for pp = 0 : X
        Year60ResultIndiv(pp+1,hh)  = numel(find(Bracket60SCNo(hh,:)  == pp));
    end
end

Year70ResultIndiv  = zeros(X+1,dimensions70(1));
```

```
for hh = 1 : dimensions70(1)
   for pp = 0 : X
      Year70ResultIndiv(pp+1,hh)  = numel(find(Bracket70SCNo(hh,:)  == pp));
   end
end

Year80ResultIndiv  = zeros(X+1,dimensions80(1));

for hh = 1 : dimensions80(1)
   for pp = 0 : X
      Year80ResultIndiv(pp+1,hh)  = numel(find(Bracket80SCNo(hh,:)  == pp));
   end
end

% Convert into percentage terms before doing the standard error

Year20ResultIndivPerc  = zeros(X+1,dimensions20(1));

for ss = 1 : dimensions20(1)
   a = sum(Year20ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year20ResultIndivPerc(dd,ss)  = (Year20ResultIndiv(dd,ss)  / a) * 100;
   end
end

Year30ResultIndivPerc  = zeros(X+1,dimensions30(1));

for ss = 1 : dimensions30(1)
   a = sum(Year30ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year30ResultIndivPerc(dd,ss)  = (Year30ResultIndiv(dd,ss)  / a) * 100;
   end
end

Year40ResultIndivPerc  = zeros(X+1,dimensions40(1));

for ss = 1 : dimensions40(1)
   a = sum(Year40ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year40ResultIndivPerc(dd,ss)  = (Year40ResultIndiv(dd,ss)  / a) * 100;
   end
end

Year50ResultIndivPerc  = zeros(X+1,dimensions50(1));

for ss = 1 : dimensions50(1)
   a = sum(Year50ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year50ResultIndivPerc(dd,ss)  = (Year50ResultIndiv(dd,ss)  / a) * 100;
   end
```

```
end

Year60ResultIndivPerc  = zeros(X+1,dimensions60(1));

for ss = 1 : dimensions60(1)
   a = sum(Year60ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year60ResultIndivPerc(dd,ss)  = (Year60ResultIndiv(dd,ss)  / a) * 100;
   end
end

Year70ResultIndivPerc  = zeros(X+1,dimensions70(1));

for ss = 1 : dimensions70(1)
   a = sum(Year70ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year70ResultIndivPerc(dd,ss)  = (Year70ResultIndiv(dd,ss)  / a) * 100;
   end
end

Year80ResultIndivPerc  = zeros(X+1,dimensions80(1));

for ss = 1 : dimensions80(1)
   a = sum(Year80ResultIndiv(:,ss));
   for dd = 1 : X + 1
      Year80ResultIndivPerc(dd,ss)  = (Year80ResultIndiv(dd,ss)  / a) * 100;
   end
end

% Calculate  the standard  error and standard  deviation  for each of the age
% bracketed  data

for ff = 1 : X+1

Year20SEMSTD(ff,1)  = std(Year20ResultIndivPerc(ff,:));
Year20SEMSTD(ff,2)  = std(Year20ResultIndivPerc(ff,:))  / sqrt(dimensions20(1));

Year30SEMSTD(ff,1)  = std(Year30ResultIndivPerc(ff,:));
Year30SEMSTD(ff,2)  = std(Year30ResultIndivPerc(ff,:))  / sqrt(dimensions30(1));

Year40SEMSTD(ff,1)  = std(Year40ResultIndivPerc(ff,:));
Year40SEMSTD(ff,2)  = std(Year40ResultIndivPerc(ff,:))  / sqrt(dimensions40(1));

Year50SEMSTD(ff,1)  = std(Year50ResultIndivPerc(ff,:));
Year50SEMSTD(ff,2)  = std(Year50ResultIndivPerc(ff,:))  / sqrt(dimensions50(1));

Year60SEMSTD(ff,1)  = std(Year60ResultIndivPerc(ff,:));
Year60SEMSTD(ff,2)  = std(Year60ResultIndivPerc(ff,:))  / sqrt(dimensions60(1));

Year70SEMSTD(ff,1)  = std(Year70ResultIndivPerc(ff,:));
```

```
Year70SEMSTD(ff,2) = std(Year70ResultIndivPerc(ff,:)) / sqrt(dimensions70(1));

Year80SEMSTD(ff,1) = std(Year80ResultIndivPerc(ff,:));
Year80SEMSTD(ff,2) = std(Year80ResultIndivPerc(ff,:)) / sqrt(dimensions80(1));

end

% Create final error matrices

FinalSEM = [Year20SEMSTD(:,2), Year30SEMSTD(:,2), Year40SEMSTD(:,2),...
   Year50SEMSTD(:,2), Year60SEMSTD(:,2), Year70SEMSTD(:,2),...
   Year80SEMSTD(:,2)];

FinalSD = [Year20SEMSTD(:,1), Year30SEMSTD(:,1), Year40SEMSTD(:,1),...
   Year50SEMSTD(:,1), Year60SEMSTD(:,1), Year70SEMSTD(:,1),...
   Year80SEMSTD(:,1)];


%% Iterative addition of important matrices

% Add up FinalResultsMatrix

FinalResultsMatrixIter = FinalResultsMatrixIter + FinalResultsMatrix;

% Add up FinalSEM

FinalSEMIter = FinalSEMIter + FinalSEM;

% Add up FinalSD

FinalSDIter = FinalSDIter + FinalSD;

% IterSampleAnalysis

if cc == IterSample(1)
   IterSampleRecord(Sample) = sum(sum(FinalResultsMatrixIter));
   IterSample(1) = [];
   Sample = Sample + 1;
end

clearvars -except FinalResultsMatrixIter FinalSEMIter FinalSDIter cc yyAllFinal X
numRuns IterSample IterSampleRecord Sample

FinalResultsMatrix = FinalResultsMatrixIter ./ cc;

FinalSEM = FinalSEMIter ./ cc;

FinalSD = FinalSDIter ./ cc;

end
```

```
%% Save the results

saveNameSD  = ['SD',num2str(X)];

saveNameSEM  = ['SEM',num2str(X)];

saveNameResult  = ['BioResult',num2str(X)];

save(saveNameSD,'FinalSD');
save(saveNameSEM,'FinalSEM');
save(saveNameResult,'FinalResultsMatrix');

clearvars  -except X

end
```

1.4.4.6. *Convert model data into percentage observed using generated distributions*

```
%% Model data to percentage data for partially deficient data
% This script takes the model data and converts it to a percentage,
% much like the biological data for purely partially COX deficient crypts.

% Number of stem cells to be used for biological data manipulation

X = 5;

% Load the distribution that is going to be used DistributionPer4-16

fileimport = ['DistributionPer',num2str(X)];

load(fileimport);

% Load the ages that need to be analysed

load('SampleAges.mat');

% Where are all your files kept?

foldername = uigetdir('','Model Data');
cd(foldername)

% Make the folder directory list

listing = dir(foldername);
a = size(listing);

% Get the correct folder name from the parent directory

for yy = 1 : a(1)
   b = char(listing(yy,1).name);
   FolderNames(yy,1) = {b};
end

% Do the following procedure for each folder

DimFolder = size(FolderNames)-2;

% Final results for the pooled partially COX deficient crypts

FinalResults = [];

for pp = 3 : 3 + DimFolder(1) - 1

   tic

   % Open the main directory if it is not already open
```

```
cd(foldername)

% Open folder where the files are contained

filename = char(FolderNames(pp,1));

load(filename);

Data = gg.MutatedSCAgeFinal;

aa = size(Data);

Results = zeros(aa(1),aa(2));

for ii = 1 : aa(1)
   for jj = 1 : aa(2)

      if Data(ii,jj) == X;
         Results(ii,jj) = 100;
      elseif Data(ii,jj) == 0;
         Results(ii,jj) = 0;
      else Results(ii,jj) = DiscSampVec2((1:99),...
            yyAllFinal(ceil(Data(ii,jj)),:),1);
      end

   end
end

% Now all the results have been converted to percentage COX deficiency at
% the transverse level much like the biological COX deficiency data

%% Identify the numbers of partially COX deficient crypts

% Identify the correct data from the list of ages

AgeData = Results(SampleAges(pp-2)*52,:);

% Identify the number of elements within the age data

sizeAgeData = size(AgeData); sizeAgeData = sizeAgeData(2);

% Take out zero values and 100 values

AgeData(AgeData == 0) = [];
AgeData(AgeData == 100) = [];

% Pool all the data together for all ages.

FinalResults = [FinalResults,AgeData];
```

```
   toc

end

% Graph the results in the same format as the biologcial data so that the
% area under the curve is one again.

% Attain the values for the histogram

 [x1,c1] = hist(FinalResults);

 % At this stage x1 needs to be a percentage of all partial crypts looked
 % at

 x1 = x1 / sum(x1)*100;

 % Modify x1 and c1 so that it includes the start and end values

 a = 0;
 b = 100;

 x1 = [a x1 a]; c1 = [a c1 b];

 % Continuous frequency distribution to be applied to the frequency
 % distribution using pchip

 y1 = pchip(c1,x1,0:1:100);

 % Smooth the function

 yy1 = smooth(y1);

 %% Normalisation for area under the curve equal to 1

 yy1  = yy1 / sum(yy1);  % Probability distribution

% This probability distribution generated can be compared to that of the
% biological data.
```

1.4.4.7. *Graph model and biological partially deficient crypts as percentages*

```
%% Model Partial Plotting

% Plot all the yy1 values

figHandle = figure(1);
set(gcf,'color','w');
set(gcf,'units','normalized','outerposition',[0 0 1 1]);

    plot(0:100,yy1Mod4SC,'b','LineWidth',2)
    hold
    plot(0:100,yy1Mod5SC,'r','LineWidth',2)
    plot(0:100,yy1Mod6SC,'g','LineWidth',2)
    plot(0:100,yy1Mod12SC,'m','LineWidth',2)
    plot(0:100,yy1Bio,'-.k','LineWidth',2)
     set(gca, ...
     'Box'         , 'off'      ,...
     'TickDir'     , 'out'      ,...
     'TickLength'  , [.01 .01] ,...
     'XColor'      , 'k'        ,...
     'YColor'      , 'k'        ,...
     'XTick'       , 0:1:5      ,...
     'LineWidth'   , 2          ,...
     'FontSize'    , 8          ,...
     'XTick'       , 0:10:100        ,...
     'XTickLabel',{'0','10','20','30','40','50','60','70','80','90','100'});
     xlabel('Percentage COX deficiency of individual crypts',...
        'FontWeight','Bold','FontSize',12);
     ylabel('Probability',...
        'FontWeight','Bold','FontSize',12);
     legend('Mod4SC','Mod5SC','Mod6SC',...
        'Mod12SC','Bio',...
        'Location','NorthEastOutside');
```