

SEACAR Oyster Analysis

Last compiled on 06 October, 2025

Contents

Important Notes	1
Oyster Figures	2
Density	2
Apalachicola Bay Aquatic Preserve	2
Apalachicola National Estuarine Research Reserve	5
Boca Ciega Bay Aquatic Preserve	8
Estero Bay Aquatic Preserve	10
Guana River Marsh Aquatic Preserve	13
Guana Tolomato Matanzas National Estuarine Research Reserve	15
Indian River-Malabar to Vero Beach Aquatic Preserve	17
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve	19
Jensen Beach to Jupiter Inlet Aquatic Preserve	21
Lemon Bay Aquatic Preserve	23
Loxahatchee River-Lake Worth Creek Aquatic Preserve	25
Mosquito Lagoon Aquatic Preserve	27
Nassau River-St. Johns River Marshes Aquatic Preserve	29
Pine Island Sound Aquatic Preserve	31
Pinellas County Aquatic Preserve	34
St. Martins Marsh Aquatic Preserve	36
Tomoka Marsh Aquatic Preserve	38
Yellow River Marsh Aquatic Preserve	40
Percent Live	43
Apalachicola Bay Aquatic Preserve	43
Apalachicola National Estuarine Research Reserve	46
Boca Ciega Bay Aquatic Preserve	49
Estero Bay Aquatic Preserve	51
Guana River Marsh Aquatic Preserve	54
Guana Tolomato Matanzas National Estuarine Research Reserve	56
Indian River-Malabar to Vero Beach Aquatic Preserve	58
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve	60
Jensen Beach to Jupiter Inlet Aquatic Preserve	62
Lemon Bay Aquatic Preserve	64
Loxahatchee River-Lake Worth Creek Aquatic Preserve	66
Mosquito Lagoon Aquatic Preserve	68
Nassau River-St. Johns River Marshes Aquatic Preserve	70
Pine Island Sound Aquatic Preserve	72
Pinellas County Aquatic Preserve	75
St. Martins Marsh Aquatic Preserve	77
Tomoka Marsh Aquatic Preserve	79
Yellow River Marsh Aquatic Preserve	81

Shell Height	83
Apalachicola Bay Aquatic Preserve	83
Apalachicola National Estuarine Research Reserve	84
Big Bend Seagrasses Aquatic Preserve	86
Boca Ciega Bay Aquatic Preserve	87
Estero Bay Aquatic Preserve	89
Guana River Marsh Aquatic Preserve	91
Guana Tolomato Matanzas National Estuarine Research Reserve	93
Indian River-Malabar to Vero Beach Aquatic Preserve	94
Indian River-Vero Beach to Ft. Pierce Aquatic Preserve	96
Jensen Beach to Jupiter Inlet Aquatic Preserve	97
Lemon Bay Aquatic Preserve	99
Loxahatchee River-Lake Worth Creek Aquatic Preserve	100
Mosquito Lagoon Aquatic Preserve	102
Nassau River-St. Johns River Marshes Aquatic Preserve	103
Nature Coast Aquatic Preserve	105
Pine Island Sound Aquatic Preserve	106
Pinellas County Aquatic Preserve	108
St. Martins Marsh Aquatic Preserve	110
Yellow River Marsh Aquatic Preserve	111
Libraries and Settings	113
File Import	114
Data Setup & Filtering	115
Managed Area Statistics	118
Density	118
Shell Height	121
Percent Live	124
Plotting setup	127
Oyster Shell Height Analysis	127
Density Analysis	140
Percent Live Analysis	147
Save & Export Results	154

Important Notes

The purpose of this script is to group oyster data, create managed area statistics, perform lme analysis on density, percent live, and shell height, and create reports in pdf and Word document form for Oyster data.

All scripts and outputs can be found on the SEACAR GitHub repository:

https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses

This markdown file is designed to be compiled by `Oyster_ReportRender.R` (https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/blob/main/Oyster/Oyster_ReportRender.R).

This script is based off of code originally written by Stephen Durham.

Compiled and edited by [J.E. Panzik](#) and [Tyler Hill](#) for SEACAR.

Oyster Figures

Density

Apalachicola Bay Aquatic Preserve

Natural

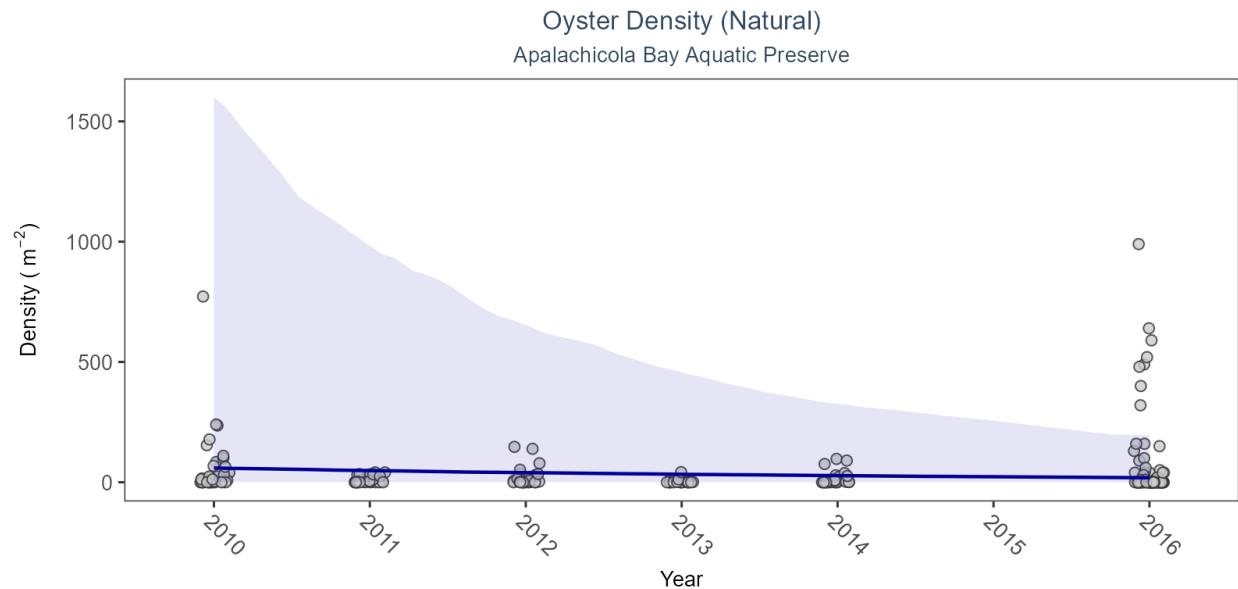


Figure 1: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 1: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-6.43	40.37	-0.08 to -225.39

For natural reefs, density decreased by an average of 6.61 oysters per square meter per year. For restored reefs, density decreased by an average of 5.53 oysters per square meter per year.

Restored

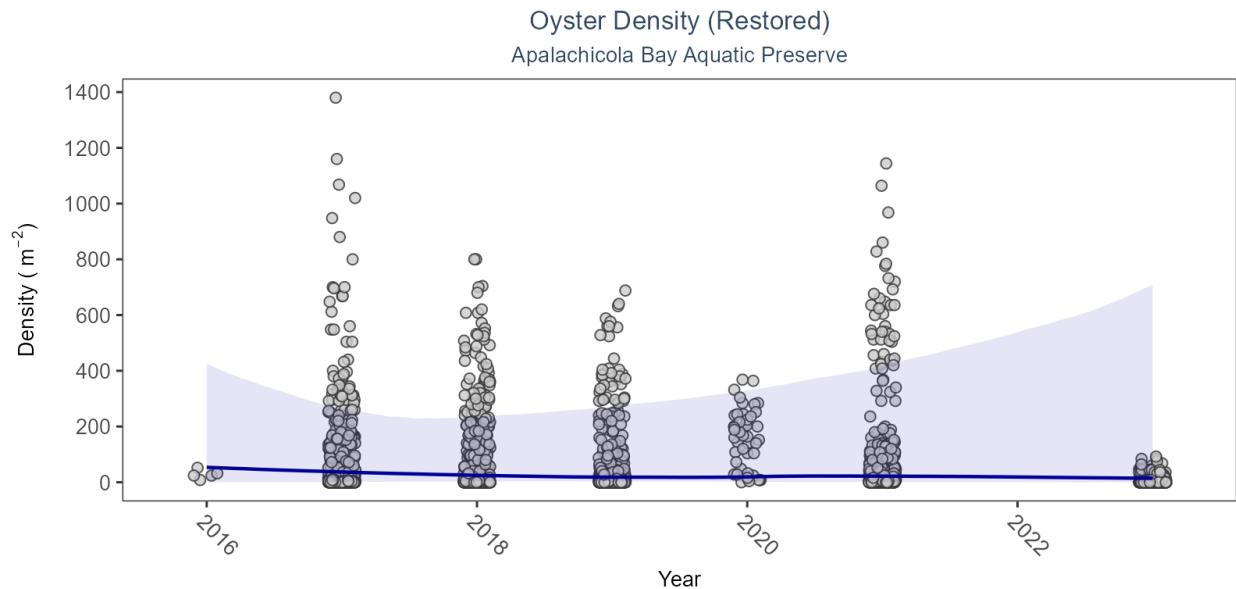
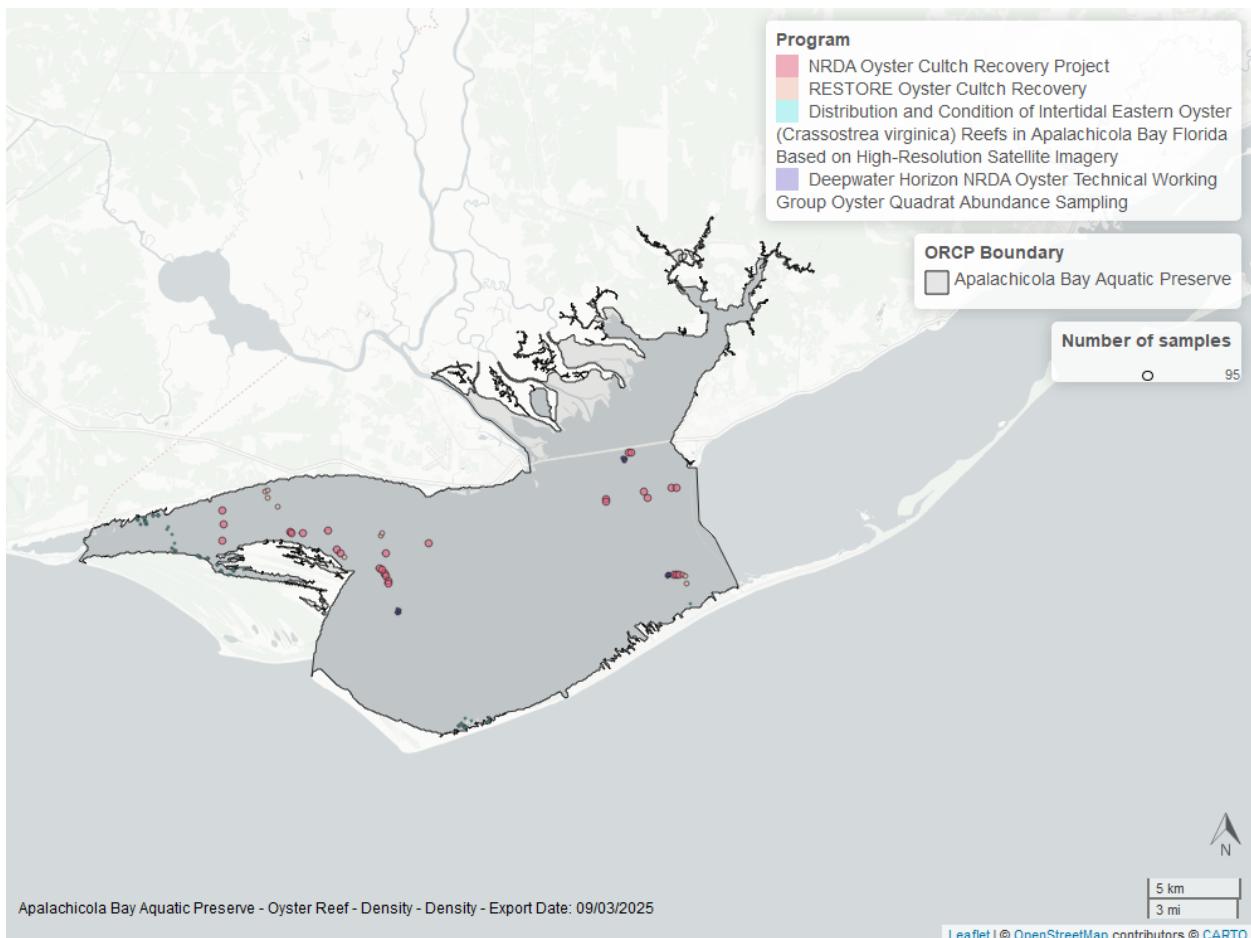


Figure 2: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 2: Model results for Oyster Density - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	No significant change	-5.99	28.88	-0.05 to 32.28

For natural reefs, density decreased by an average of 6.61 oysters per square meter per year. For restored reefs, density decreased by an average of 5.53 oysters per square meter per year.



Apalachicola National Estuarine Research Reserve

Natural

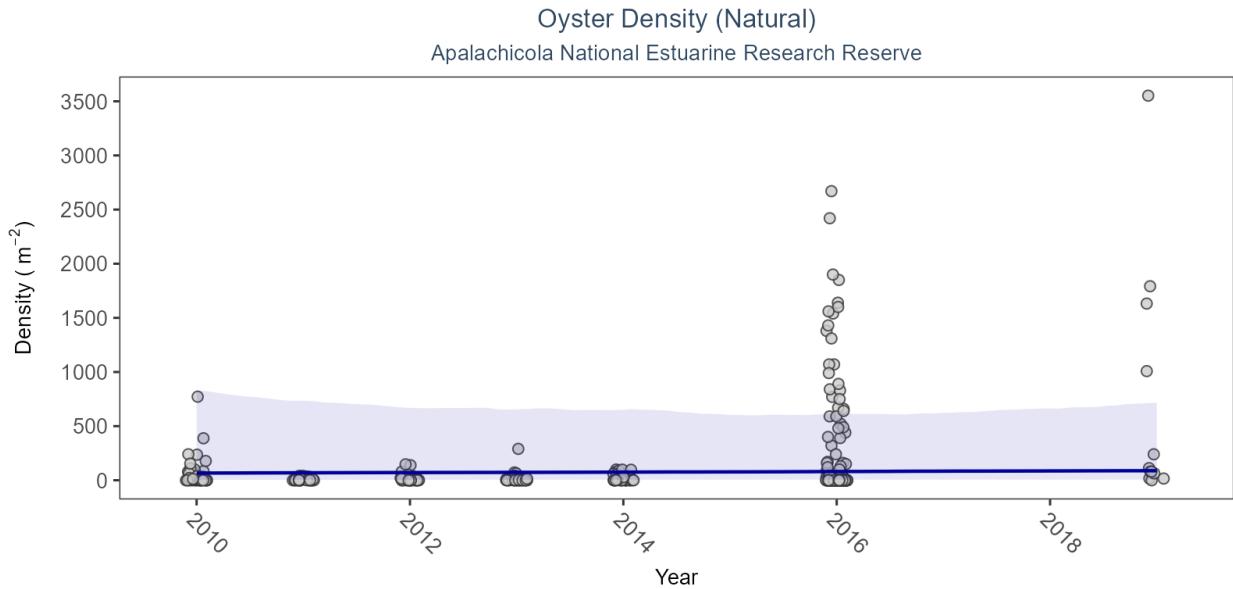


Figure 3: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 3: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	1.96	74.38	0.34 to -23.11

For natural reefs, density increased by an average of 2.03 oysters per square meter per year. For restored reefs, density decreased by an average of 11.42 oysters per square meter per year.

Restored

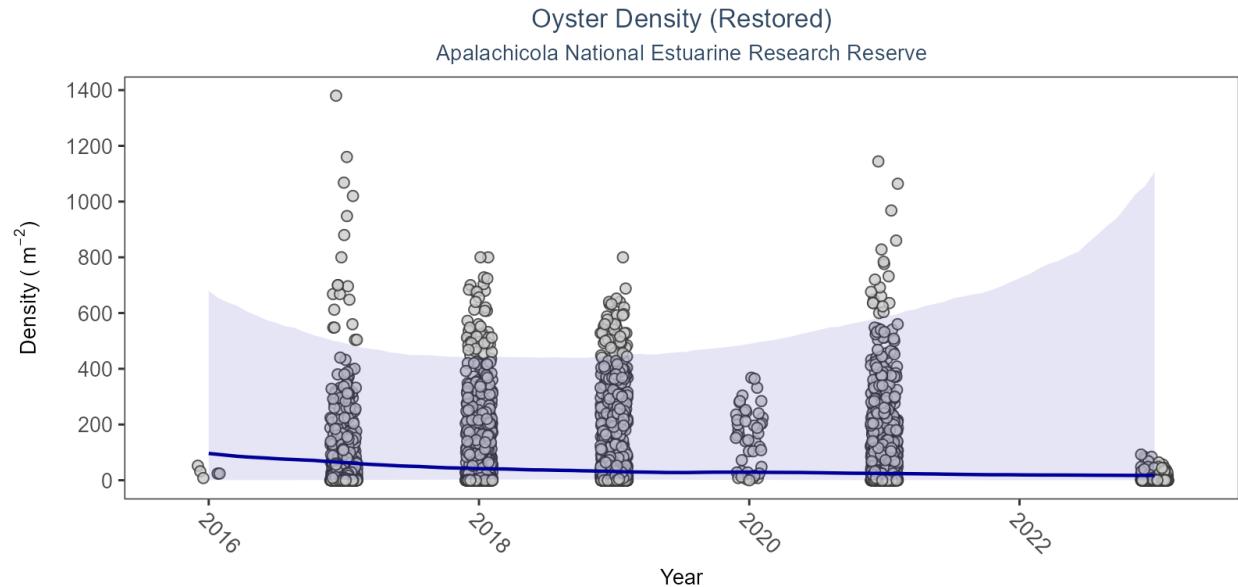
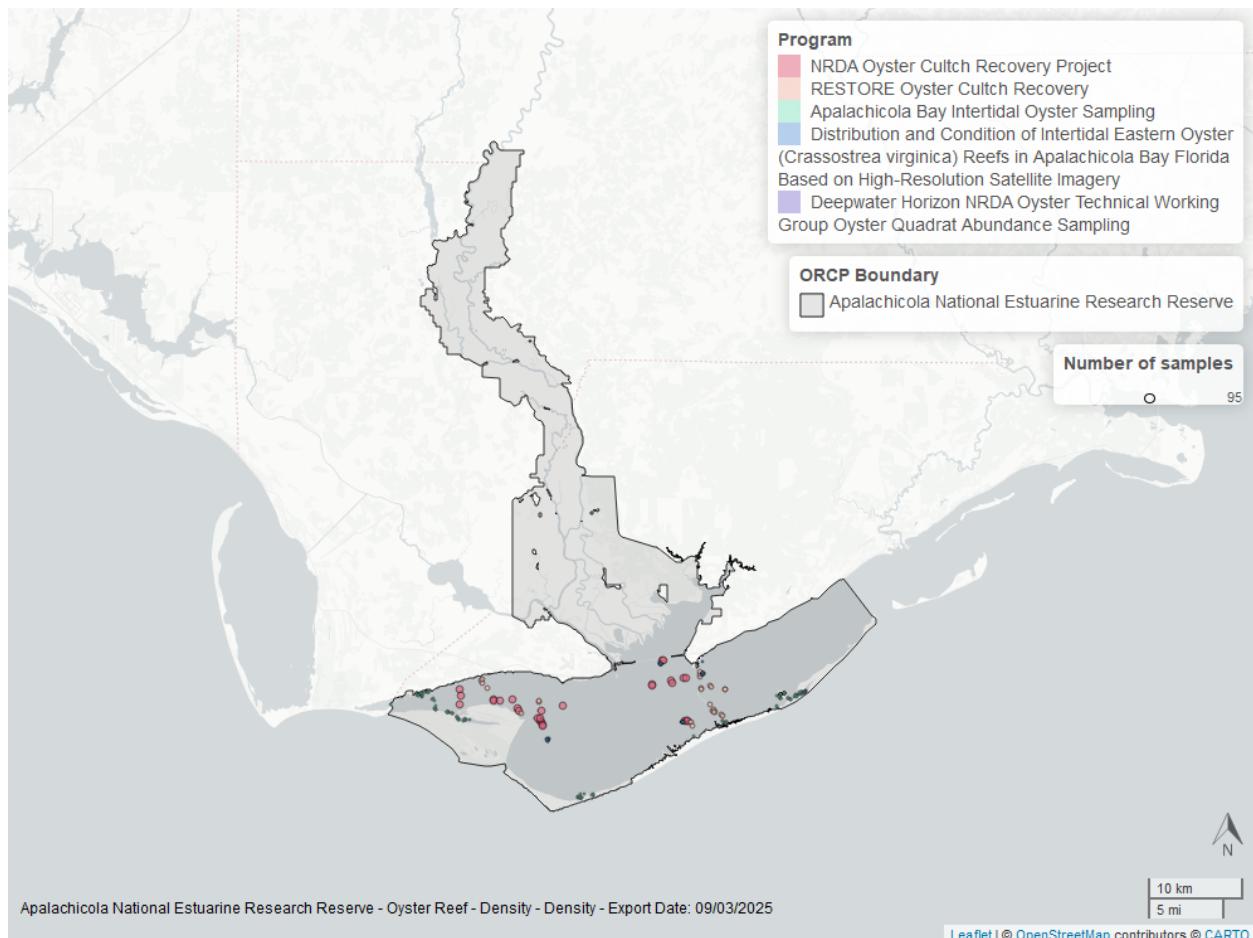


Figure 4: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 4: Model results for Oyster Density - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	No significant change	-11.95	59.04	-0.03 to 77.11

For natural reefs, density increased by an average of 2.03 oysters per square meter per year. For restored reefs, density decreased by an average of 11.42 oysters per square meter per year.



Boca Ciega Bay Aquatic Preserve

Natural

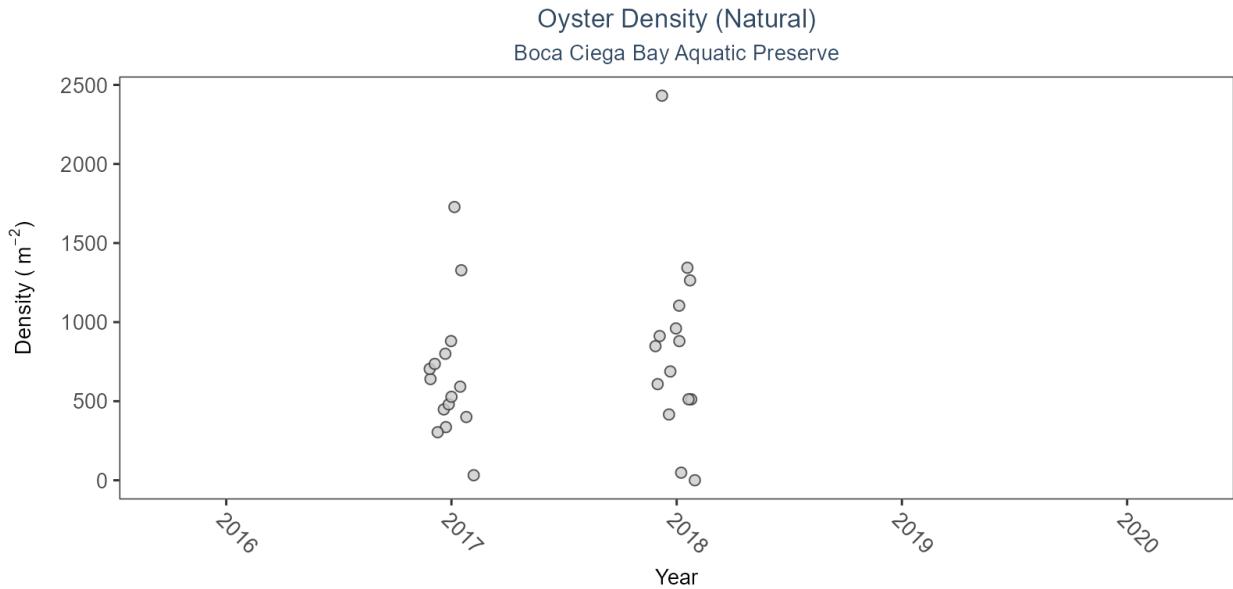
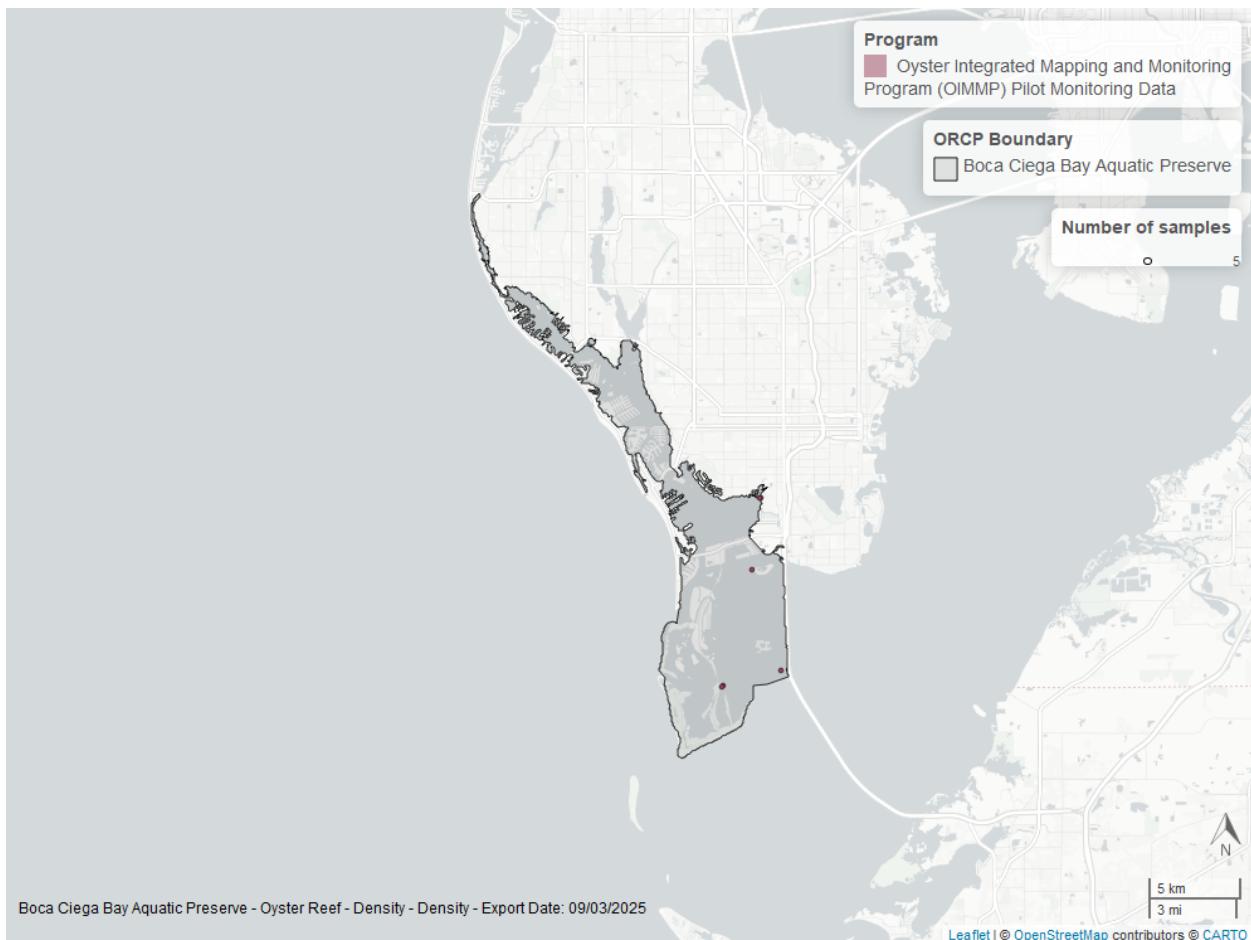


Figure 5: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 5: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Estero Bay Aquatic Preserve

Restored

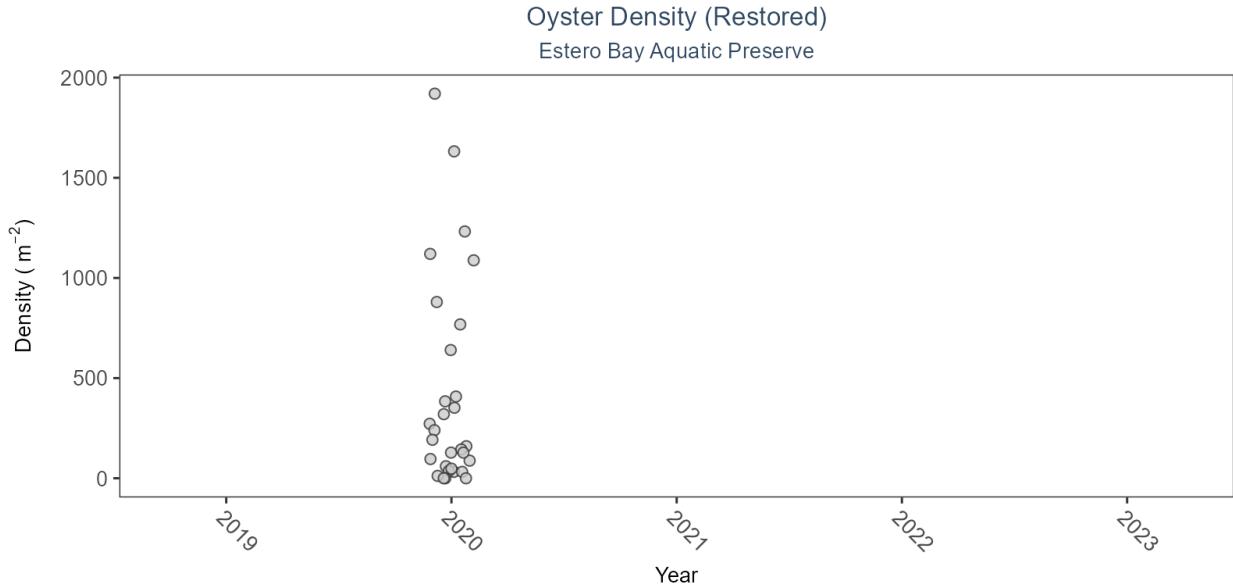


Figure 6: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 6: Model results for Oyster Density - Restored

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Restored	-	-	-	-

For natural reefs, density decreased by an average of 6.54 oysters per square meter per year. For restored reefs, there was insufficient data to calculate a trend for density.

Natural

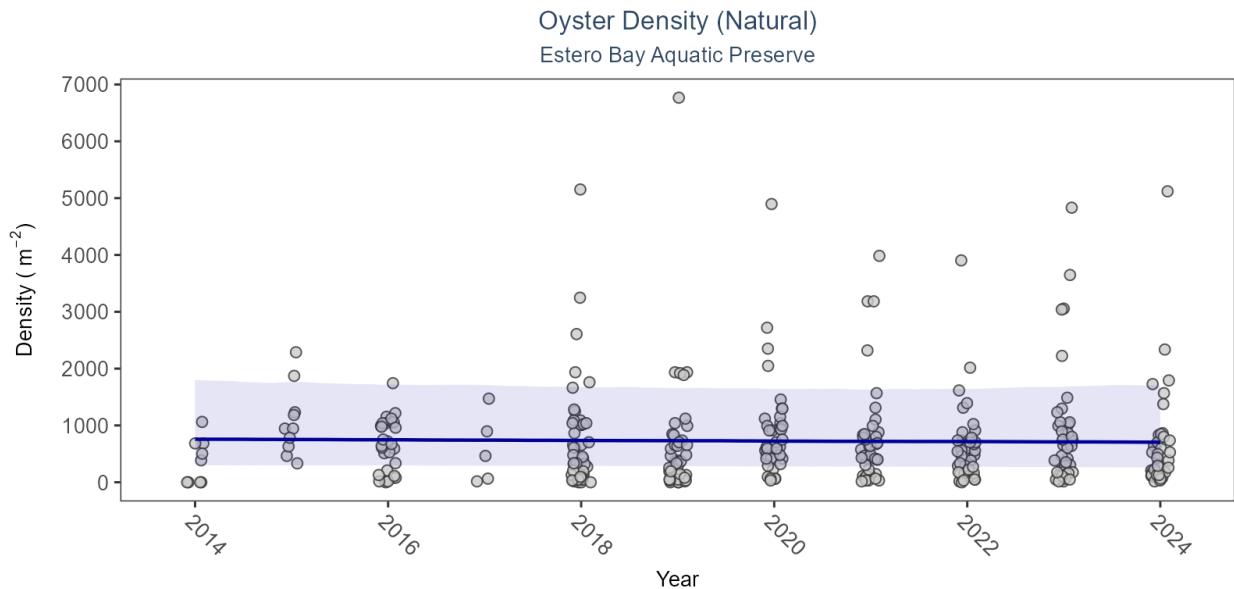
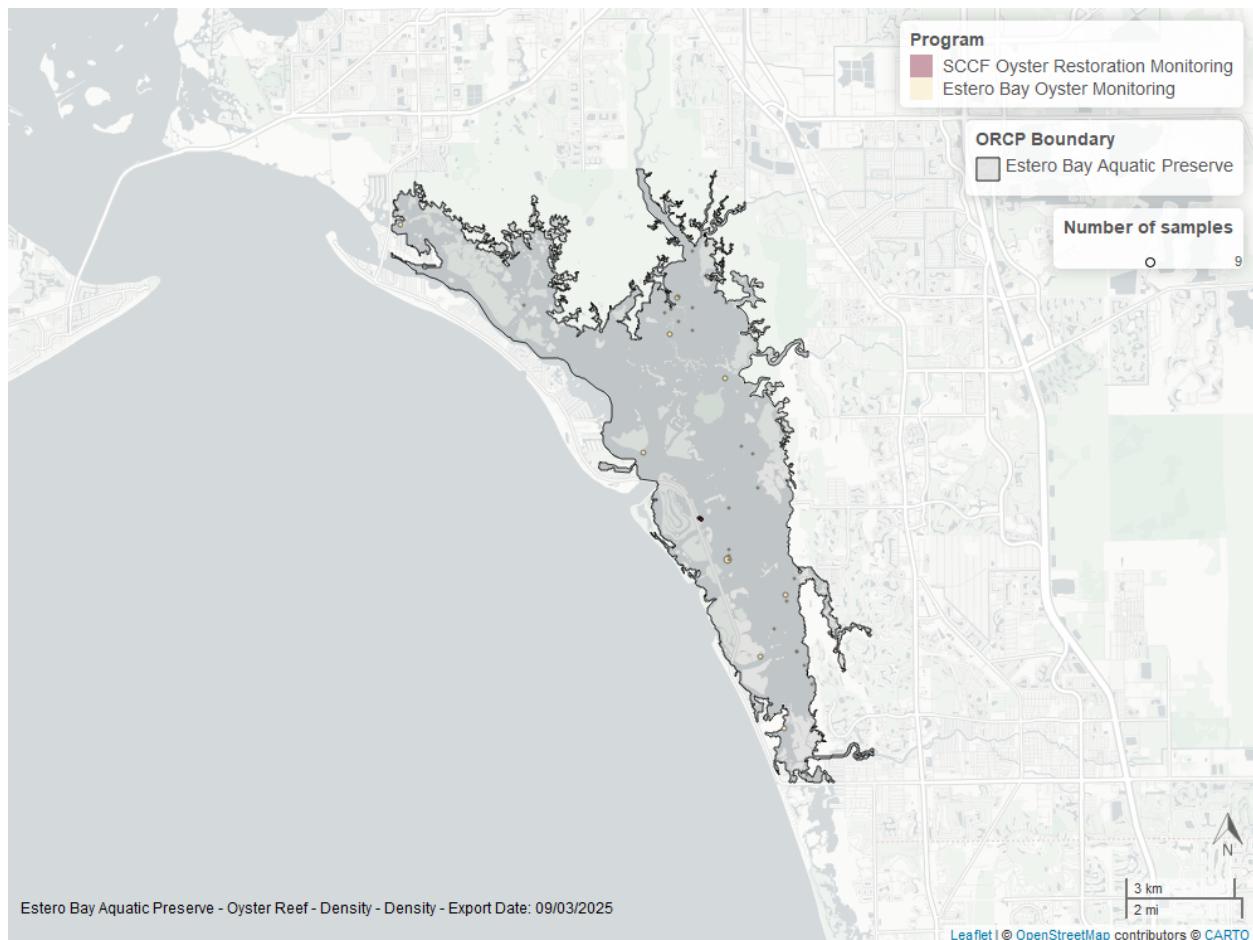


Figure 7: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 7: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-6.42	289.31	-3.28 to -13.5

For natural reefs, density decreased by an average of 6.54 oysters per square meter per year. For restored reefs, there was insufficient data to calculate a trend for density.



Guana River Marsh Aquatic Preserve

Natural

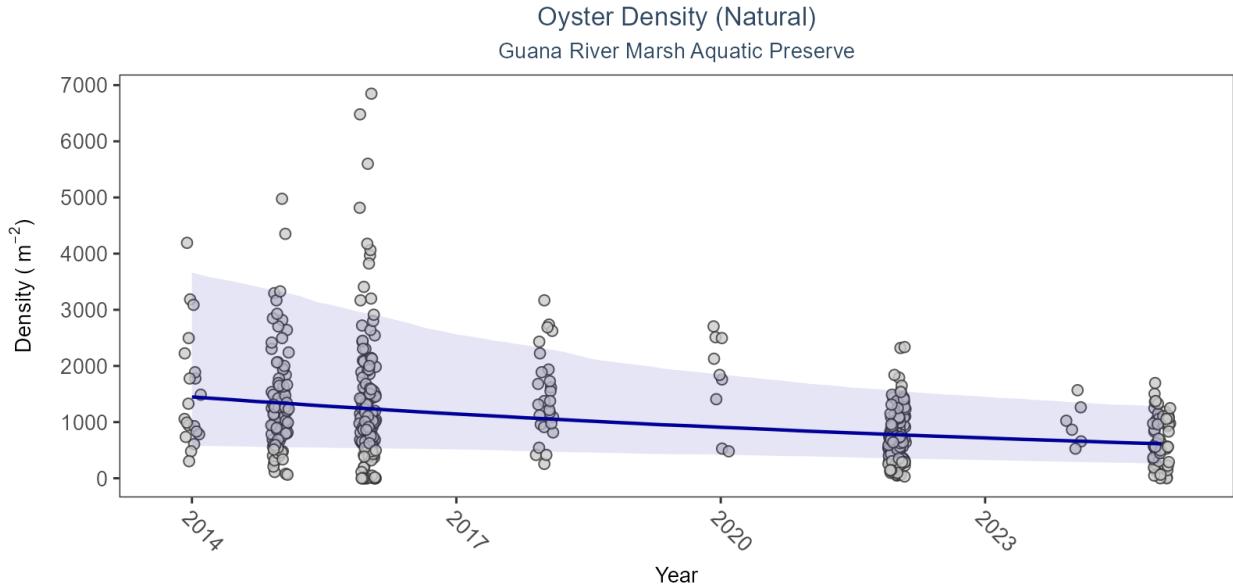
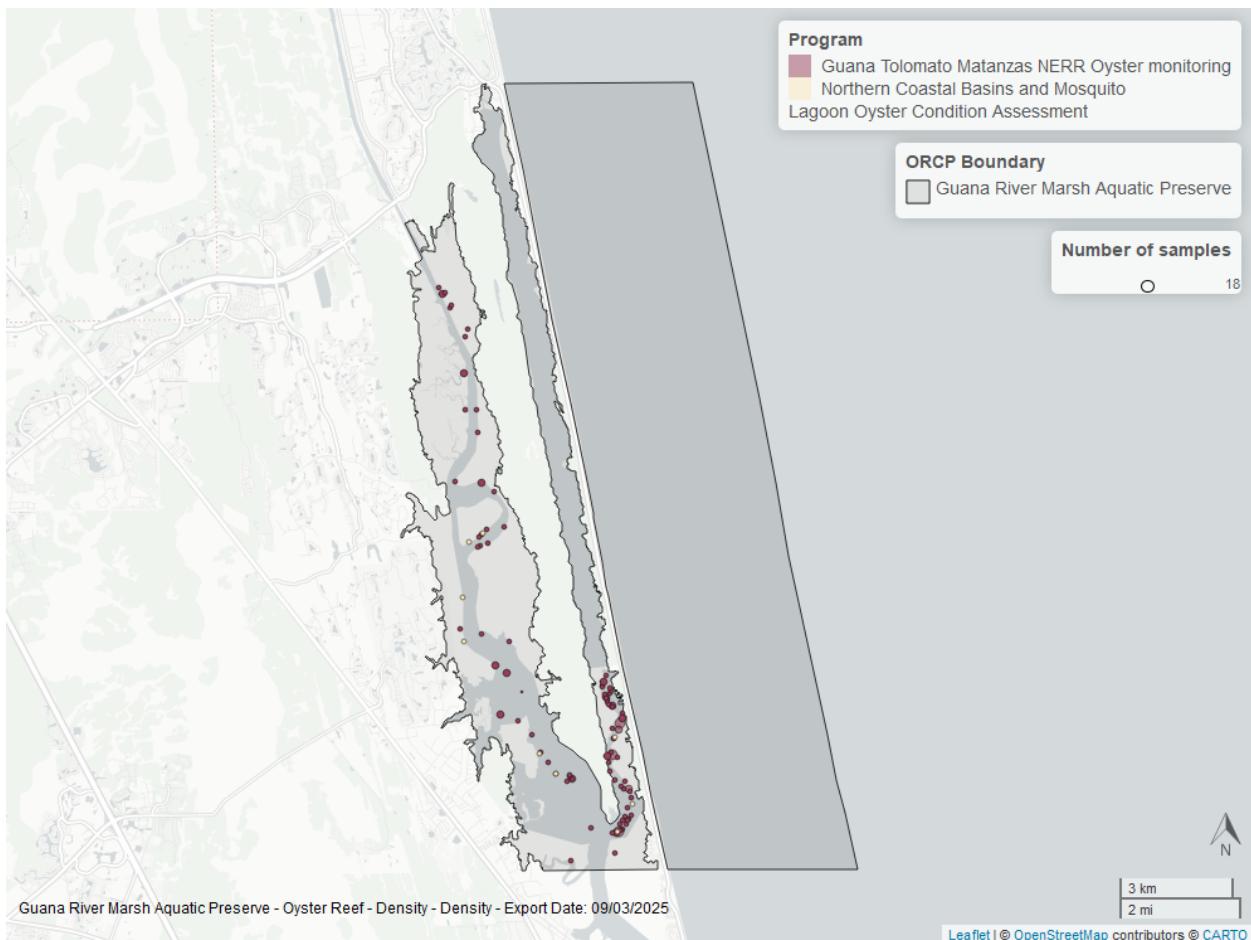


Figure 8: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 8: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-74.15	362.85	-27.5 to -207.29

For natural reefs, density decreased by an average of 75.9 oysters per square meter per year.



Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

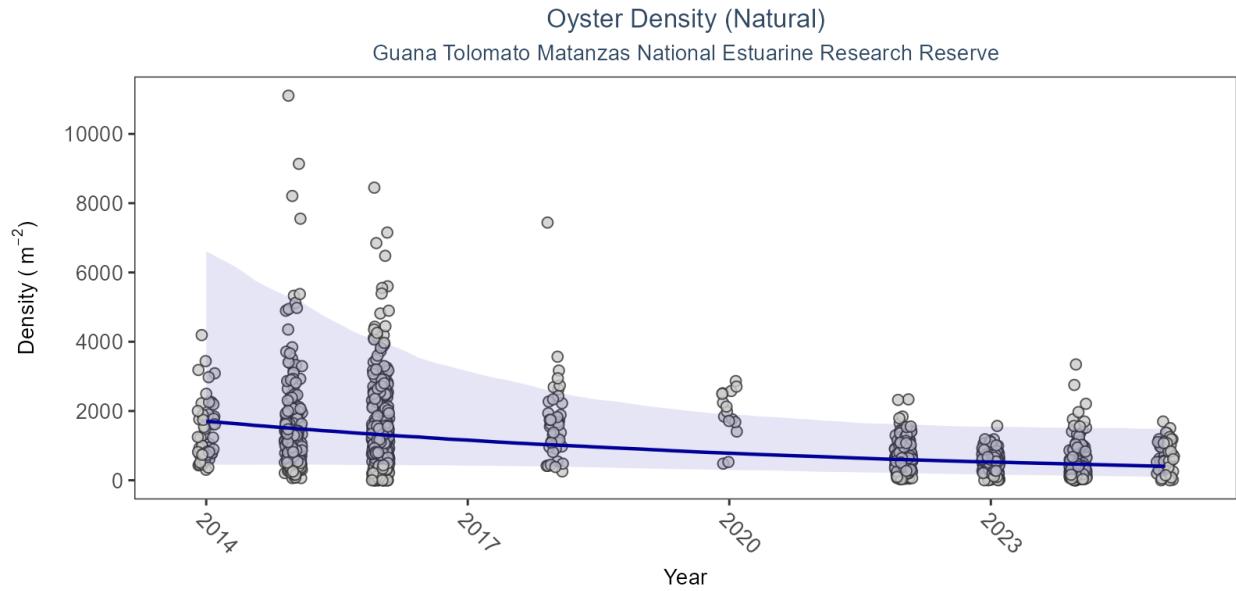
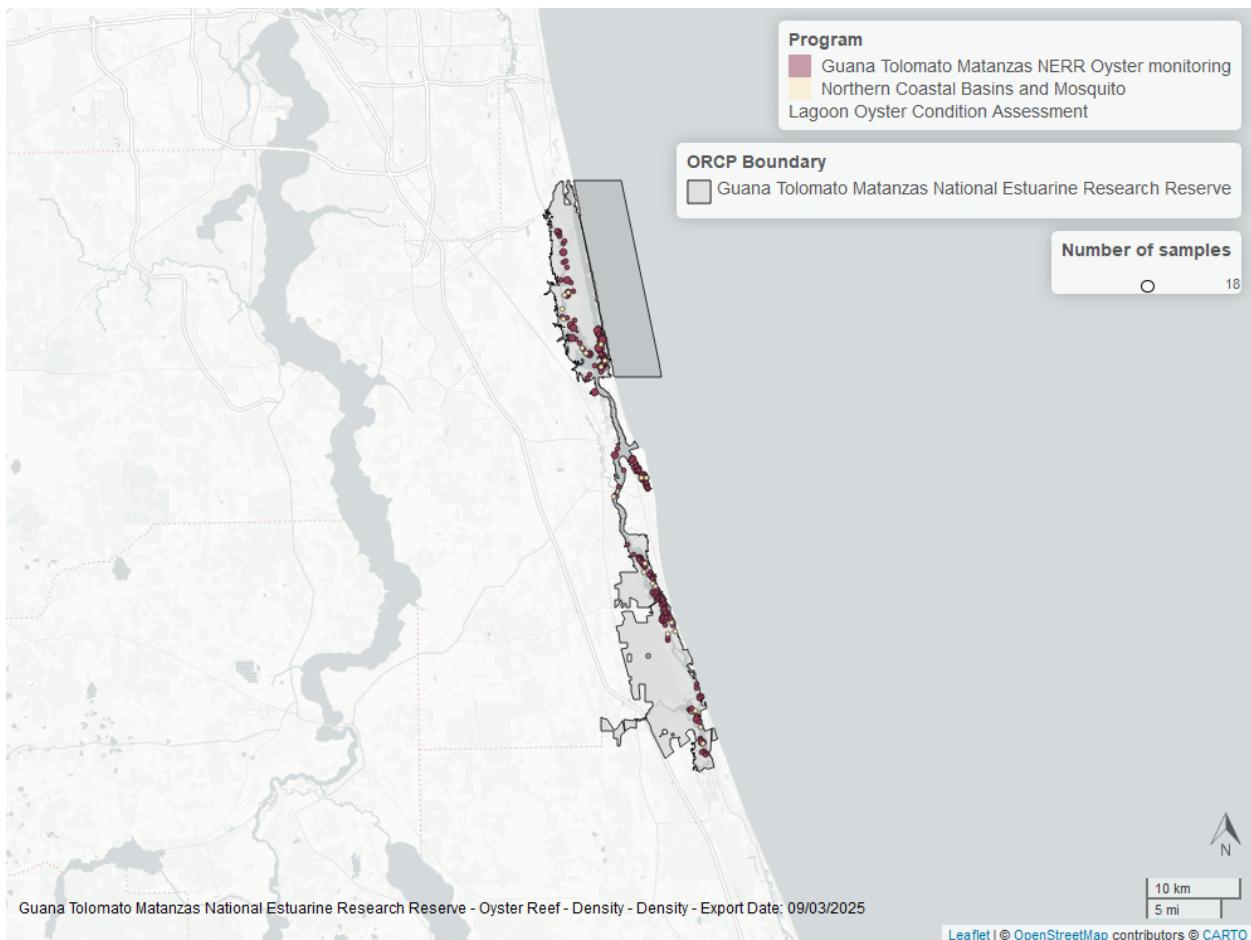


Figure 9: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 9: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-115.49	465.61	-31.66 to -445.43

For natural reefs, density decreased by an average of 116.06 oysters per square meter per year.



Indian River-Malabar to Vero Beach Aquatic Preserve

Natural

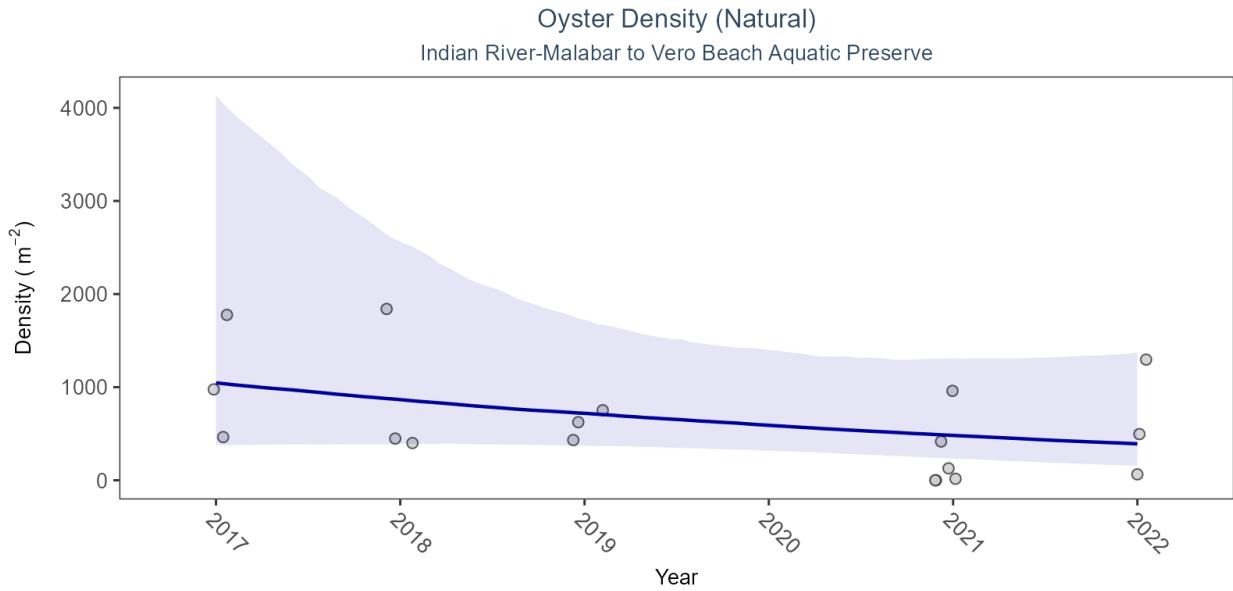
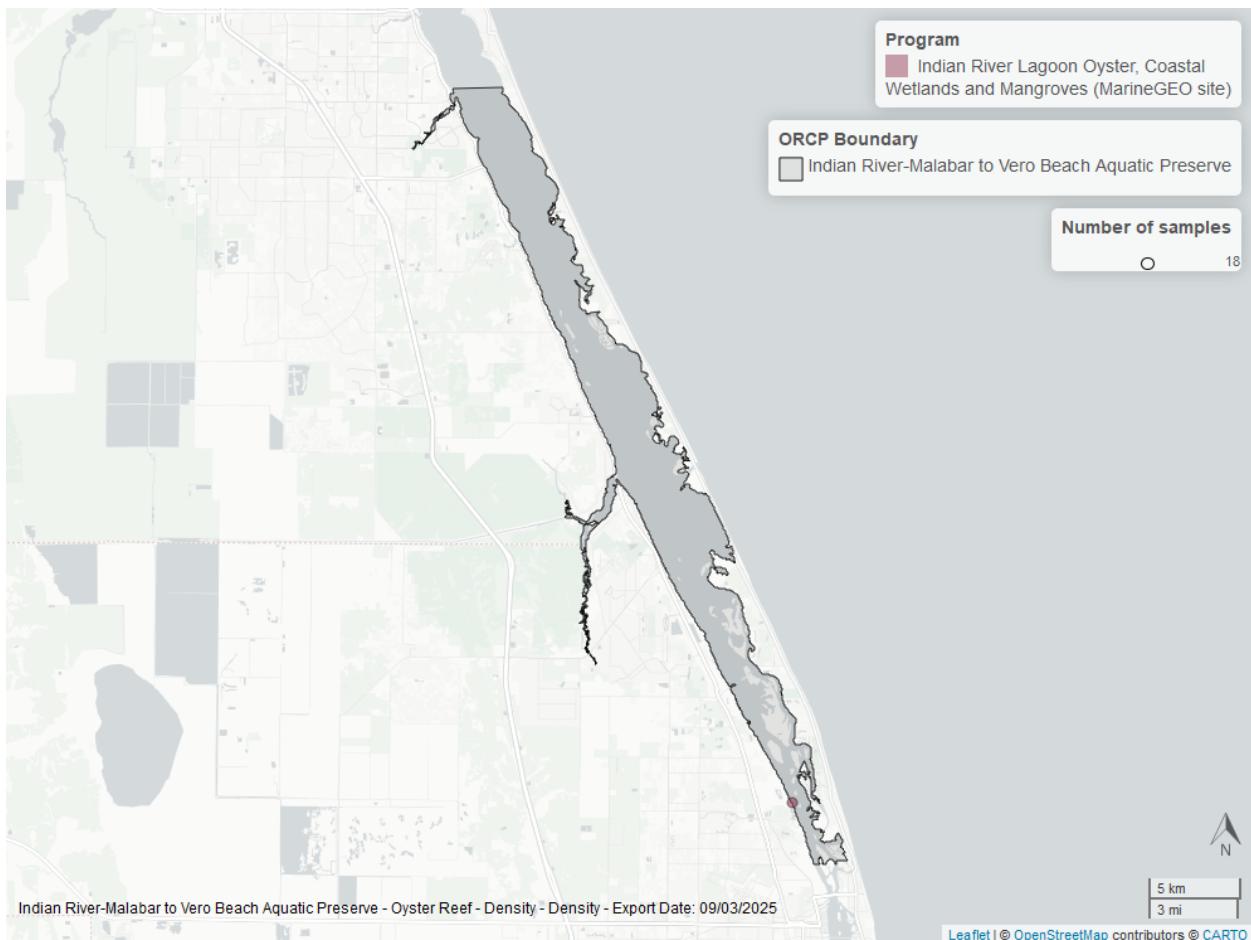


Figure 10: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 10: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-130.86	278.08	-44.24 to -551.55

For natural reefs, density decreased by an average of 130.86 oysters per square meter per year.



Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

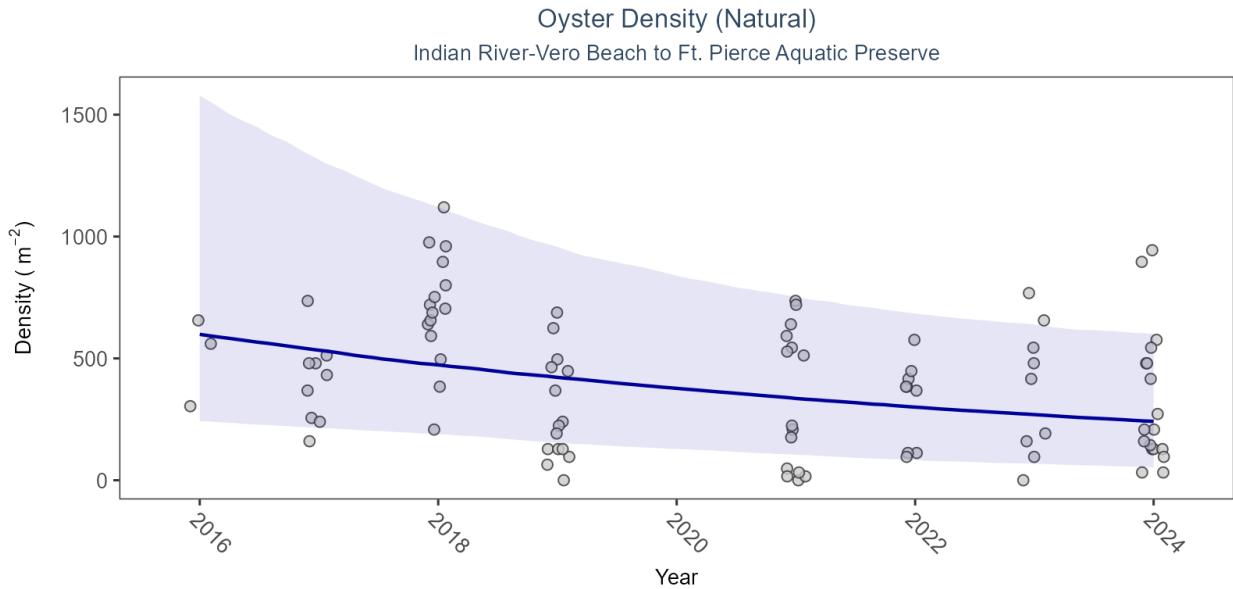
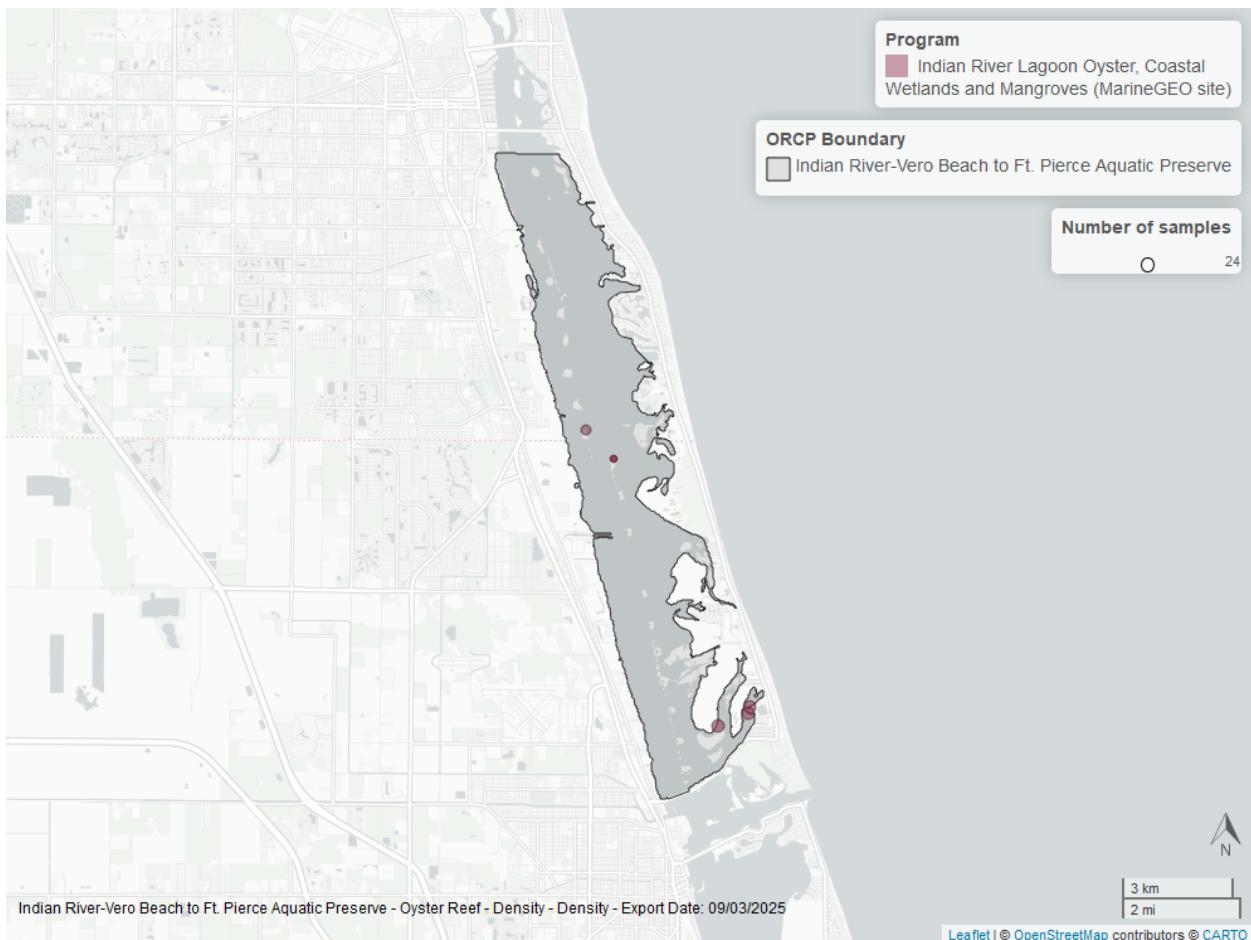


Figure 11: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 11: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-44.1	218.52	-23.88 to -118.93

For natural reefs, density decreased by an average of 44.37 oysters per square meter per year.



Jensen Beach to Jupiter Inlet Aquatic Preserve

Natural

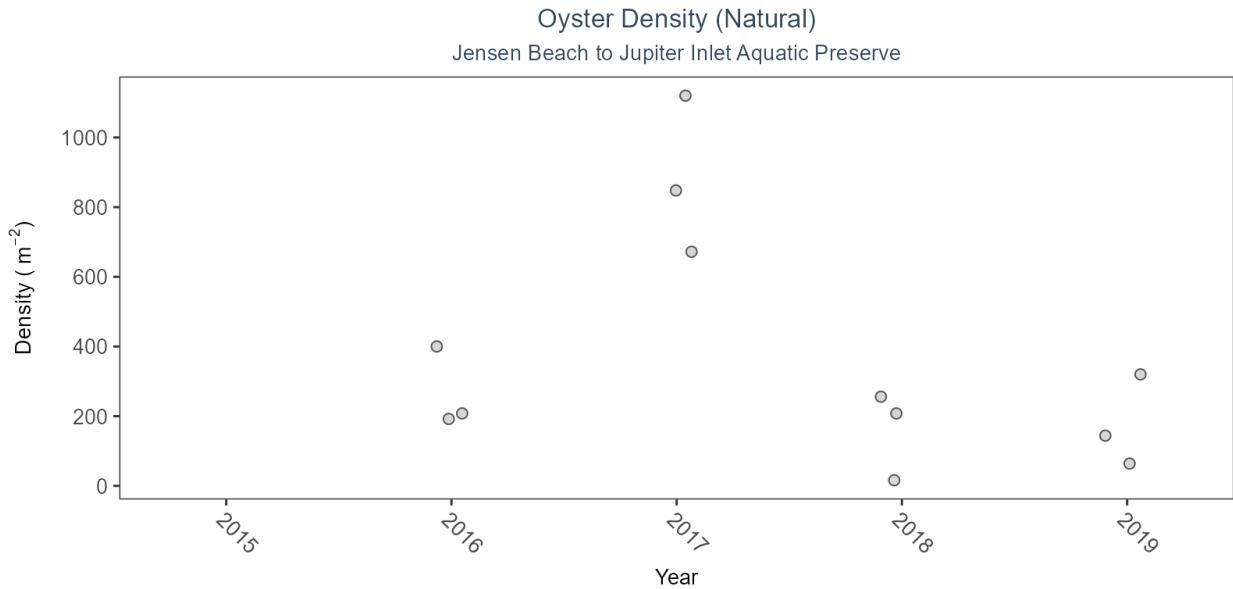
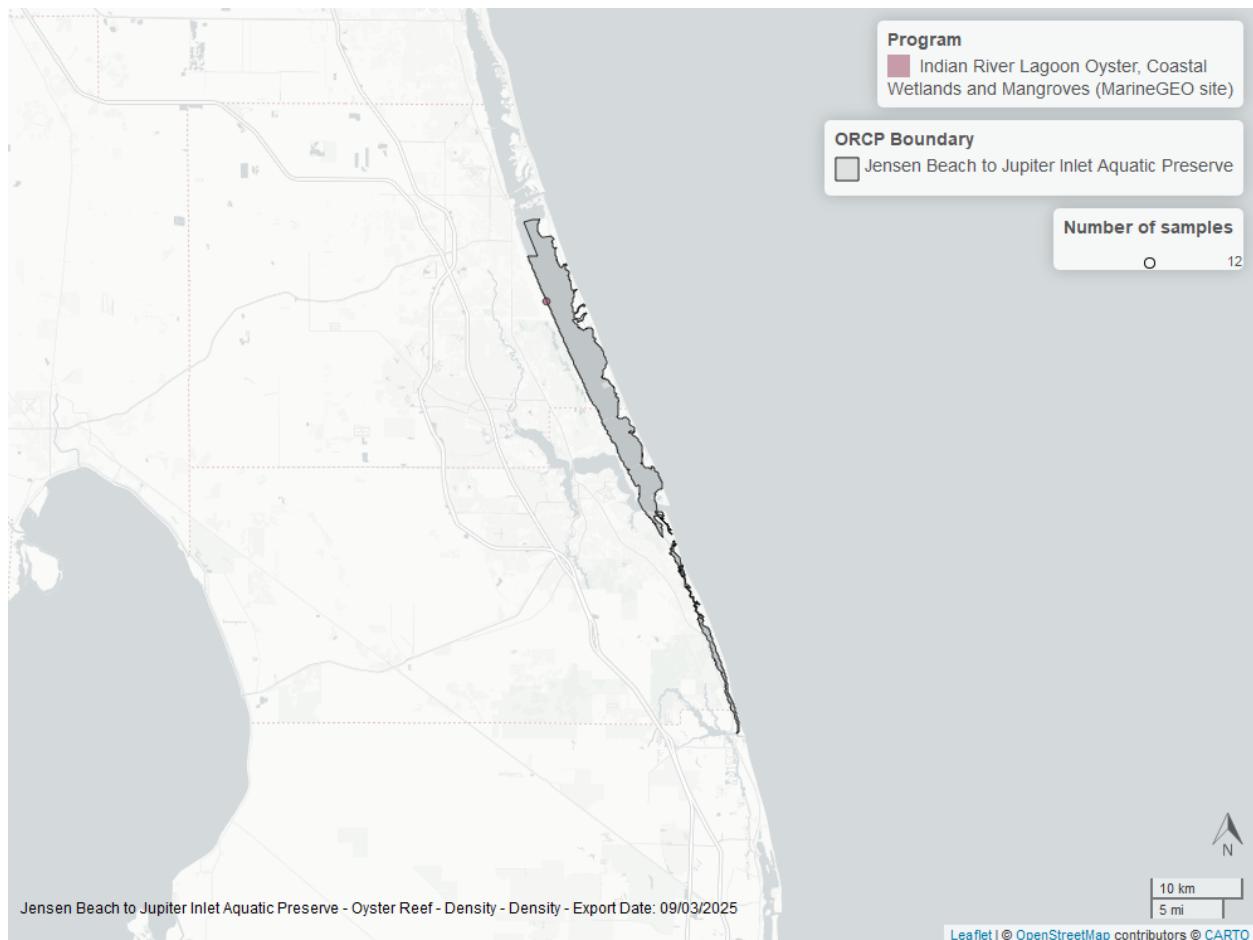


Figure 12: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 12: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Lemon Bay Aquatic Preserve

Natural

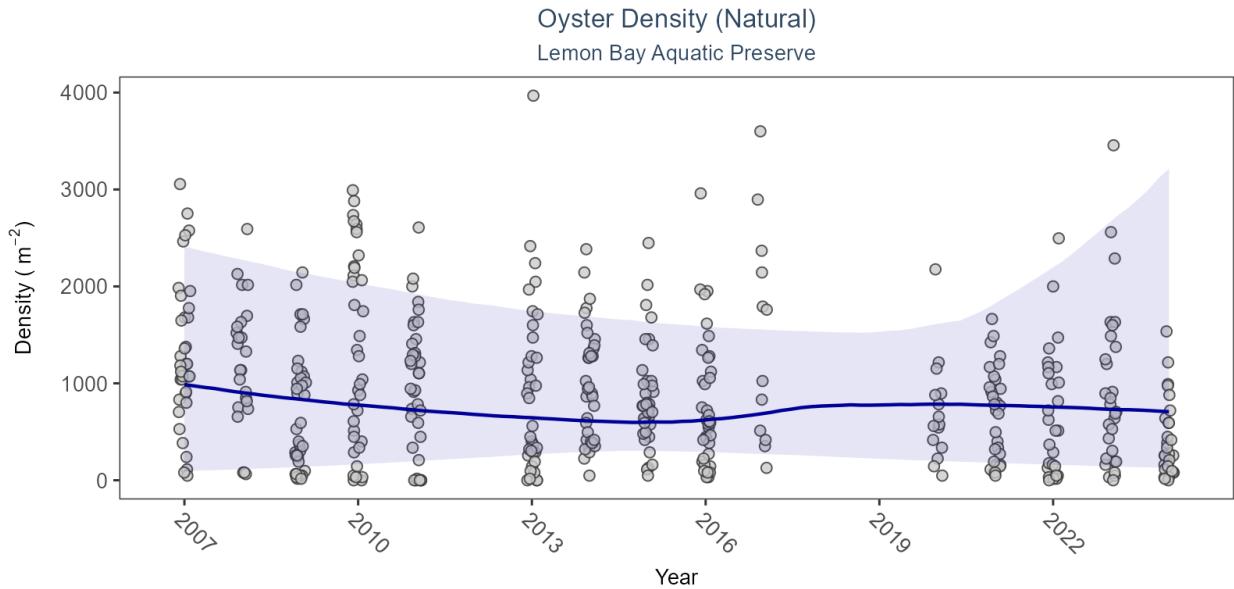
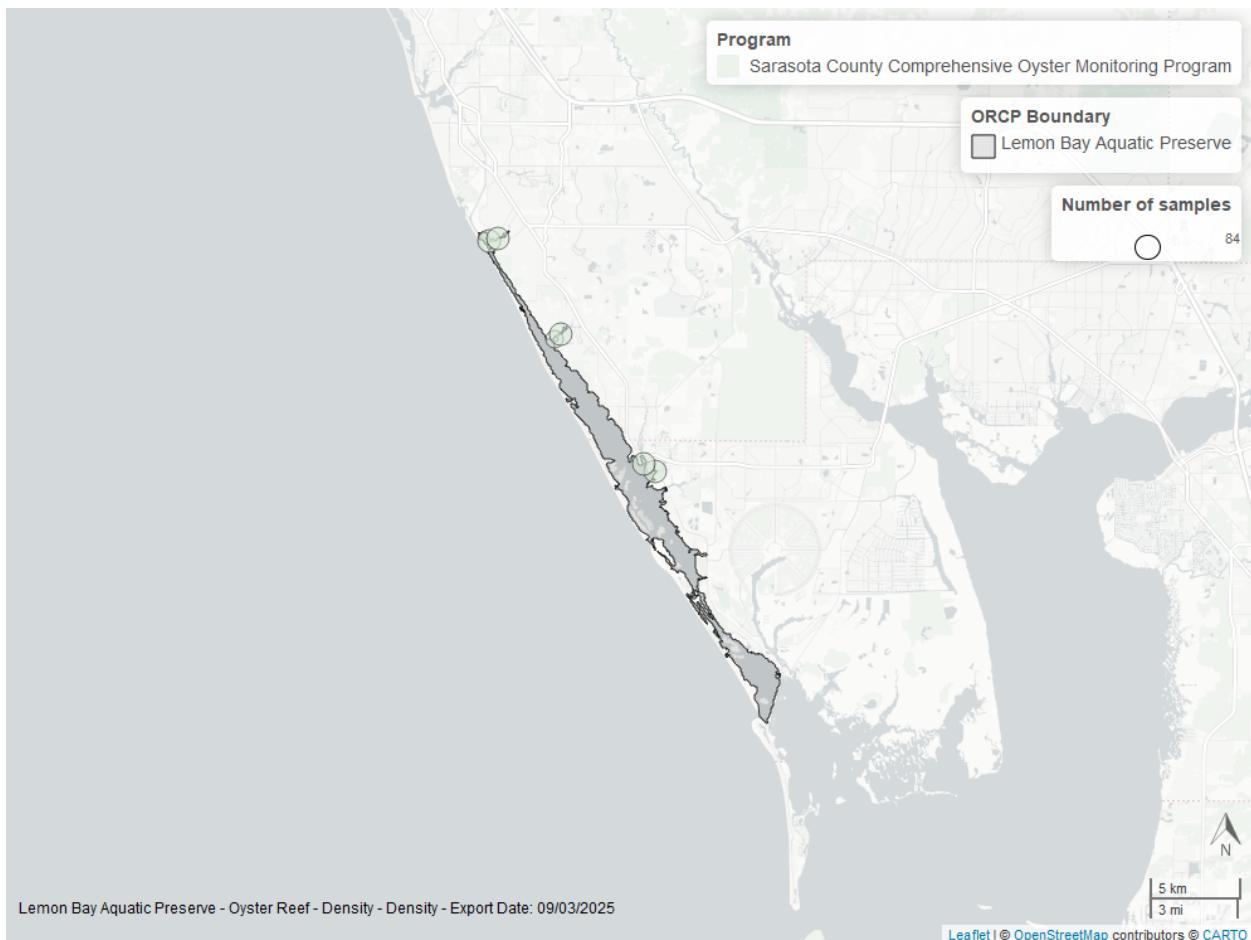


Figure 13: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 13: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-18.38	612.39	1.95 to 25.27

For natural reefs, density decreased by an average of 17.58 oysters per square meter per year.



Loxahatchee River-Lake Worth Creek Aquatic Preserve

Natural

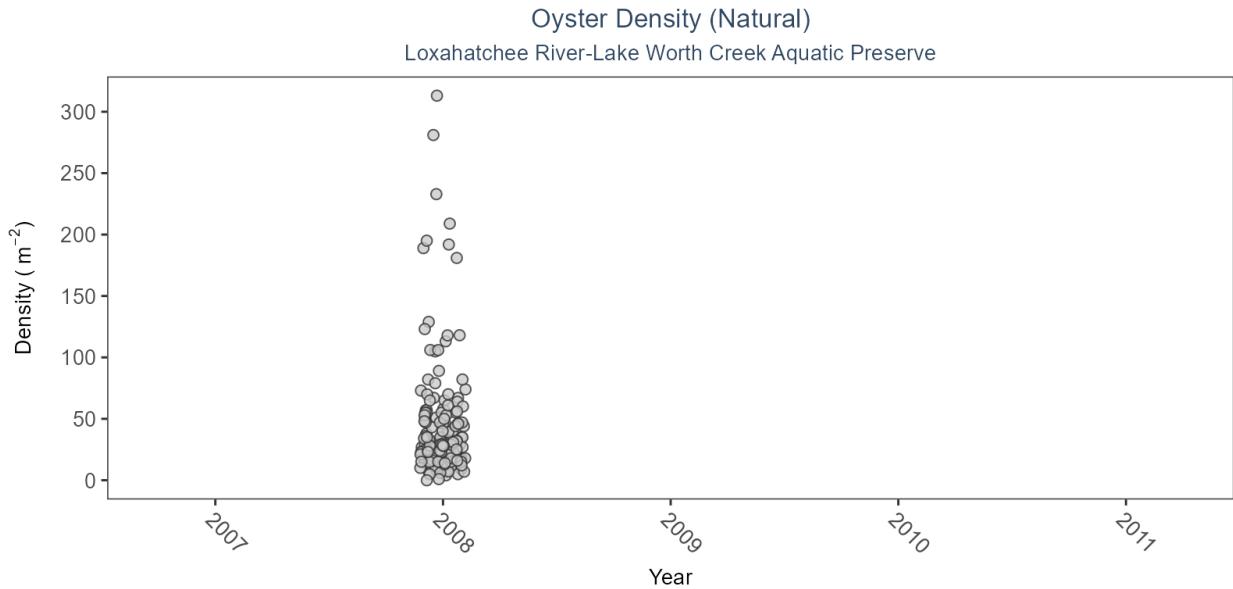
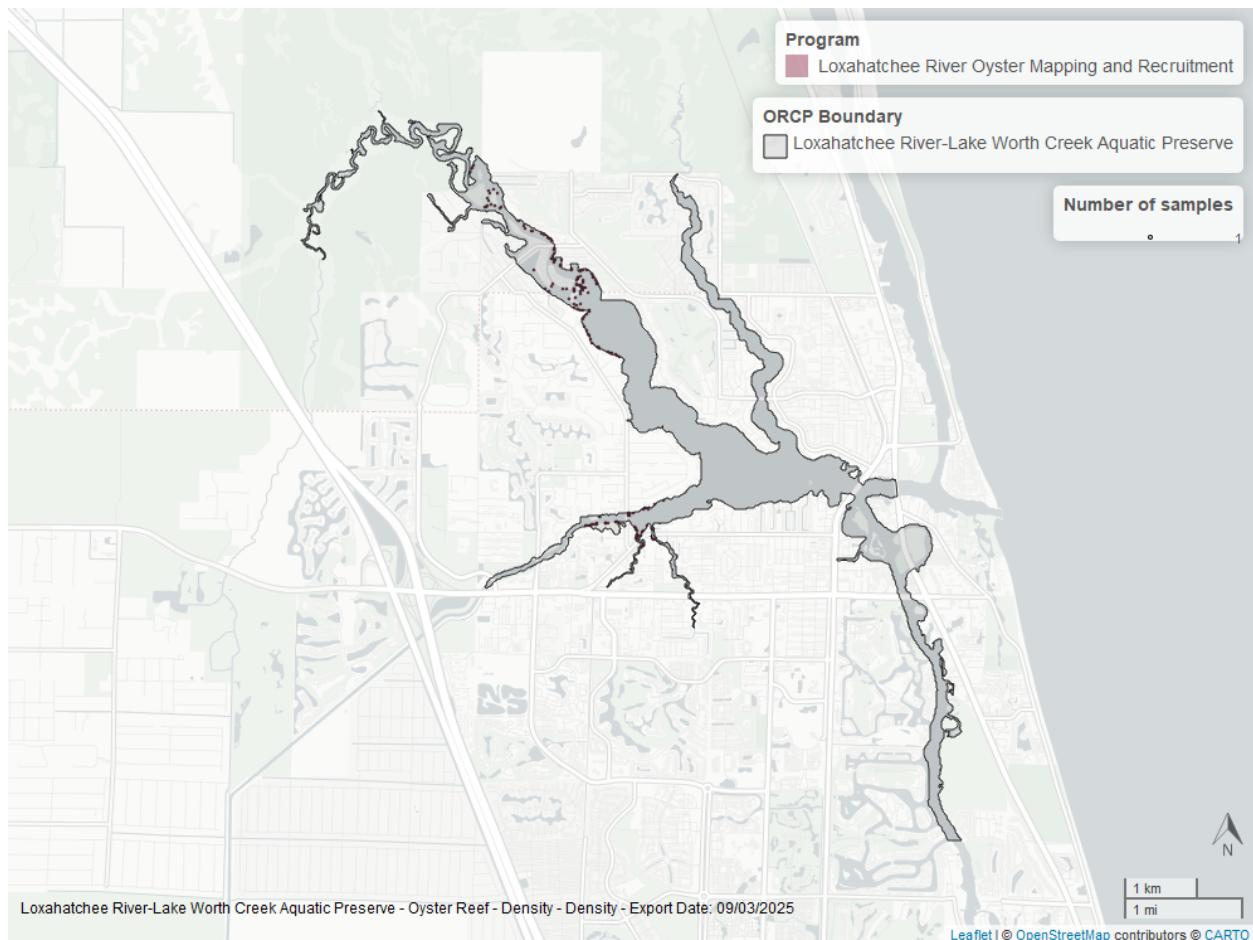


Figure 14: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 14: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Mosquito Lagoon Aquatic Preserve

Natural

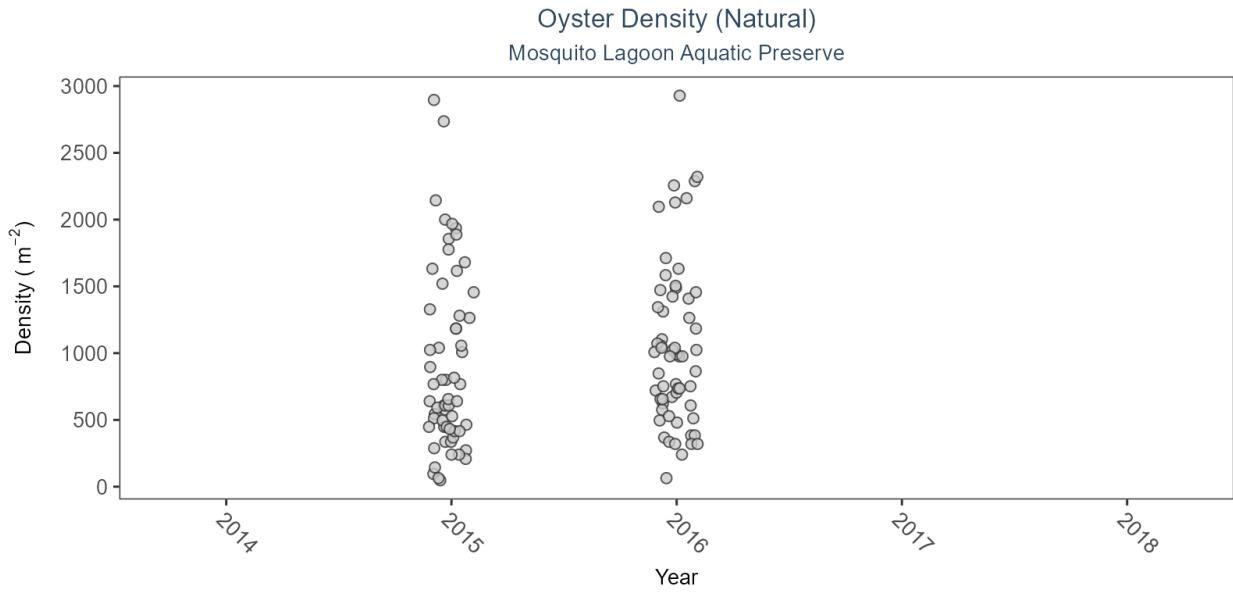
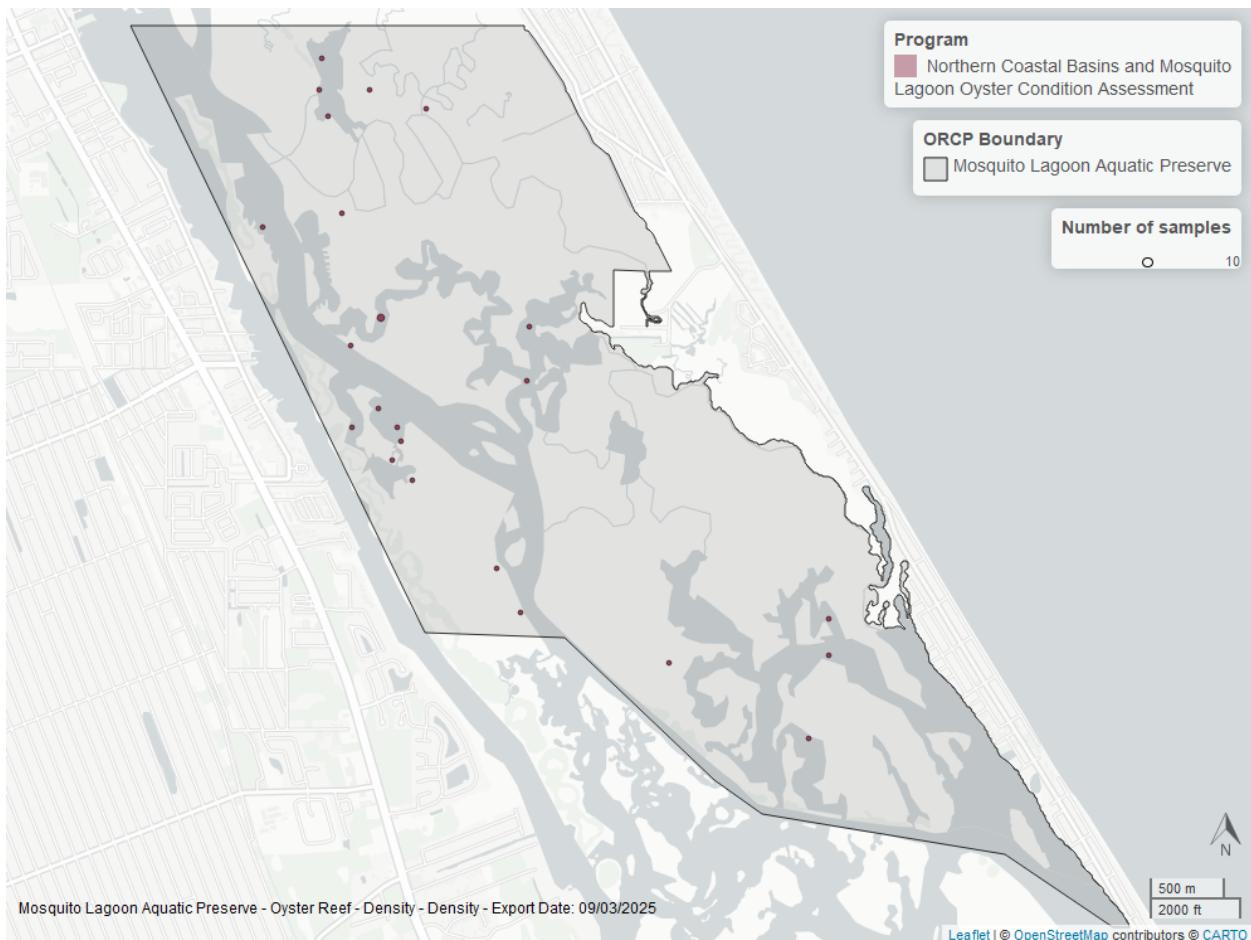


Figure 15: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 15: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Nassau River-St. Johns River Marshes Aquatic Preserve

Natural

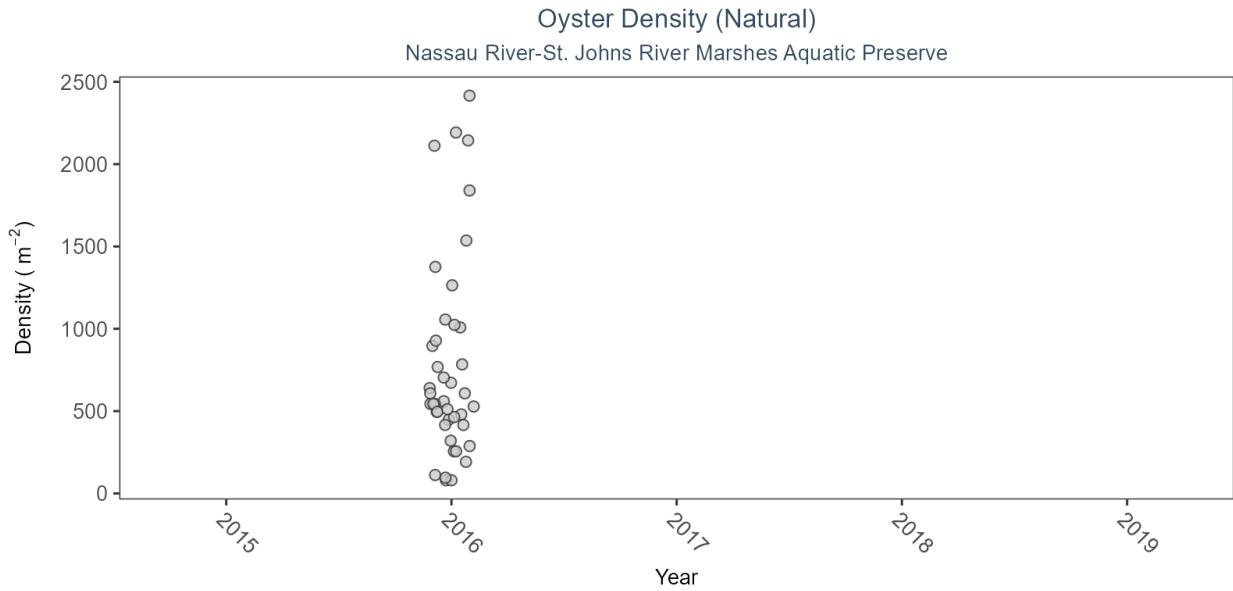
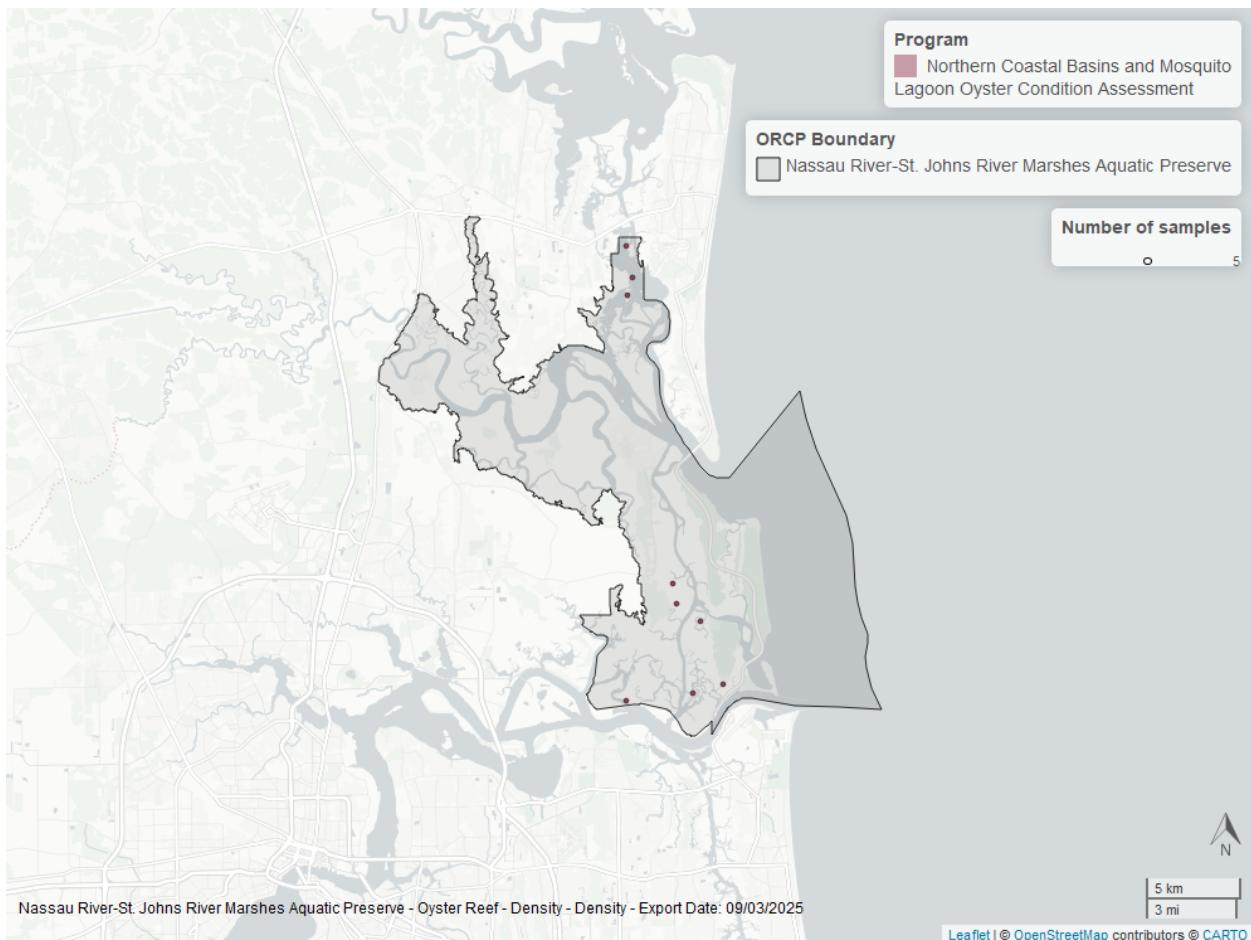


Figure 16: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 16: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Pine Island Sound Aquatic Preserve

Natural

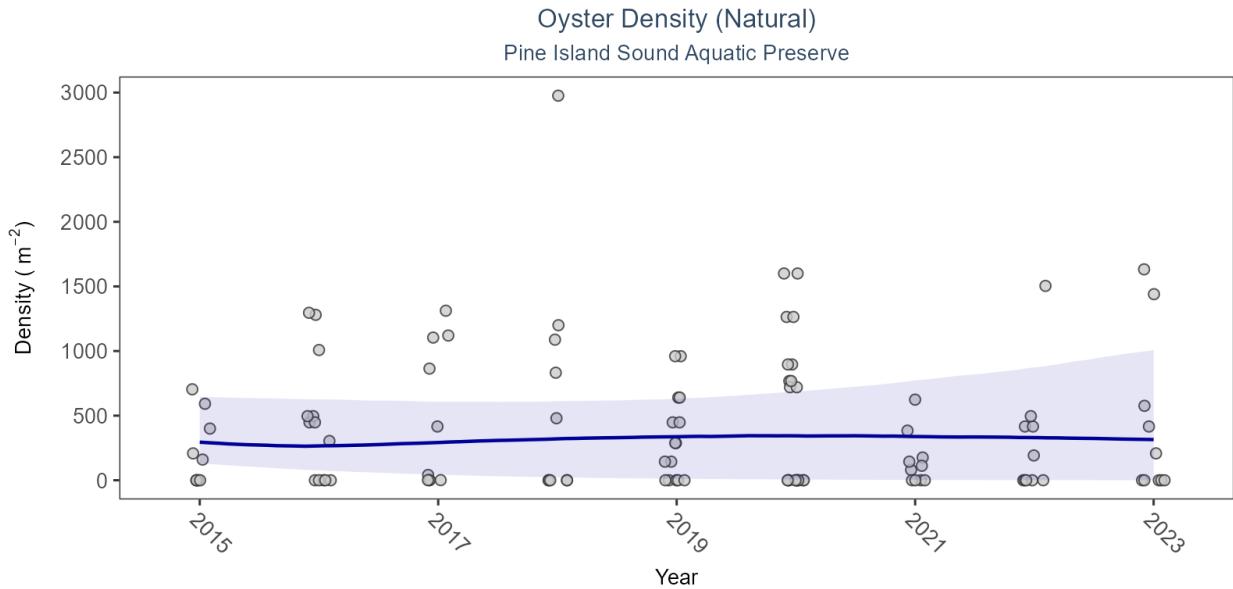


Figure 17: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 17: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-33.52	109.78	-16.09 to 40.37

For natural reefs, density decreased by an average of 5.15 oysters per square meter per year. For restored reefs, density decreased by an average of 214.22 oysters per square meter per year.

Restored

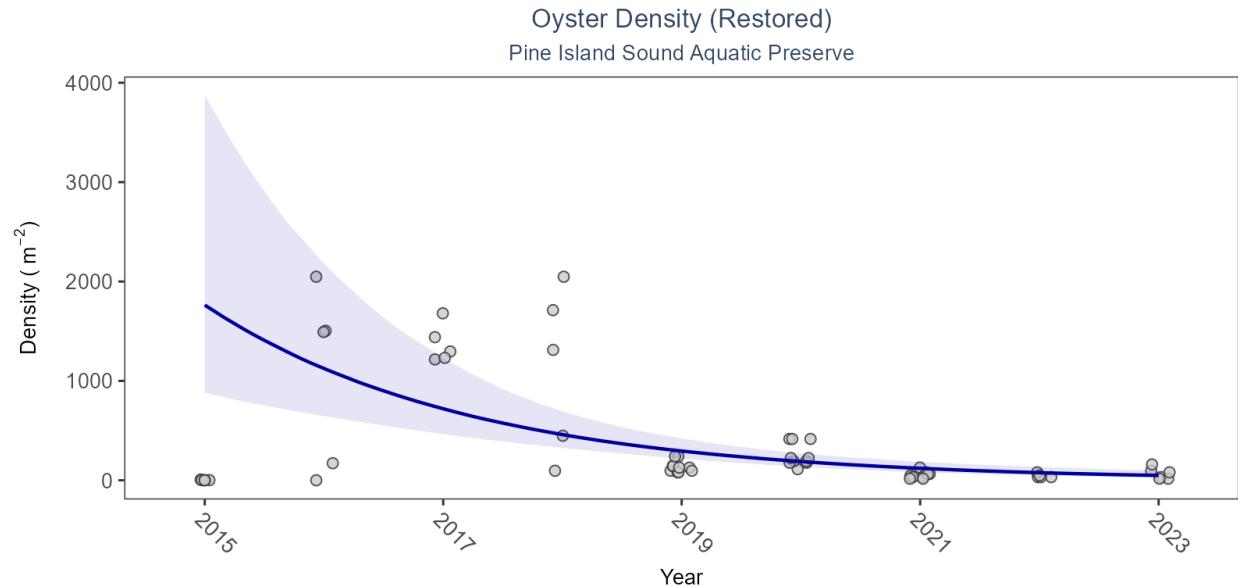
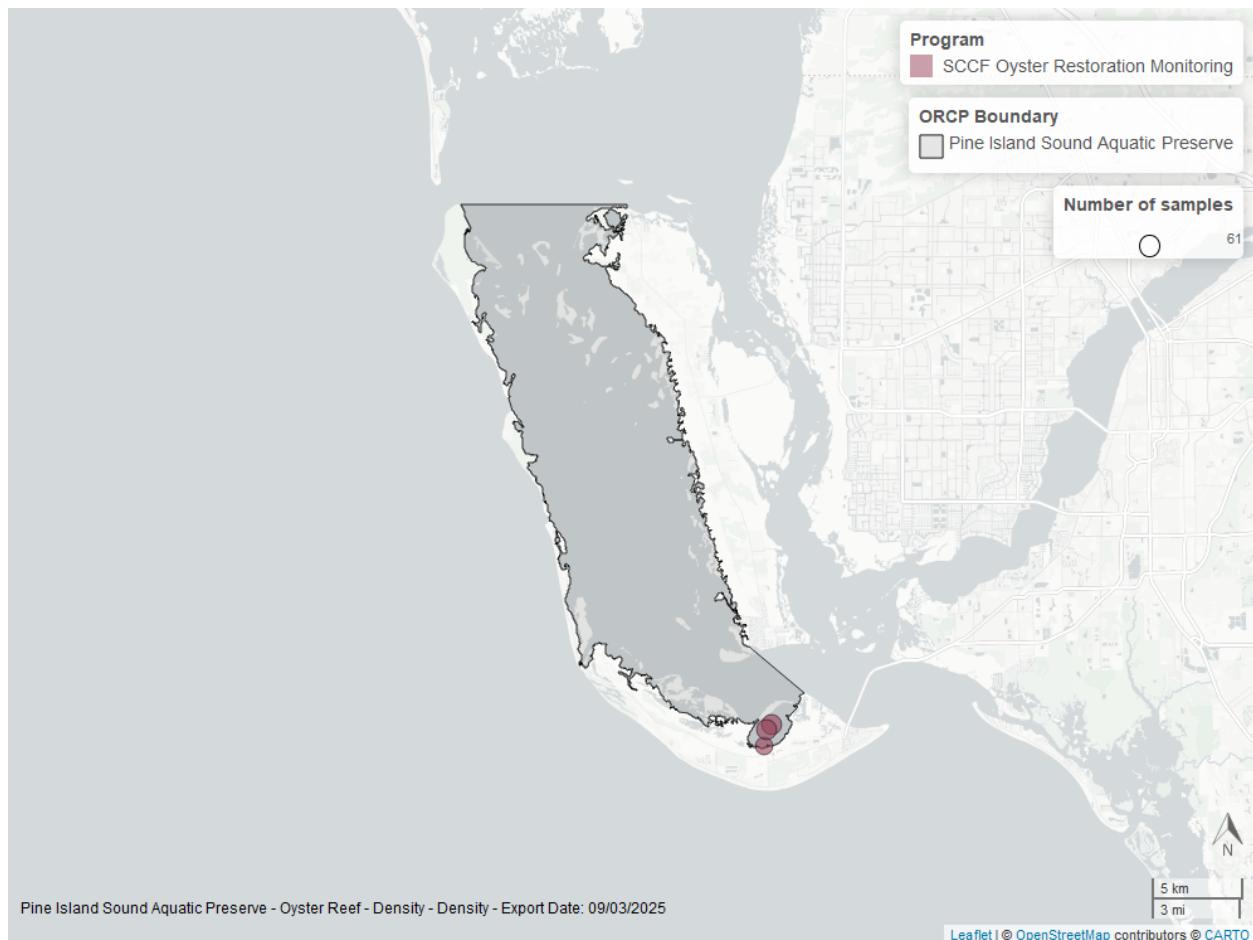


Figure 18: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 18: Model results for Oyster Density - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	No significant change	-214.22	128.39	-106.83 to -471.59

For natural reefs, density decreased by an average of 5.15 oysters per square meter per year. For restored reefs, density decreased by an average of 214.22 oysters per square meter per year.



Pinellas County Aquatic Preserve

Natural

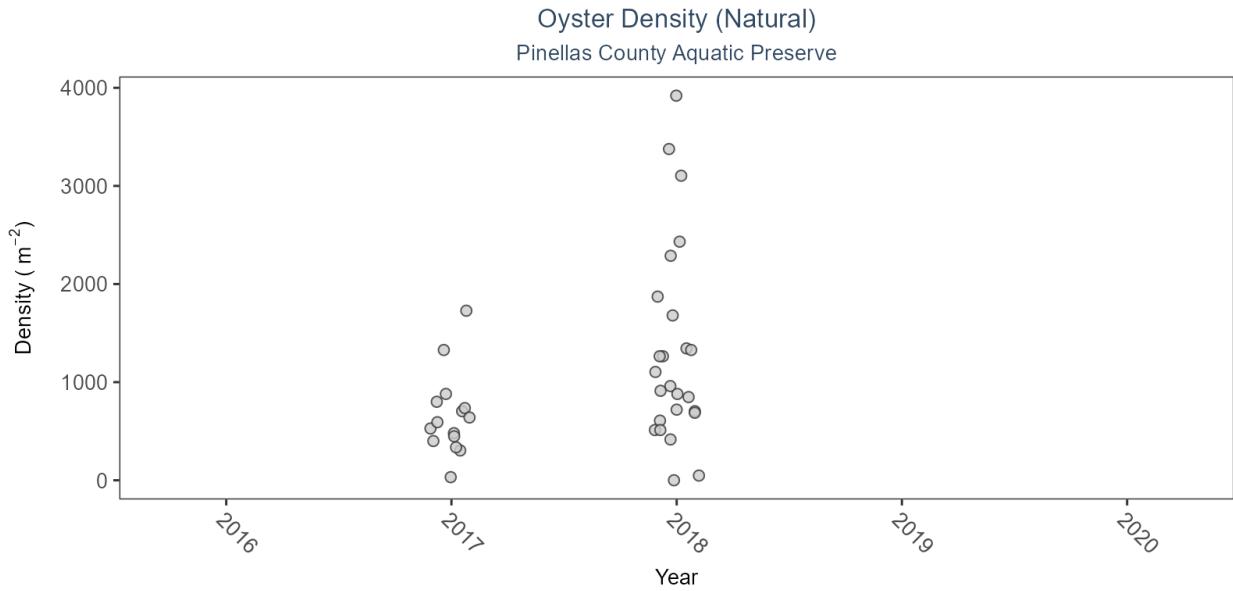
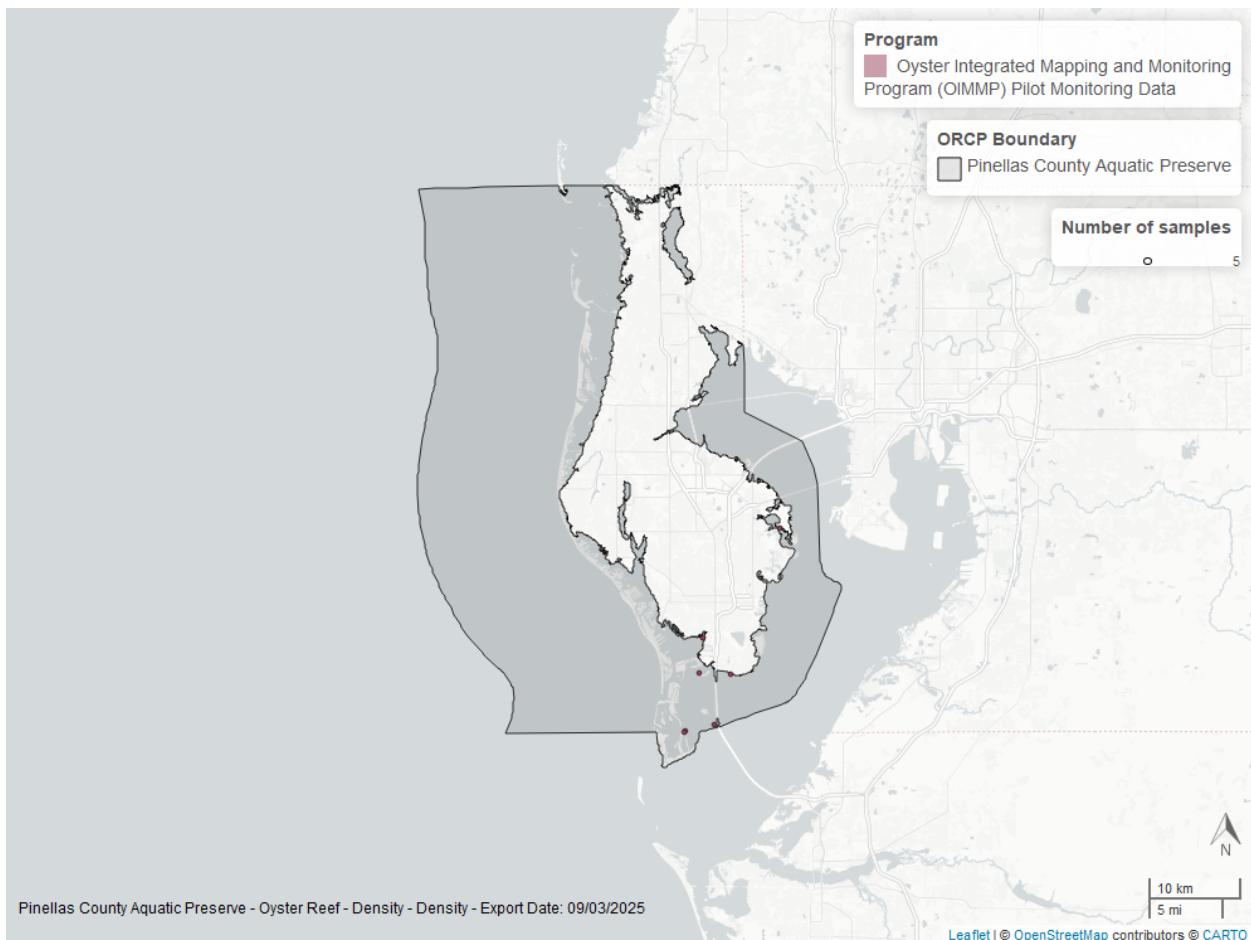


Figure 19: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 19: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



St. Martins Marsh Aquatic Preserve

Natural

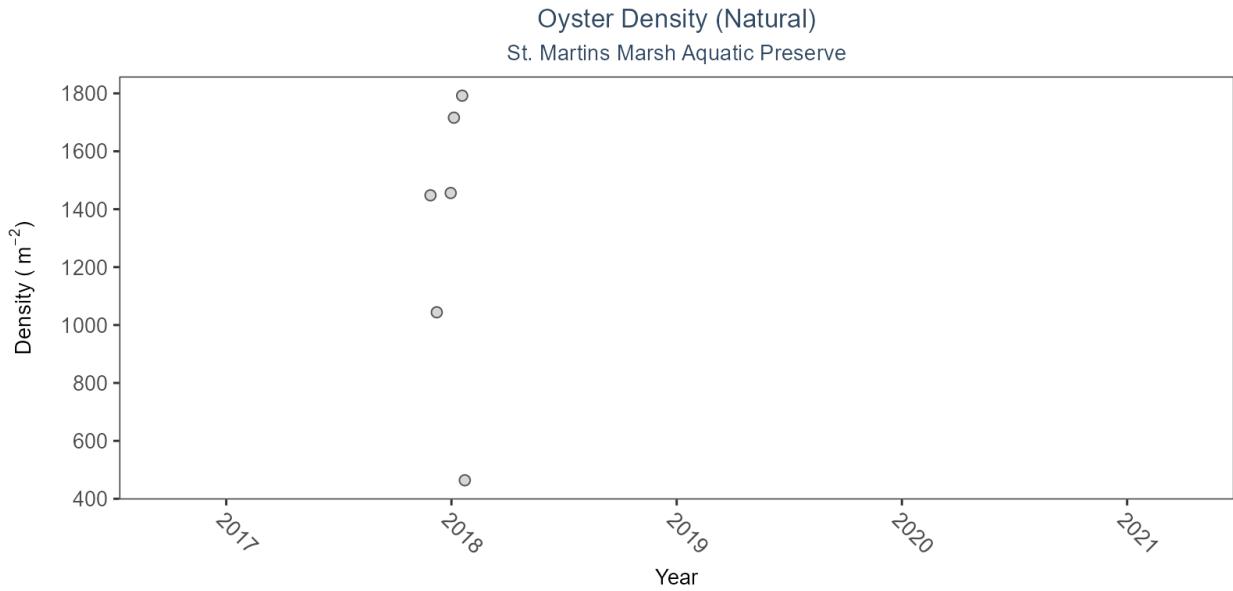
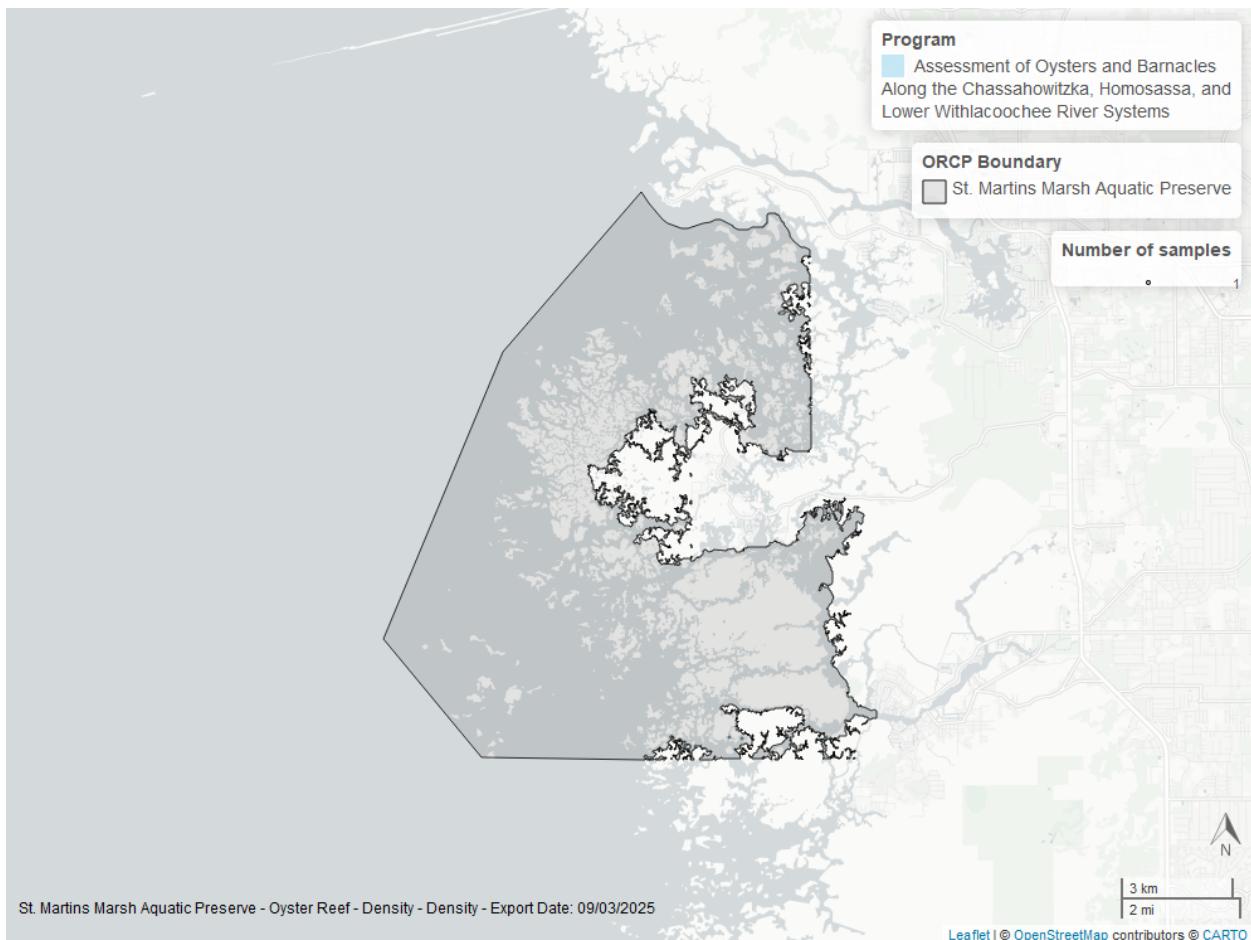


Figure 20: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 20: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Tomoka Marsh Aquatic Preserve

Natural

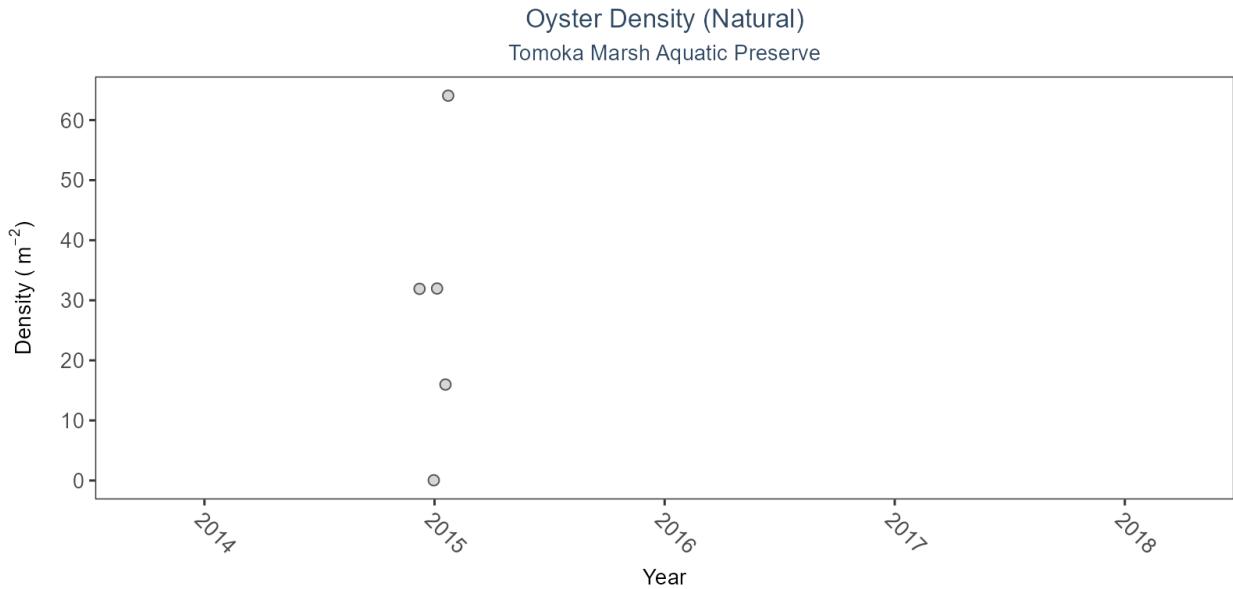
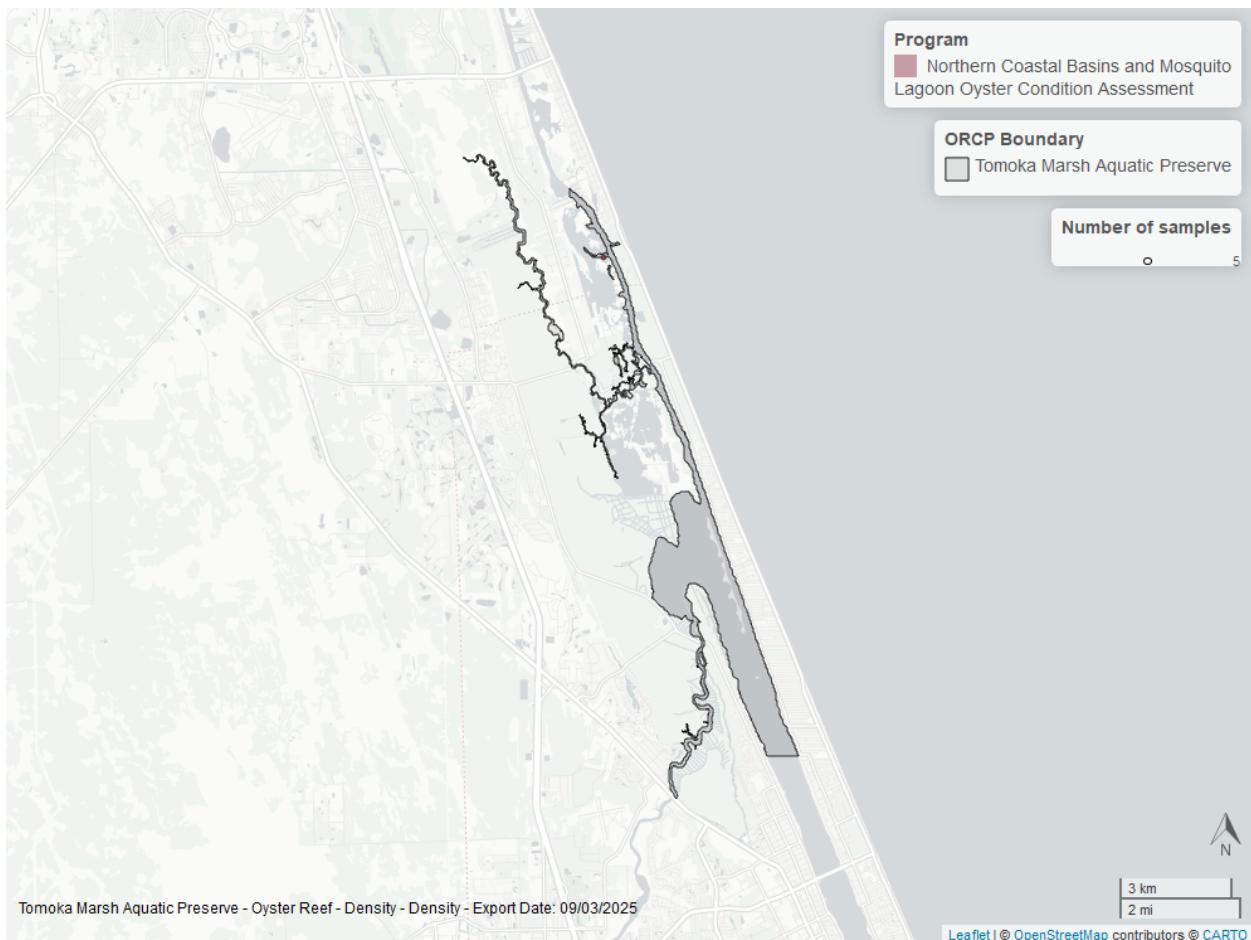


Figure 21: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 21: Model results for Oyster Density - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density.



Yellow River Marsh Aquatic Preserve

Natural

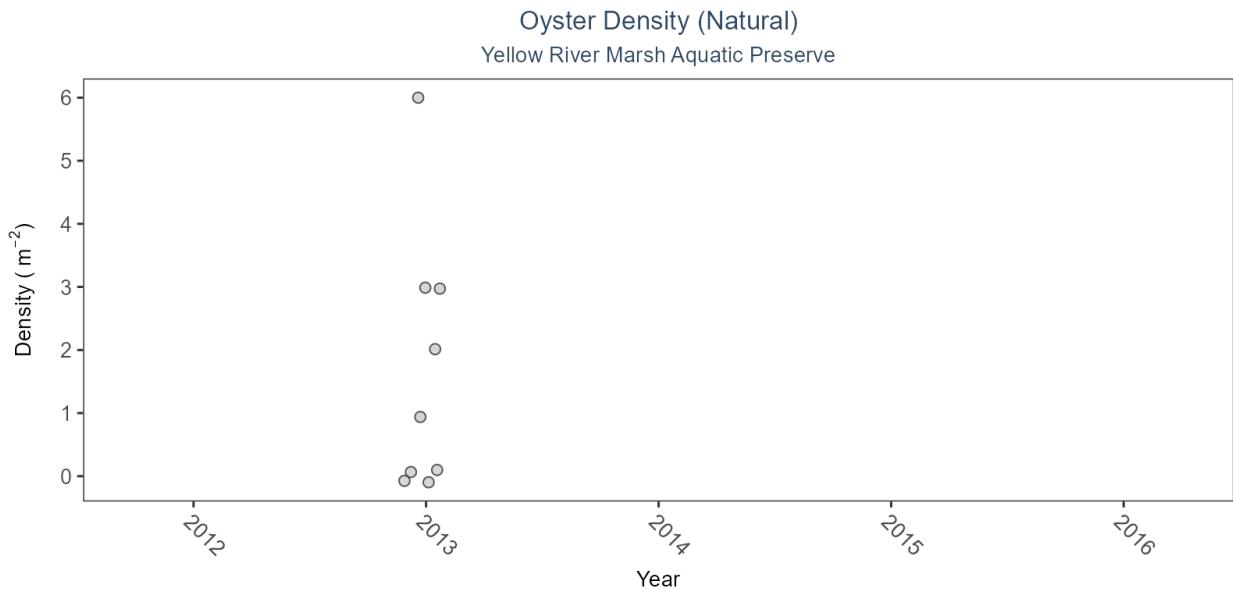


Figure 22: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 22: Model results for Oyster Density - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for density. For restored reefs, density increased by an average of 9.92 oysters per square meter per year.

Restored

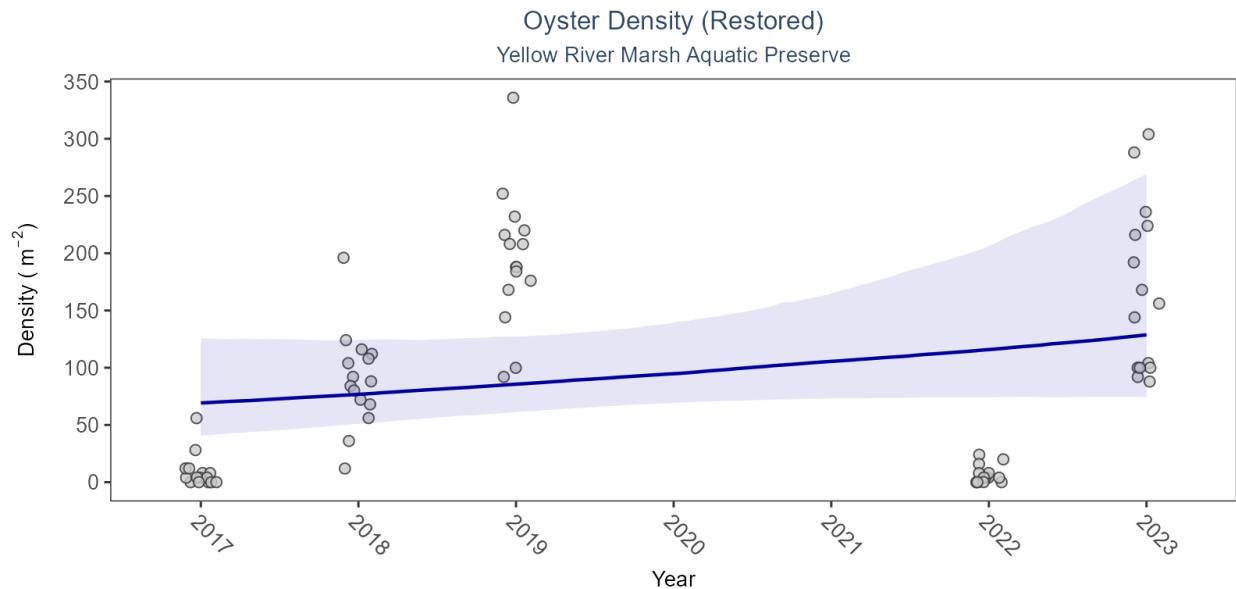
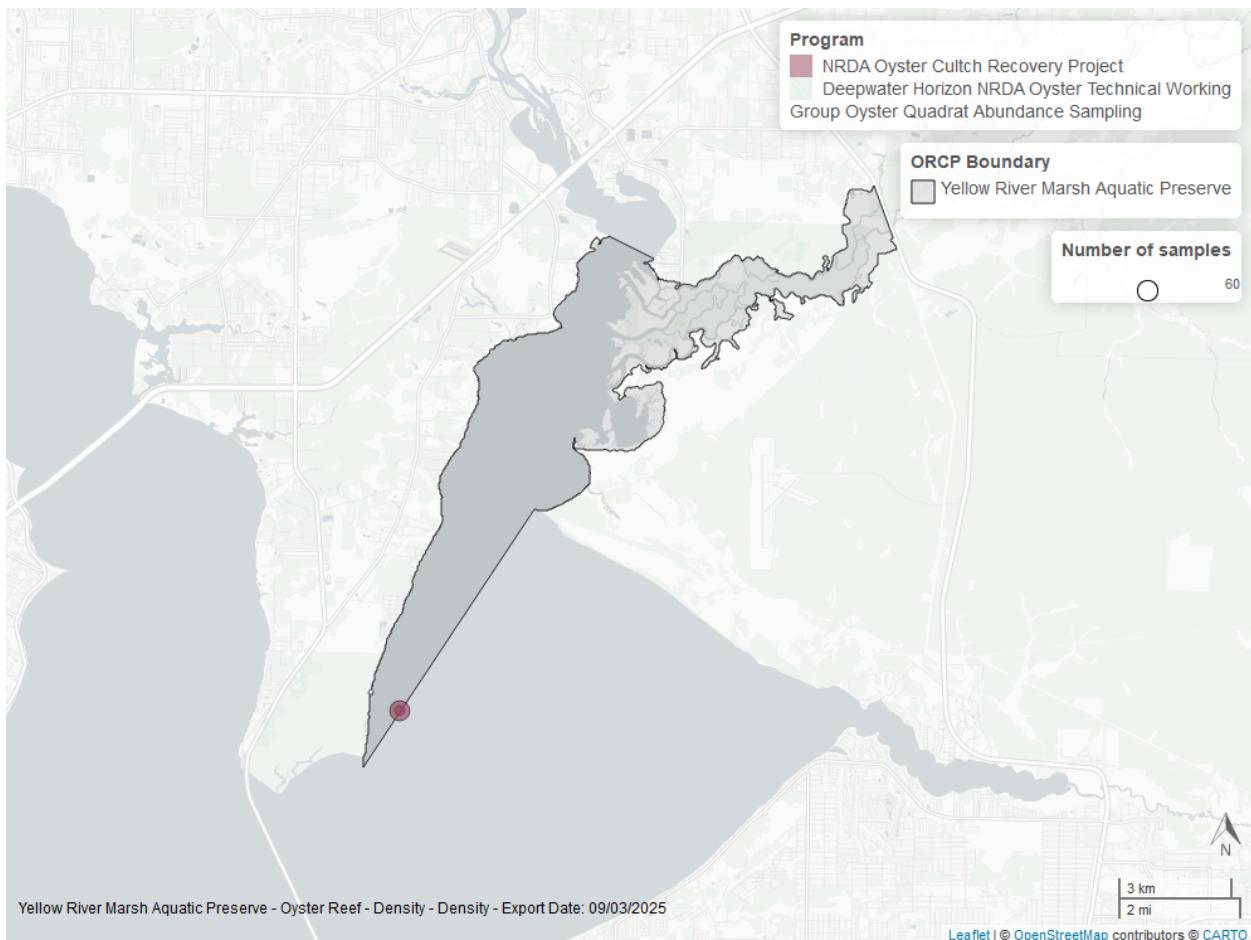


Figure 23: Scatter plot of oyster density over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 23: Model results for Oyster Density - Restored

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Restored	No significant change	9.92	21.37	5.64 to 23.87

For natural reefs, there was insufficient data to calculate a trend for density. For restored reefs, density increased by an average of 9.92 oysters per square meter per year.



Percent Live

Apalachicola Bay Aquatic Preserve

Natural

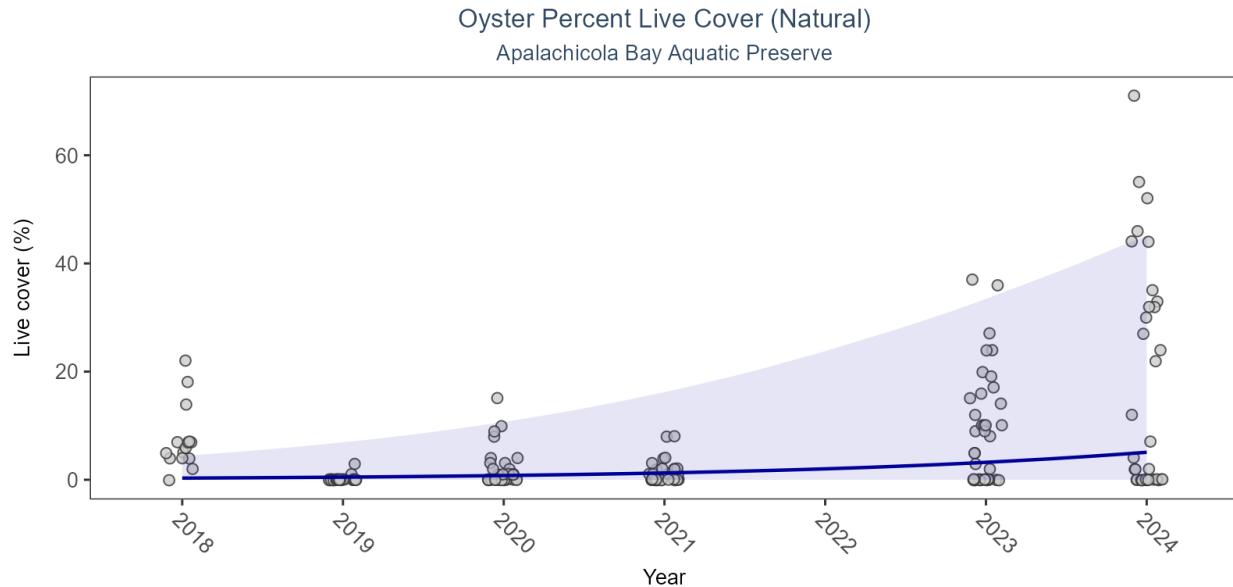


Figure 24: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 24: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	0.87	2.72	0 to 6.81

For natural reefs, percent live cover increased by an average of 0.78% per year. For restored reefs, percent live cover decreased by an average of 2.14% per year.

Restored

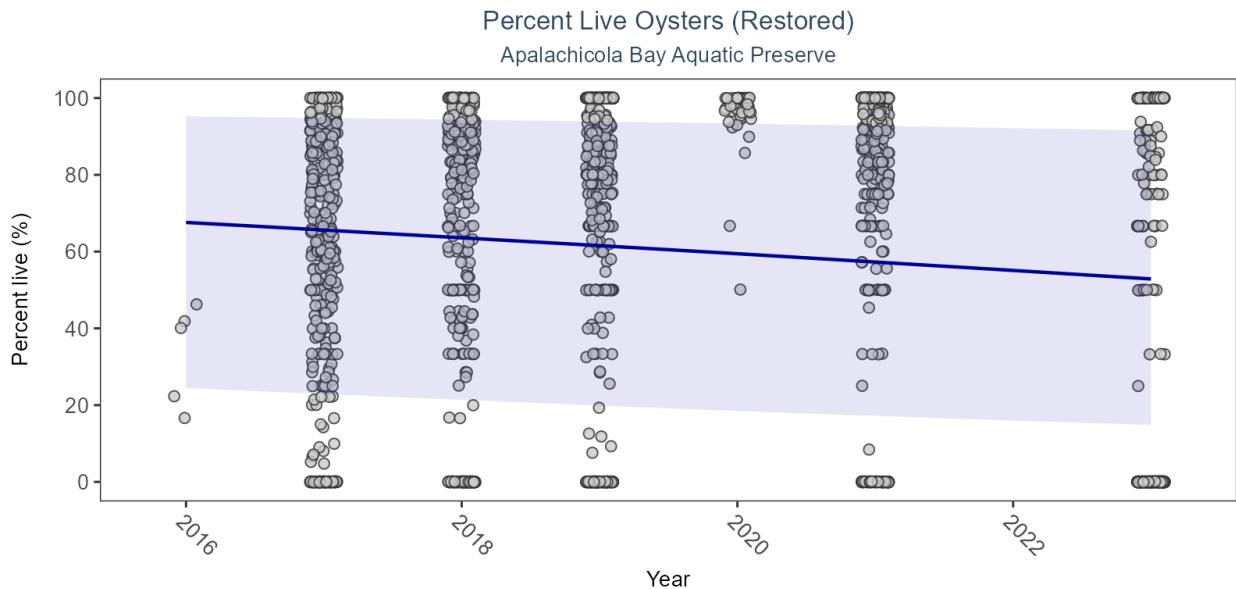
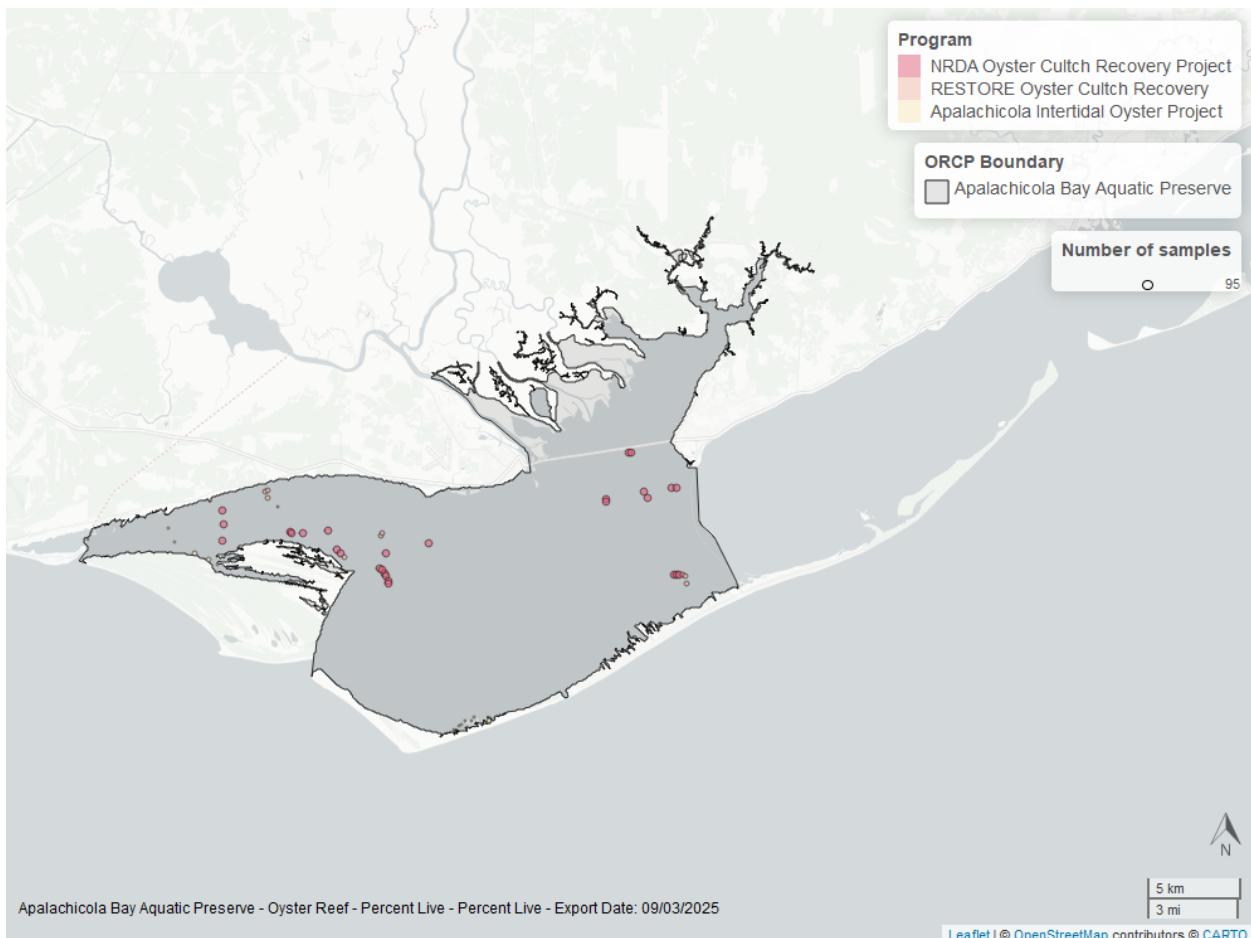


Figure 25: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 25: Model results for Oyster Percent Live - Restored

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Restored	No significant change	-2.15	36.27	-1.35 to -0.51

For natural reefs, percent live cover increased by an average of 0.78% per year. For restored reefs, percent live cover decreased by an average of 2.14% per year.



Apalachicola National Estuarine Research Reserve

Natural

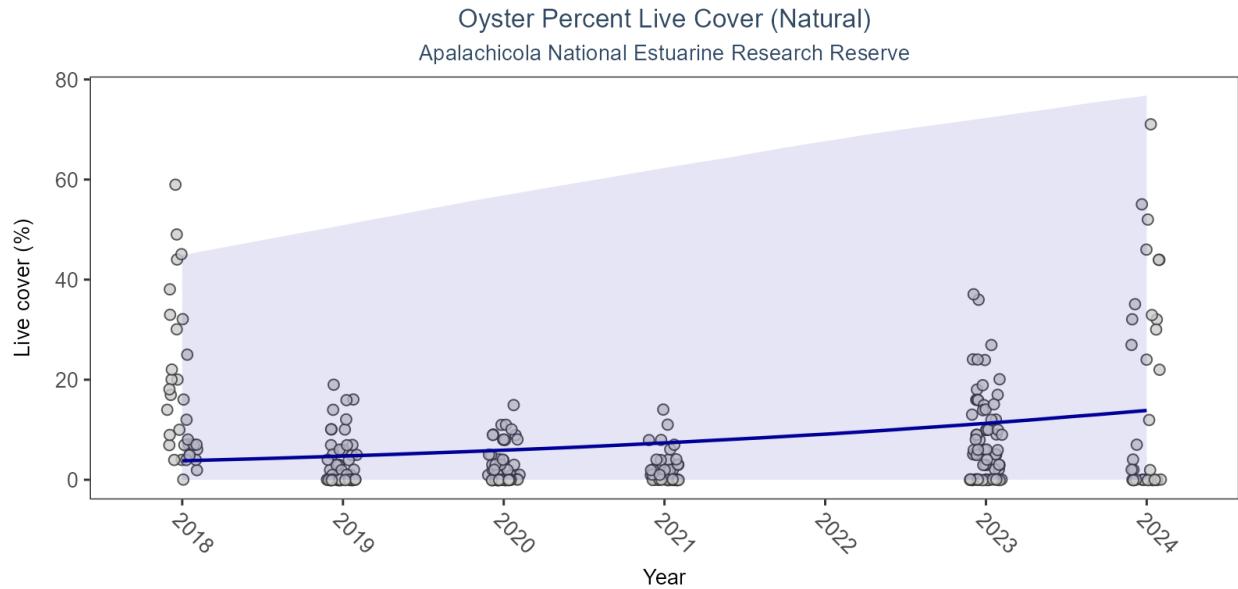


Figure 26: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 26: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	1.69	9.76	0 to 5.27

For natural reefs, percent live cover increased by an average of 1.71% per year. For restored reefs, percent live cover decreased by an average of 1.53% per year.

Restored

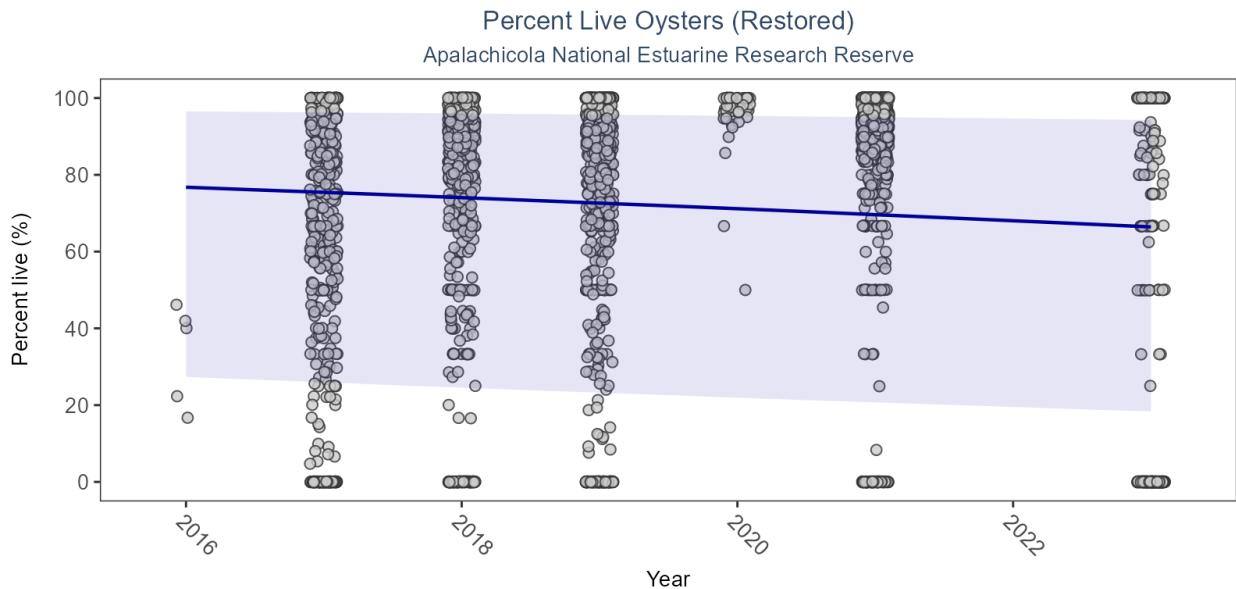
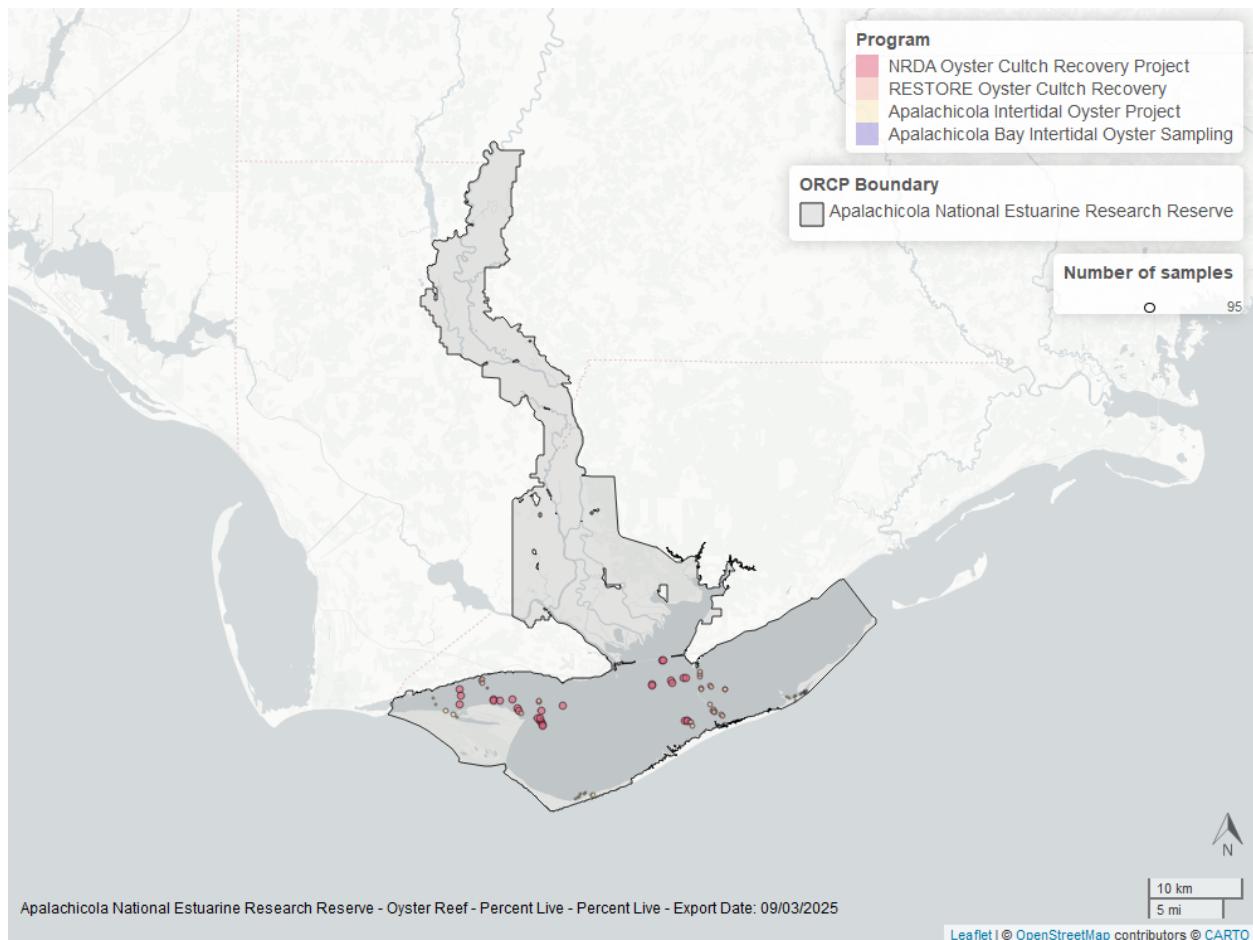


Figure 27: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 27: Model results for Oyster Percent Live - Restored

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Restored	No significant change	-1.54	31.6	-1.15 to -0.36

For natural reefs, percent live cover increased by an average of 1.71% per year. For restored reefs, percent live cover decreased by an average of 1.53% per year.



Boca Ciega Bay Aquatic Preserve

Natural

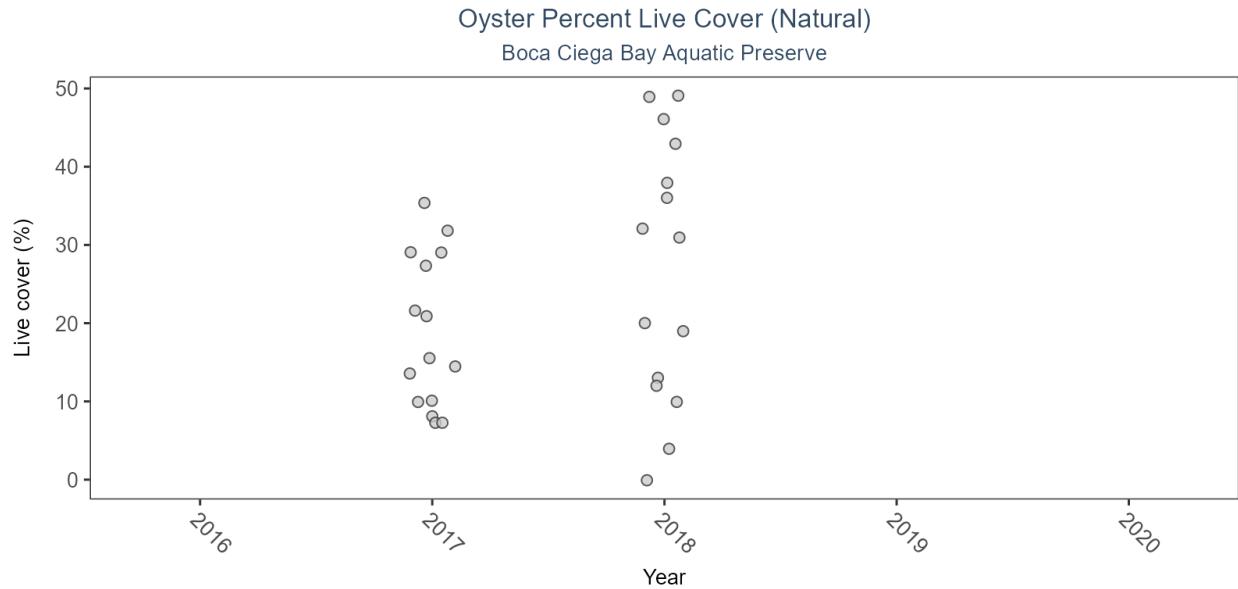
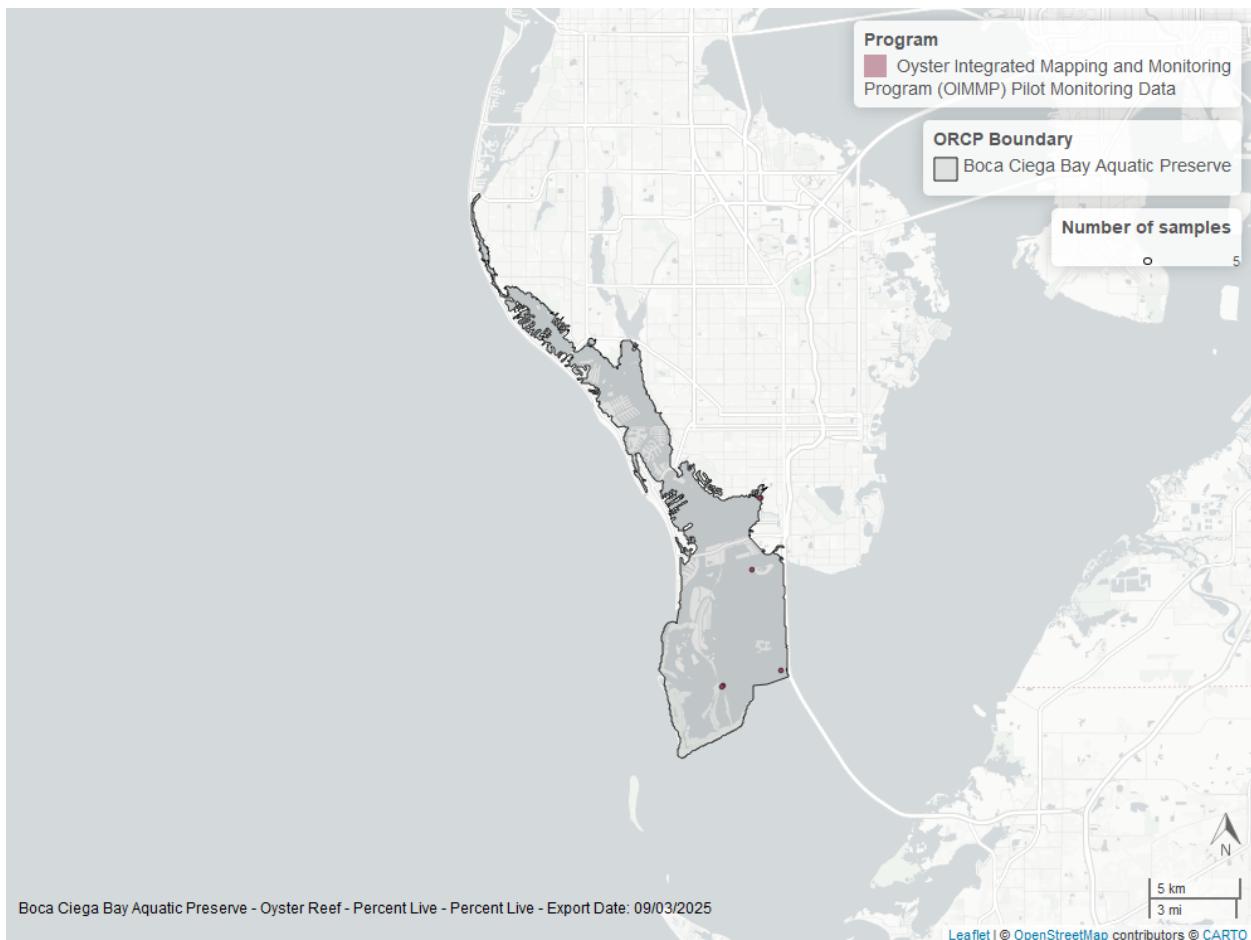


Figure 28: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 28: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Estero Bay Aquatic Preserve

Restored

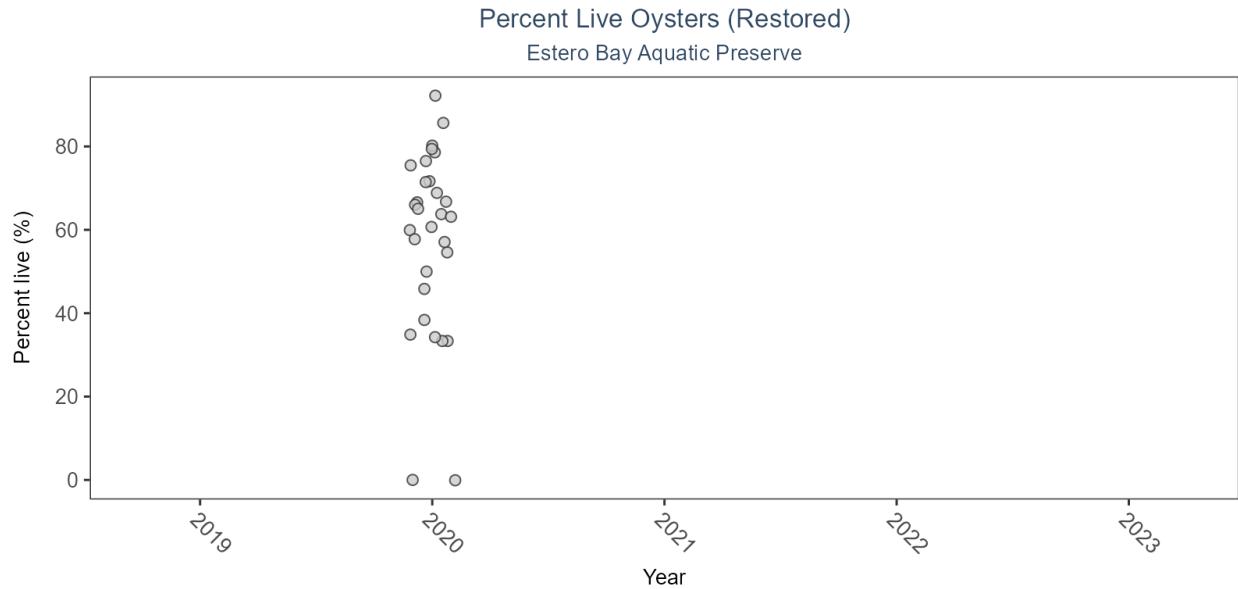


Figure 29: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 29: Model results for Oyster Percent Live - Restored

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Restored	-	-	-	-

For natural reefs, percent live cover decreased by an average of 0.59% per year. For restored reefs, there was insufficient data to calculate a trend for percent live cover.

Natural

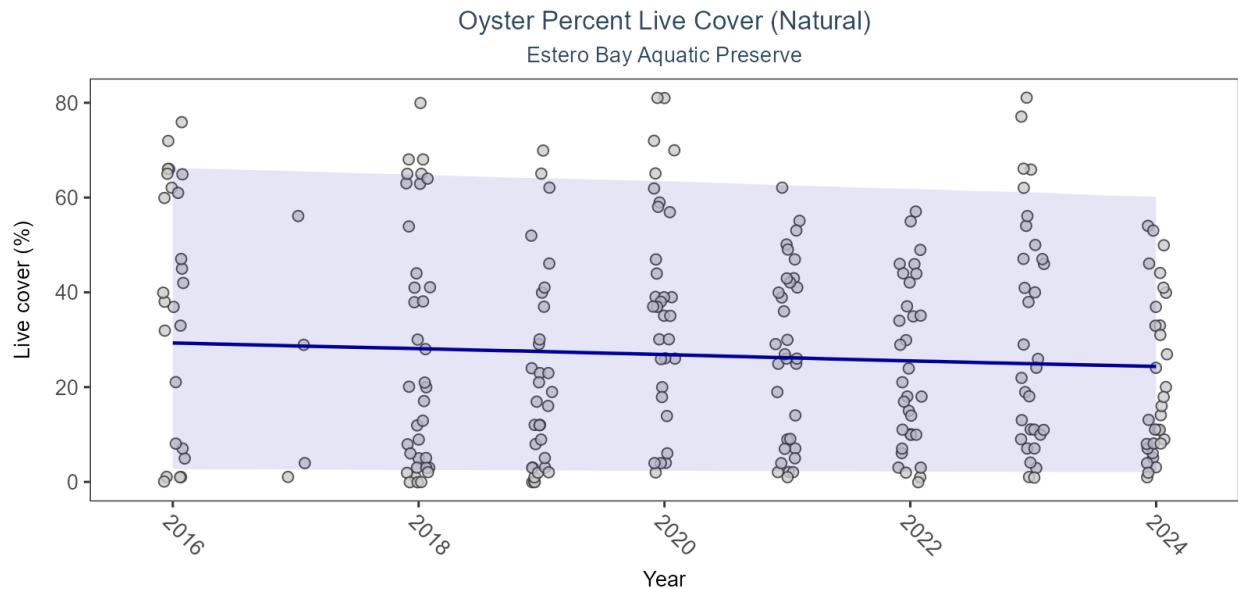
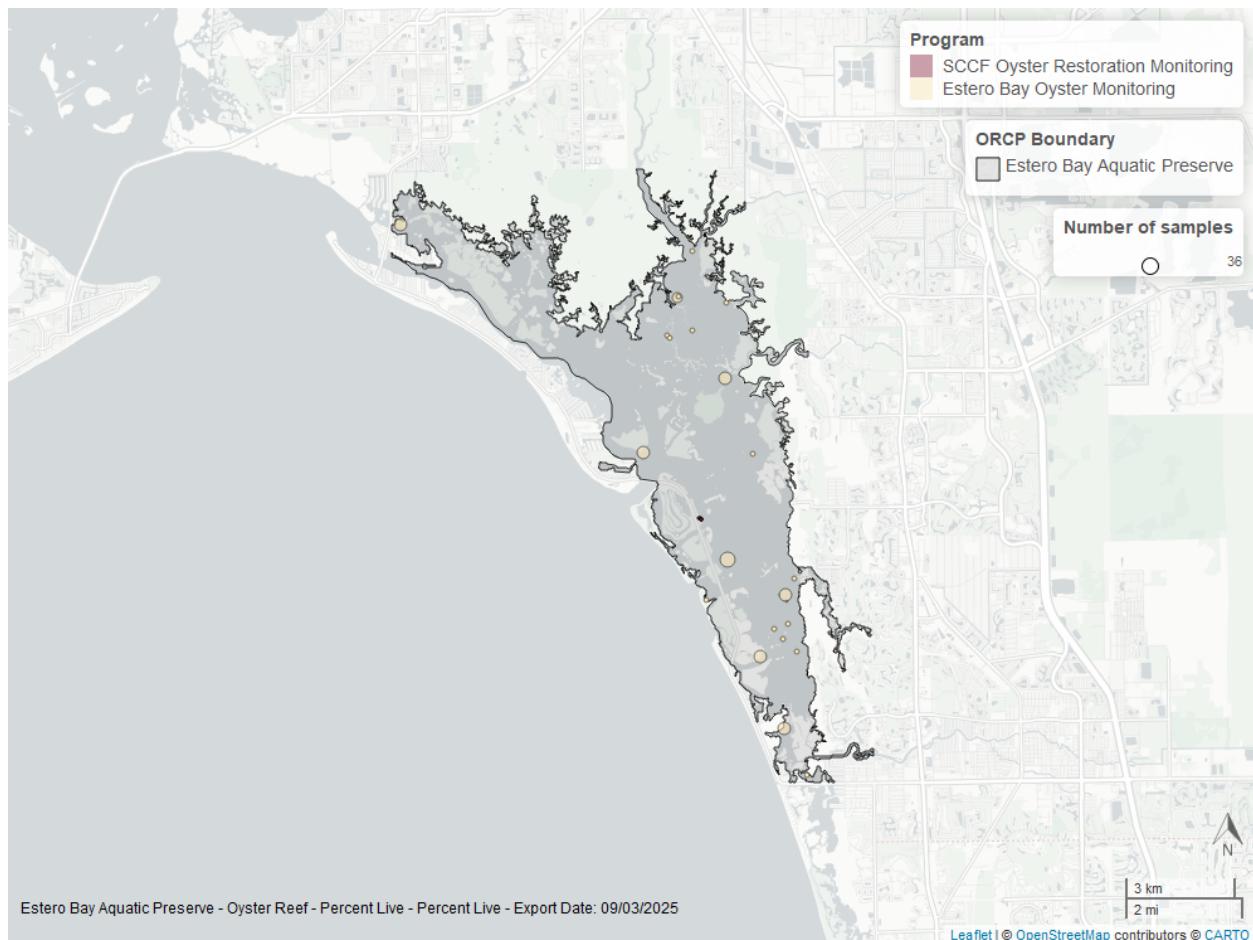


Figure 30: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 30: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-0.6	18	-0.08 to -0.76

For natural reefs, percent live cover decreased by an average of 0.59% per year. For restored reefs, there was insufficient data to calculate a trend for percent live cover.



Guana River Marsh Aquatic Preserve

Natural

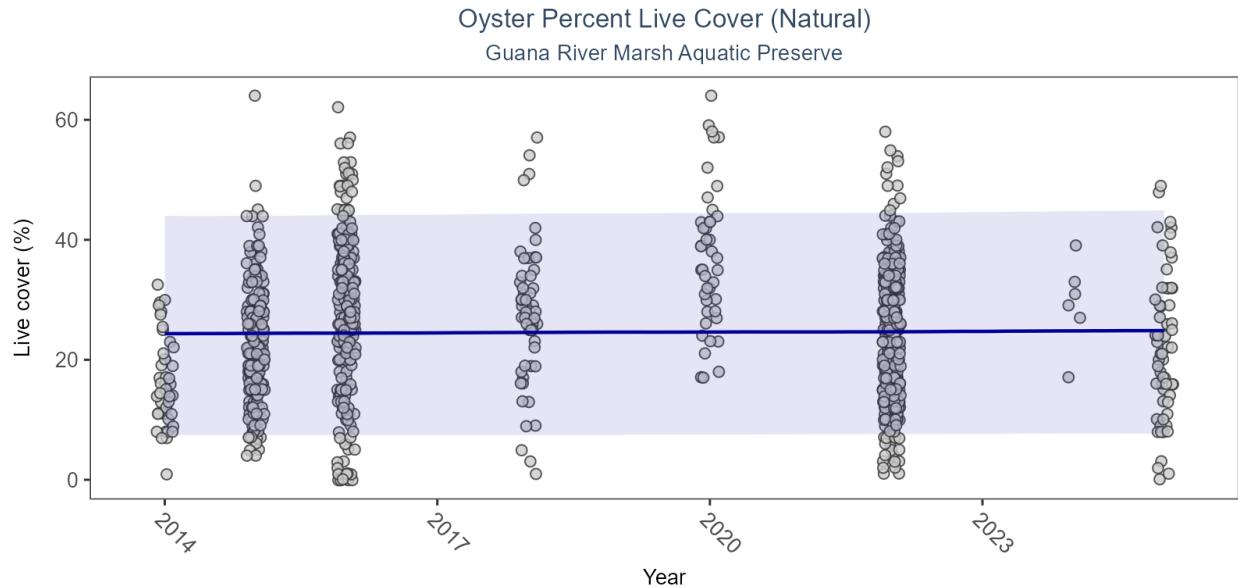
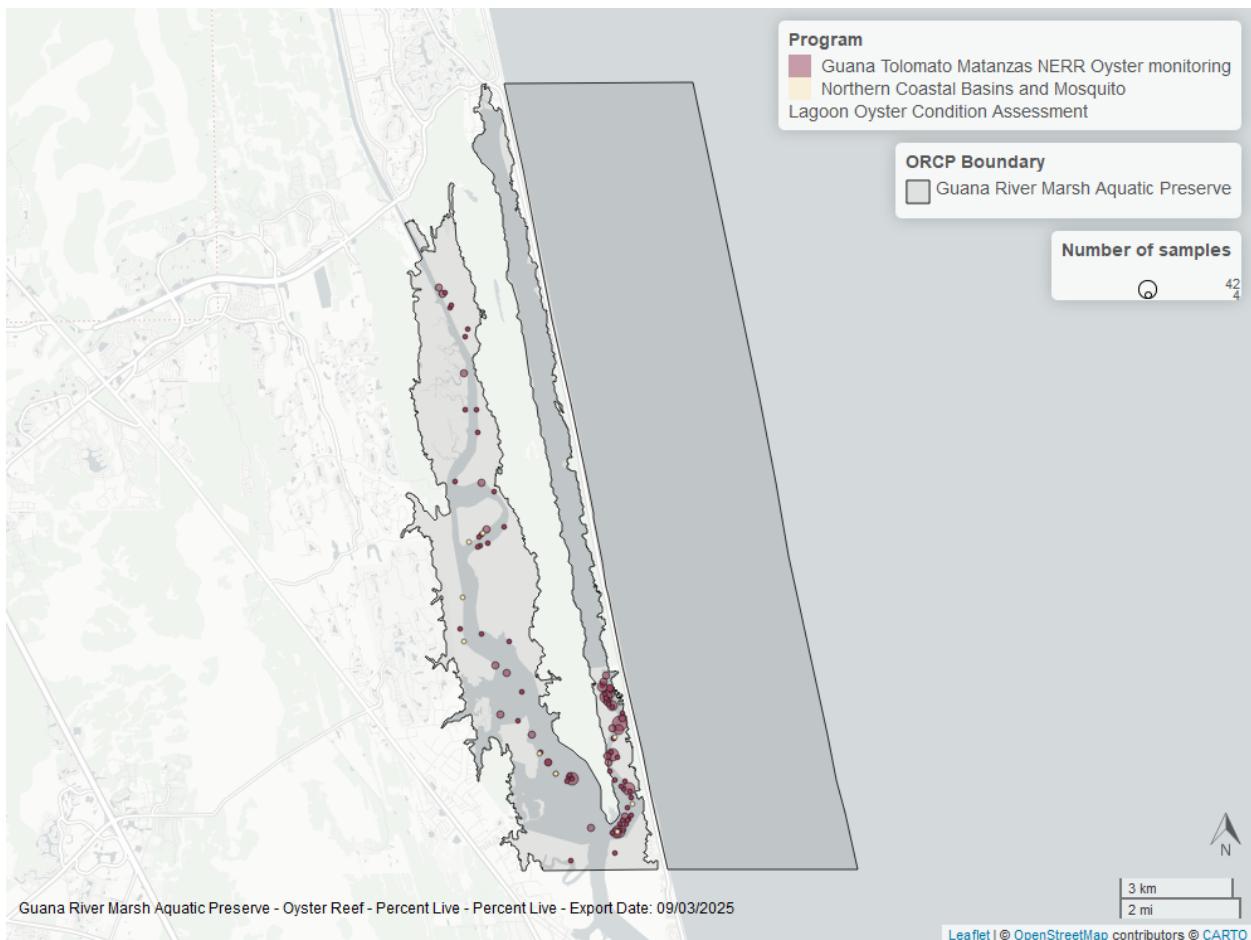


Figure 31: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 31: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	0.04	9.65	0.02 to 0.08

For natural reefs, percent live cover increased by an average of 0.05% per year.



Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

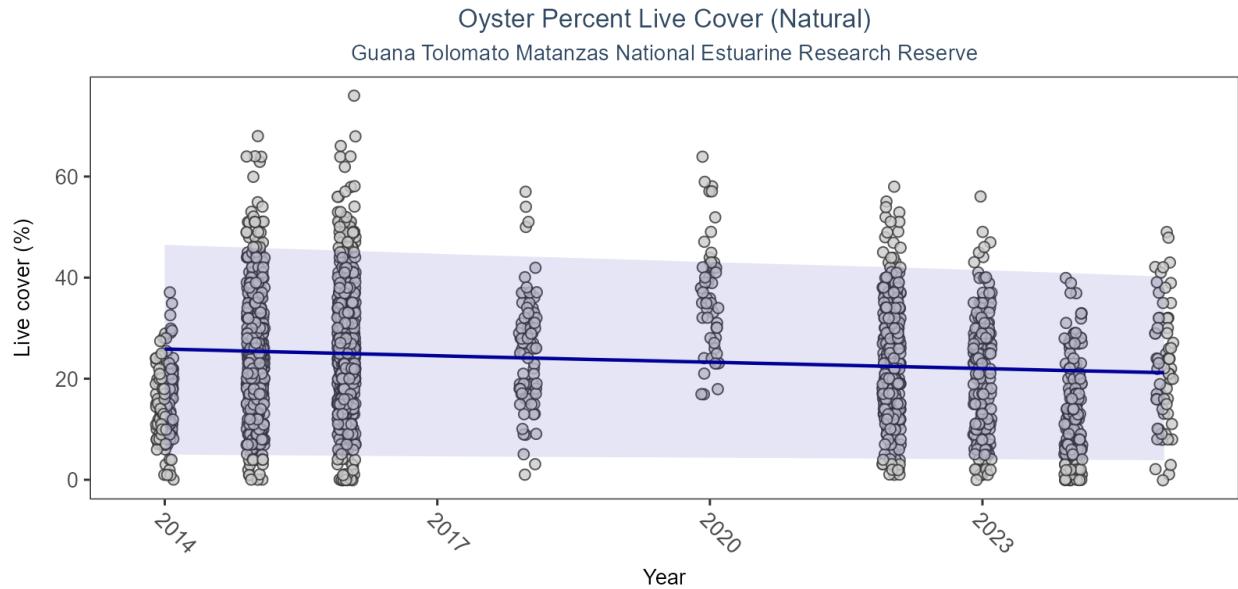
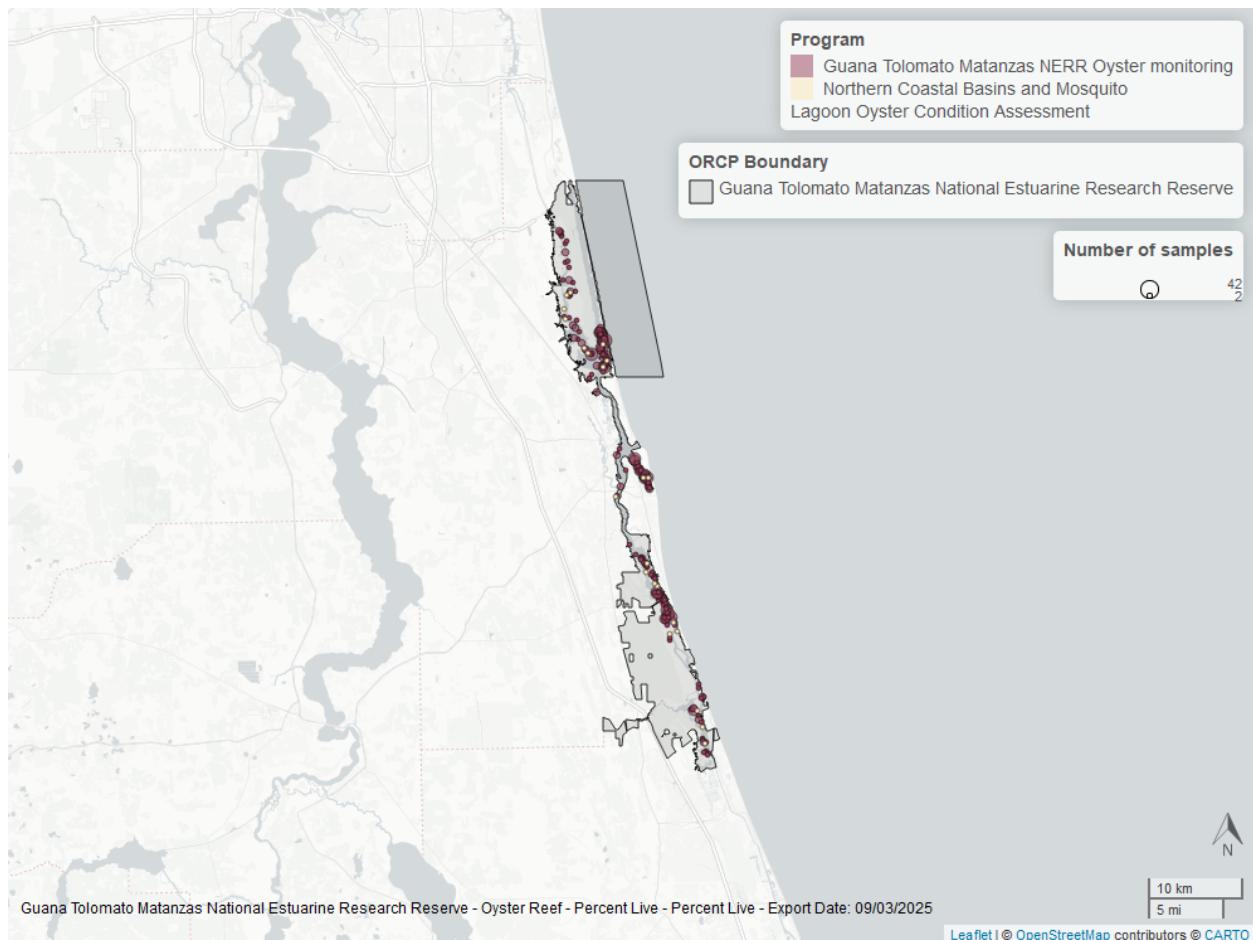


Figure 32: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 32: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-0.44	10.14	-0.1 to -0.6

For natural reefs, percent live cover decreased by an average of 0.44% per year.



Indian River-Malabar to Vero Beach Aquatic Preserve

Natural

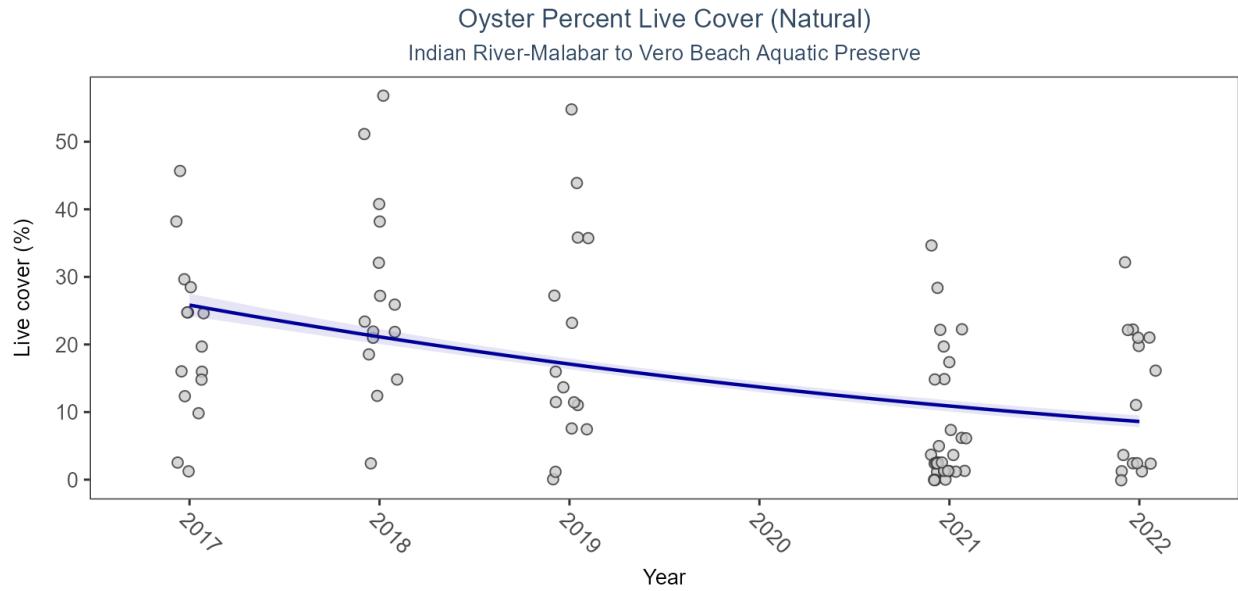
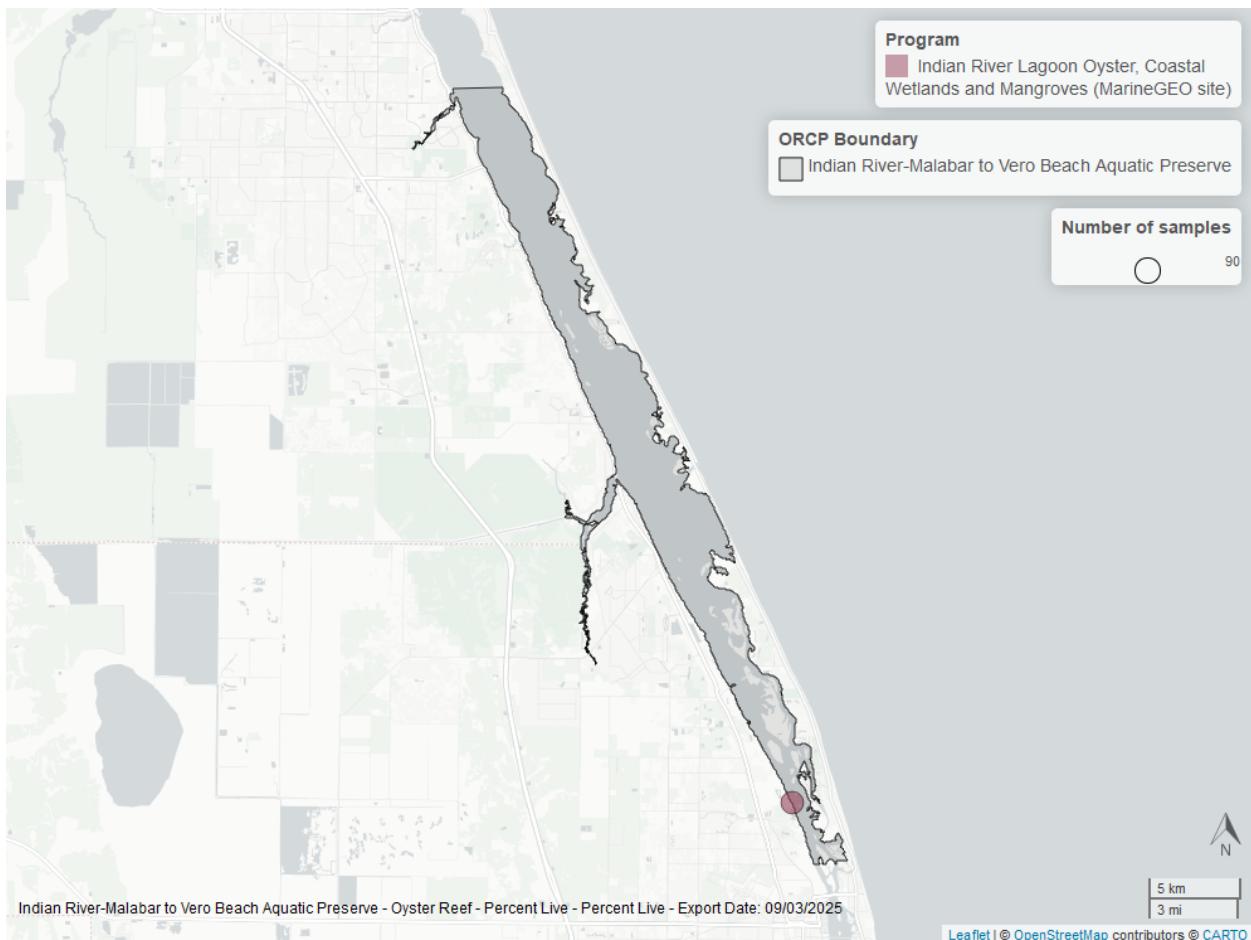


Figure 33: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 33: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-3.45	0.48	-3.29 to -3.61

For natural reefs, percent live cover decreased by an average of 3.45% per year.



Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

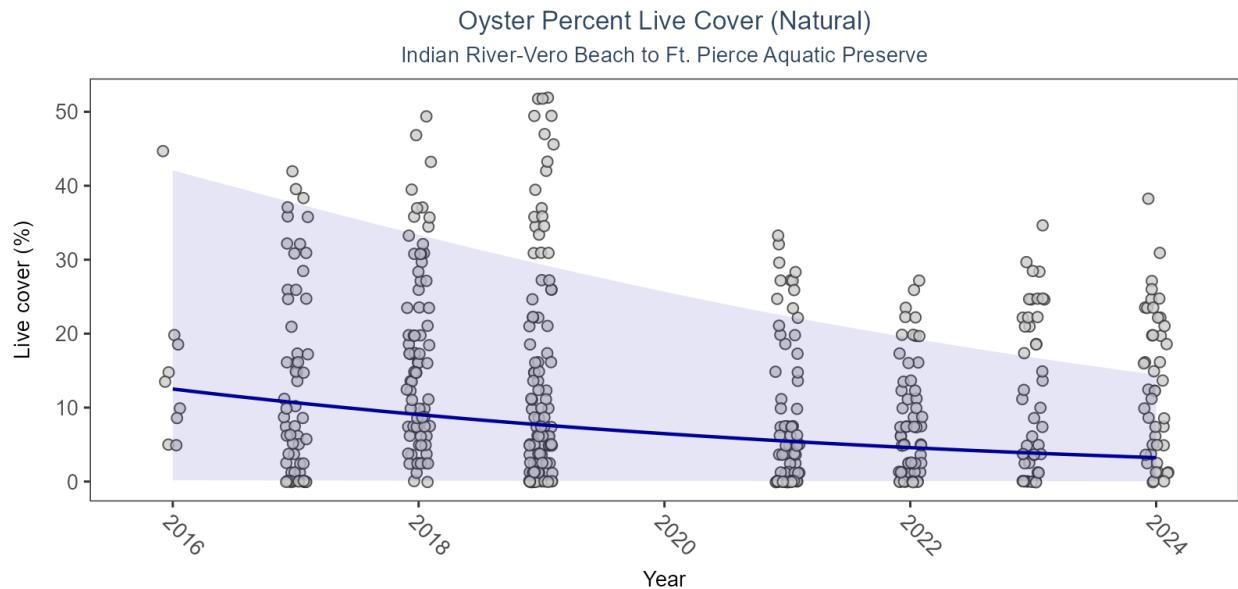
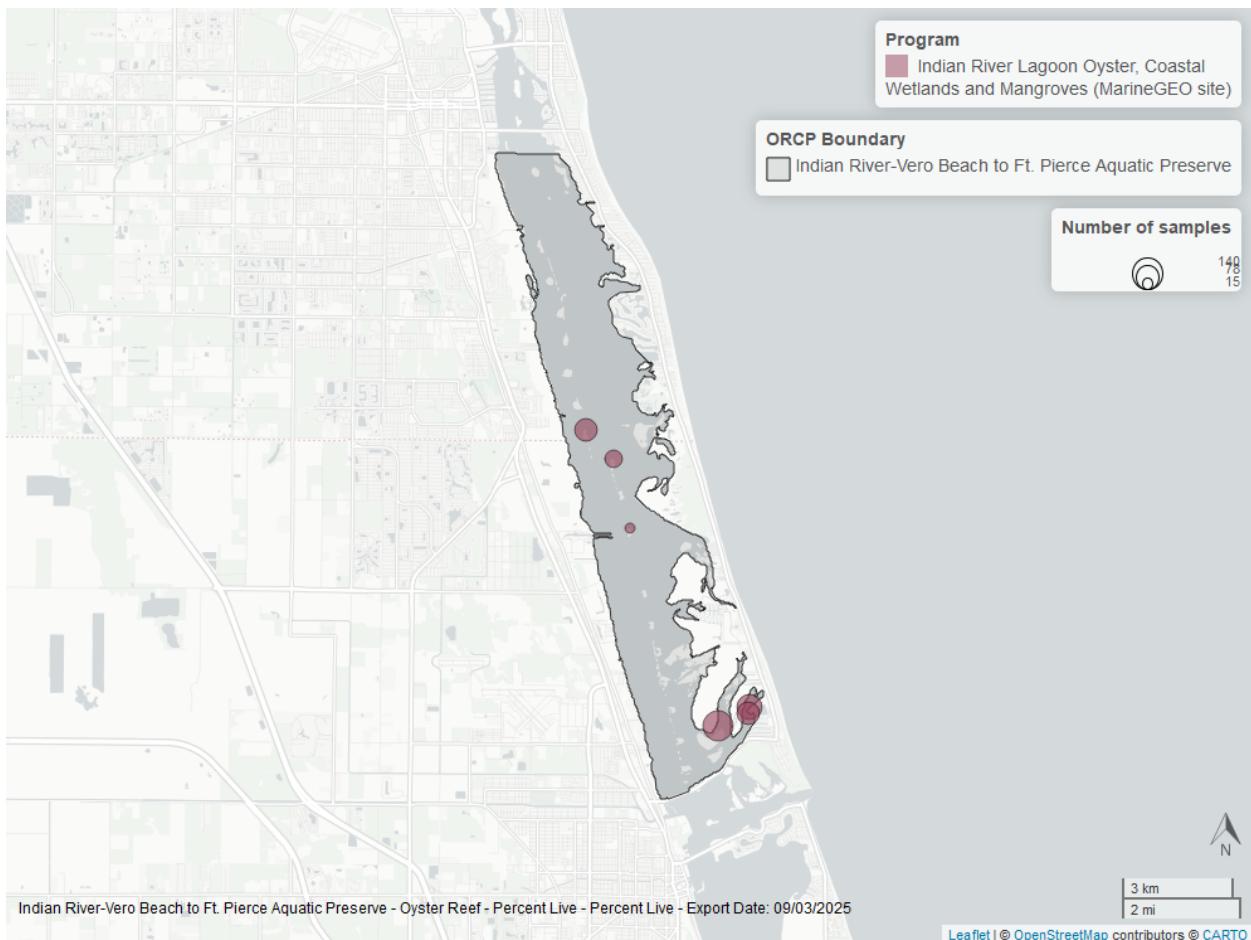


Figure 34: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 34: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	-1.16	9.92	-0.01 to -3.46

For natural reefs, percent live cover decreased by an average of 1.17% per year.



Jensen Beach to Jupiter Inlet Aquatic Preserve

Natural

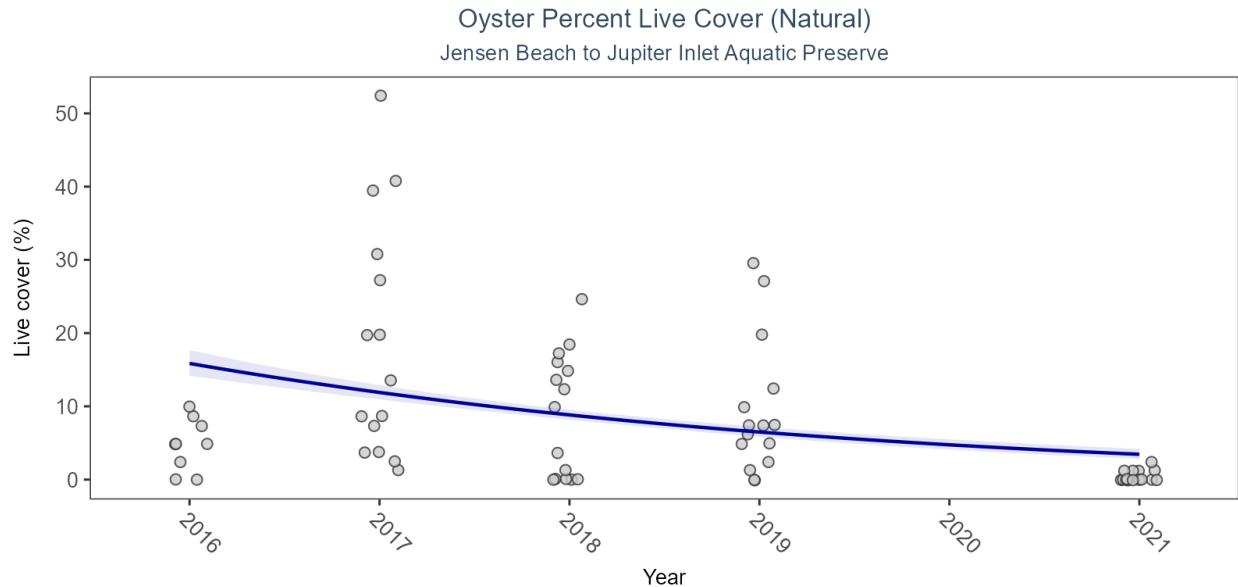
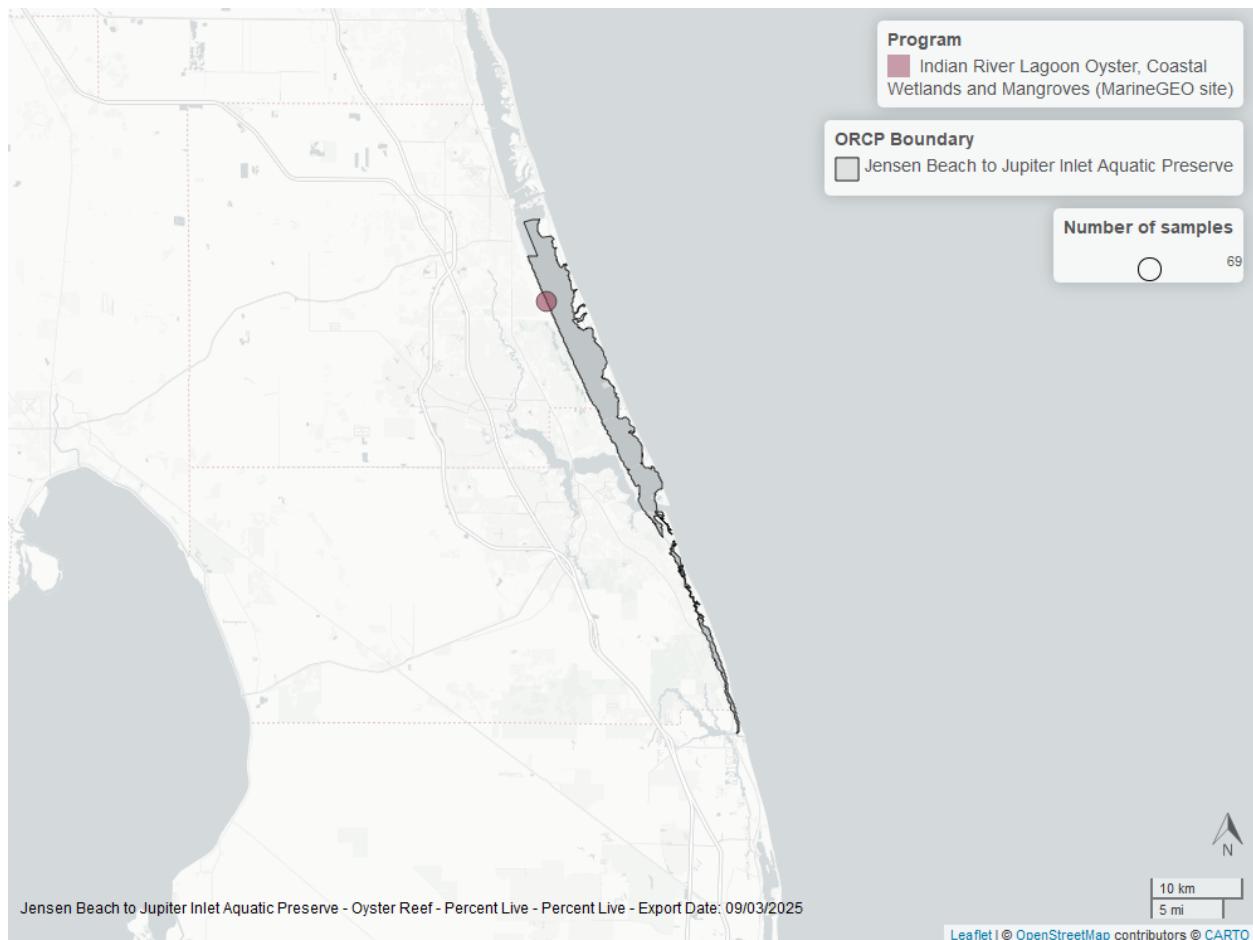


Figure 35: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 35: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	-2.48	0.43	-2.27 to -2.69

For natural reefs, percent live cover decreased by an average of 2.48% per year.



Lemon Bay Aquatic Preserve

Natural

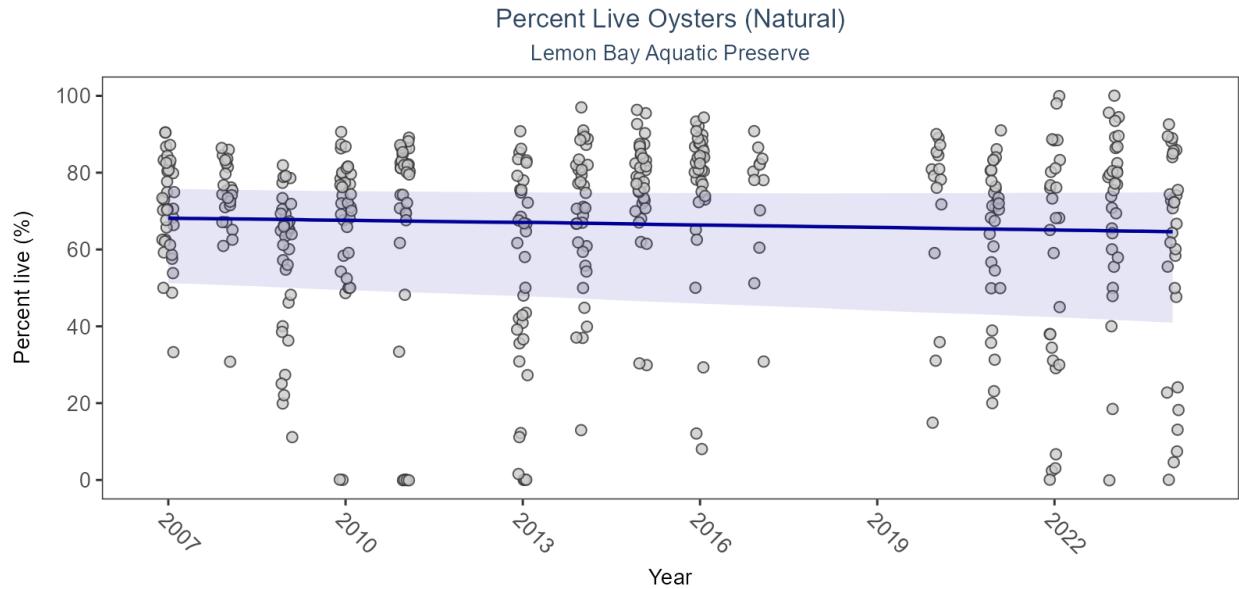
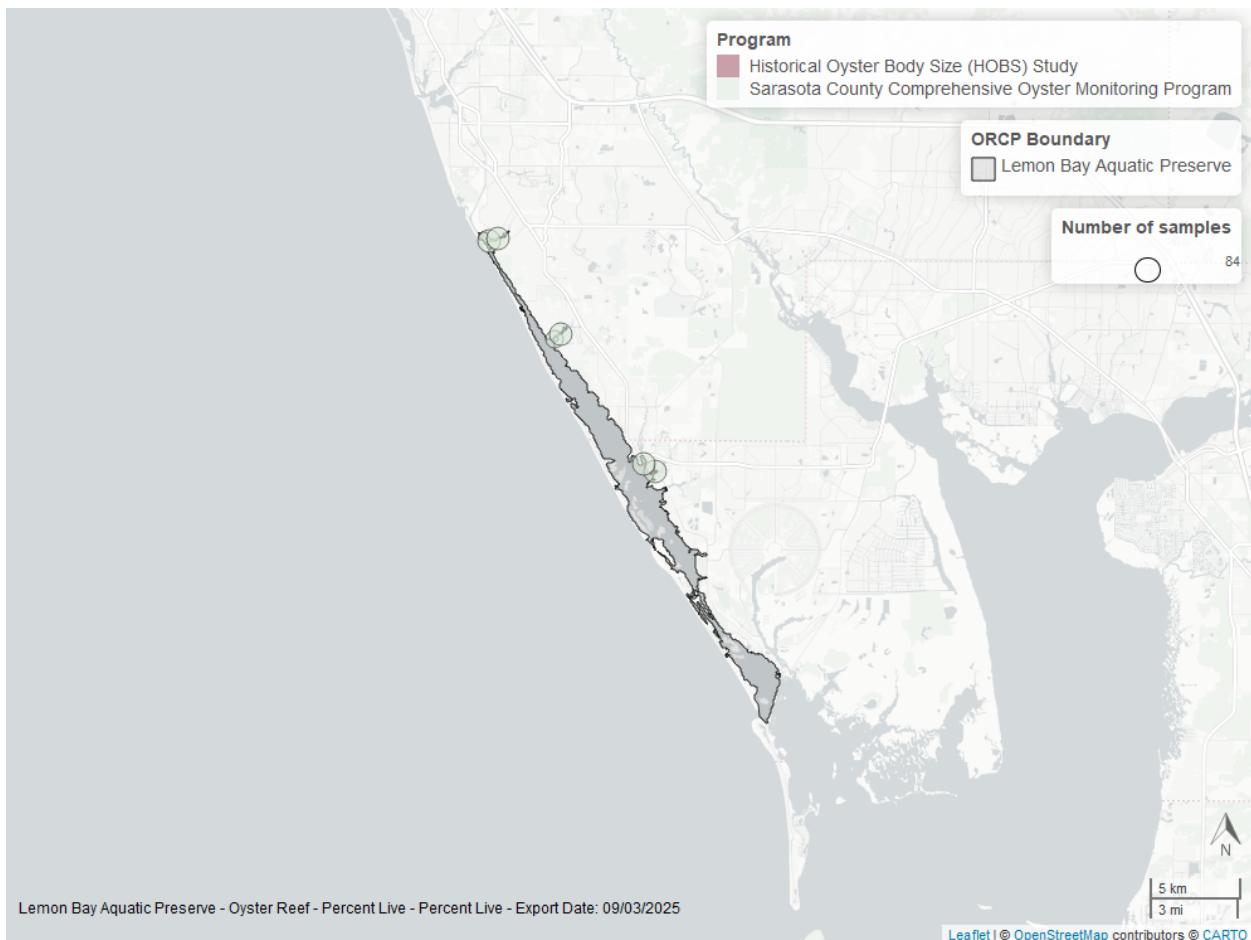


Figure 36: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 36: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	No significant change	0	0.09	-0.01 to 0

For natural reefs, percent live cover decreased by an average of 0% per year.



Loxahatchee River-Lake Worth Creek Aquatic Preserve

Natural

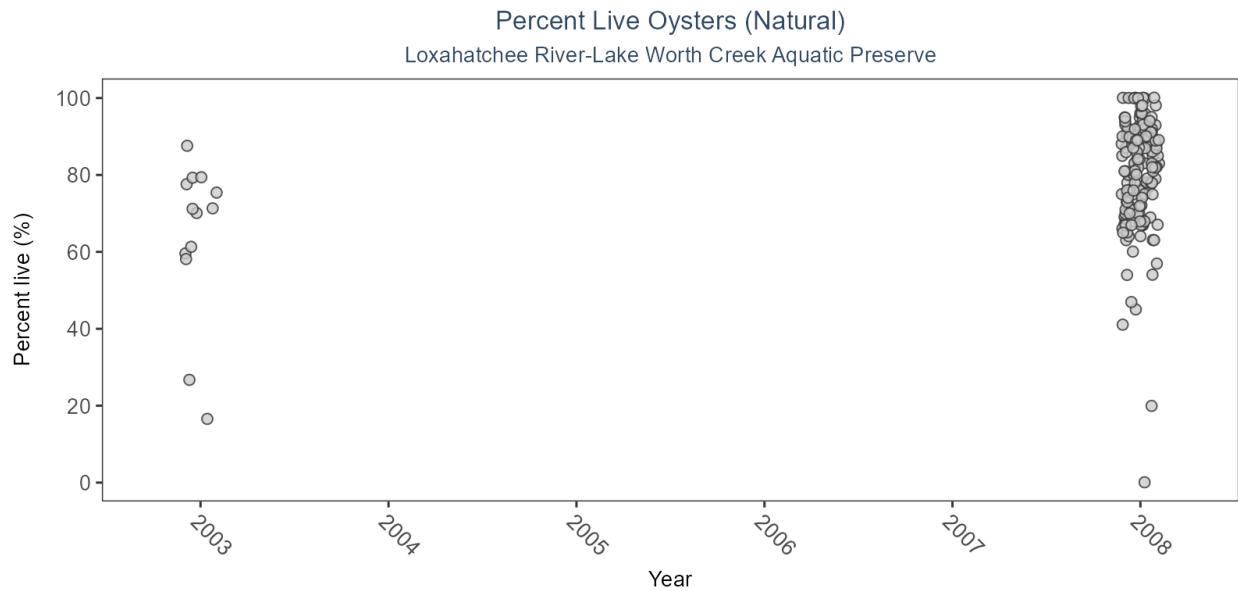
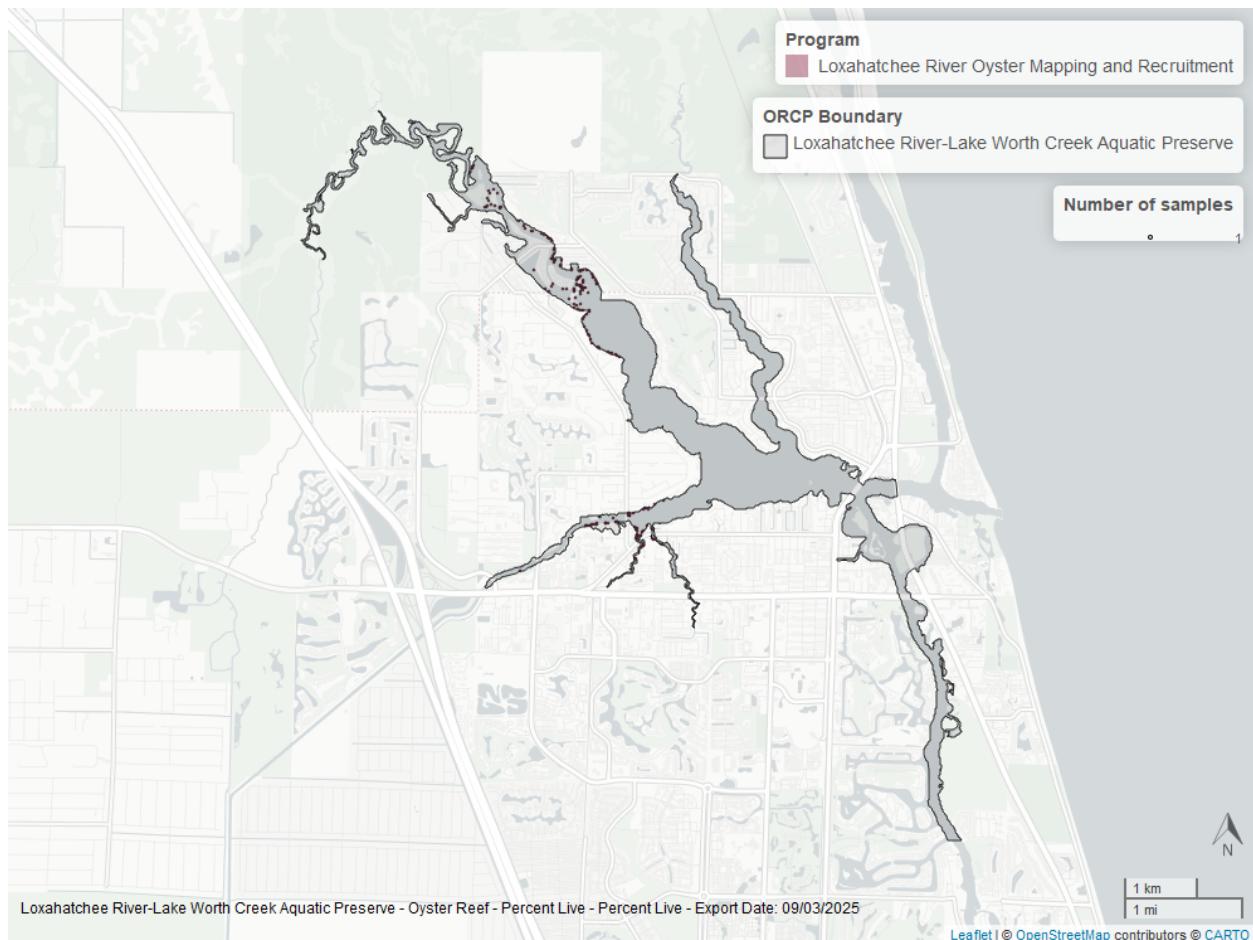


Figure 37: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 37: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Mosquito Lagoon Aquatic Preserve

Natural

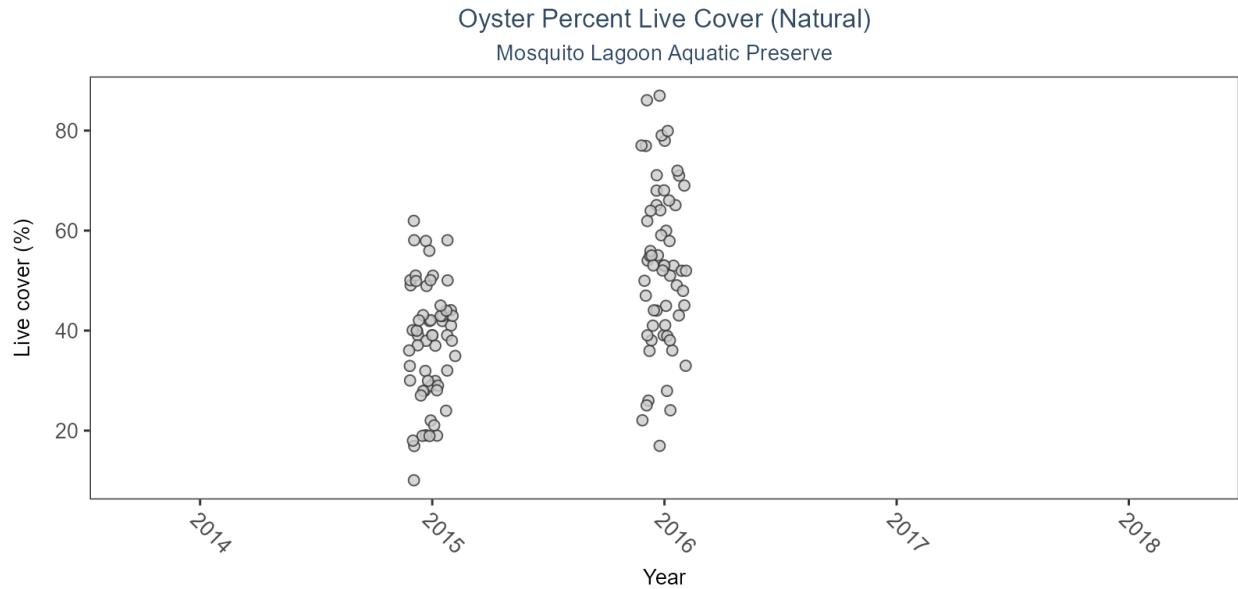
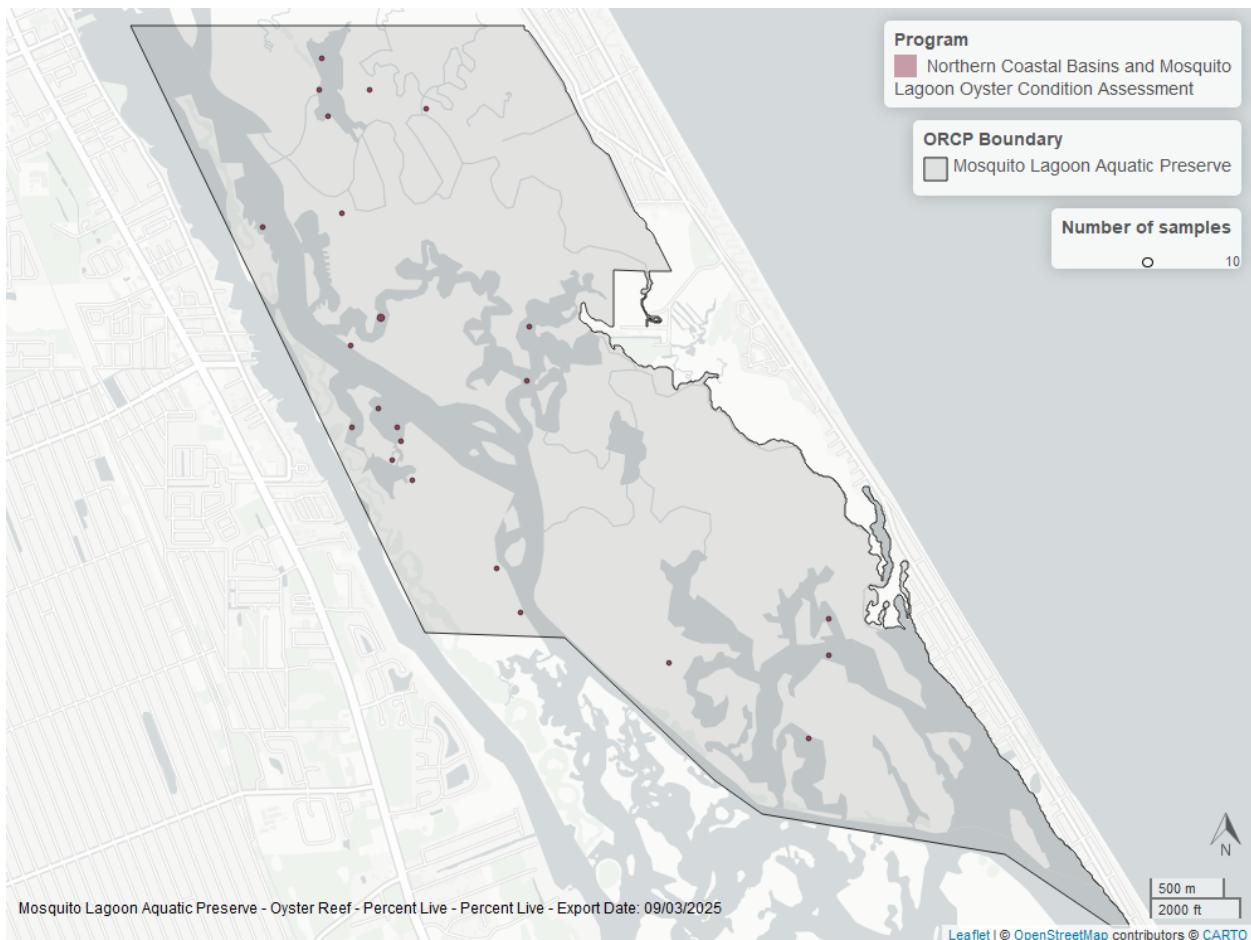


Figure 38: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 38: Model results for Oyster Percent Live - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Nassau River-St. Johns River Marshes Aquatic Preserve

Natural

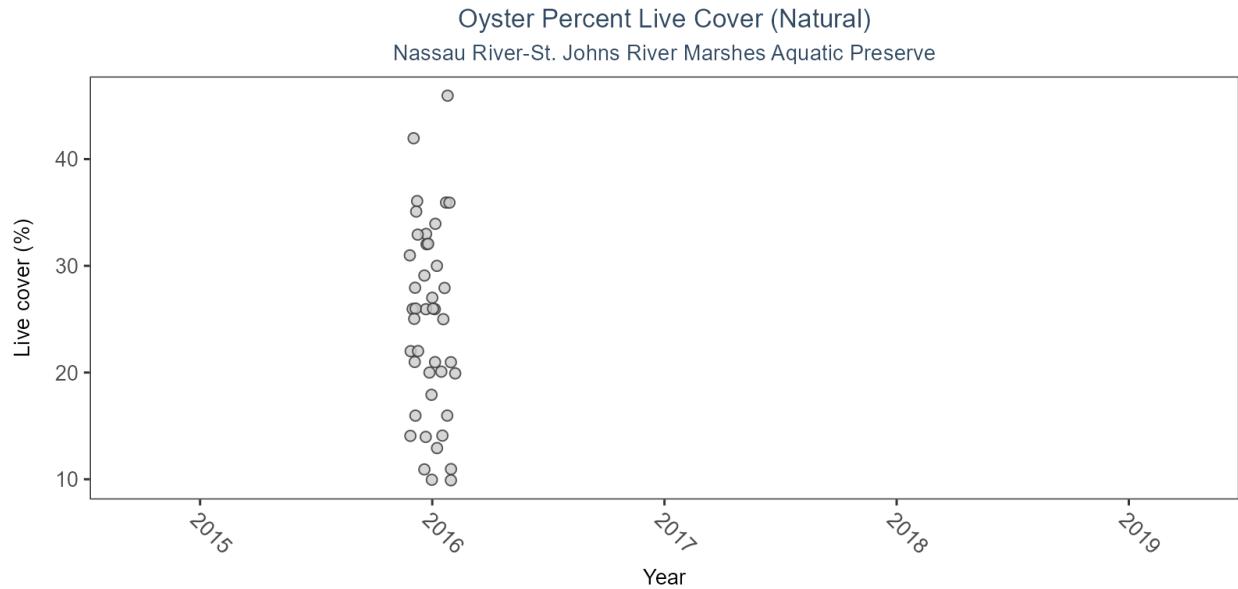
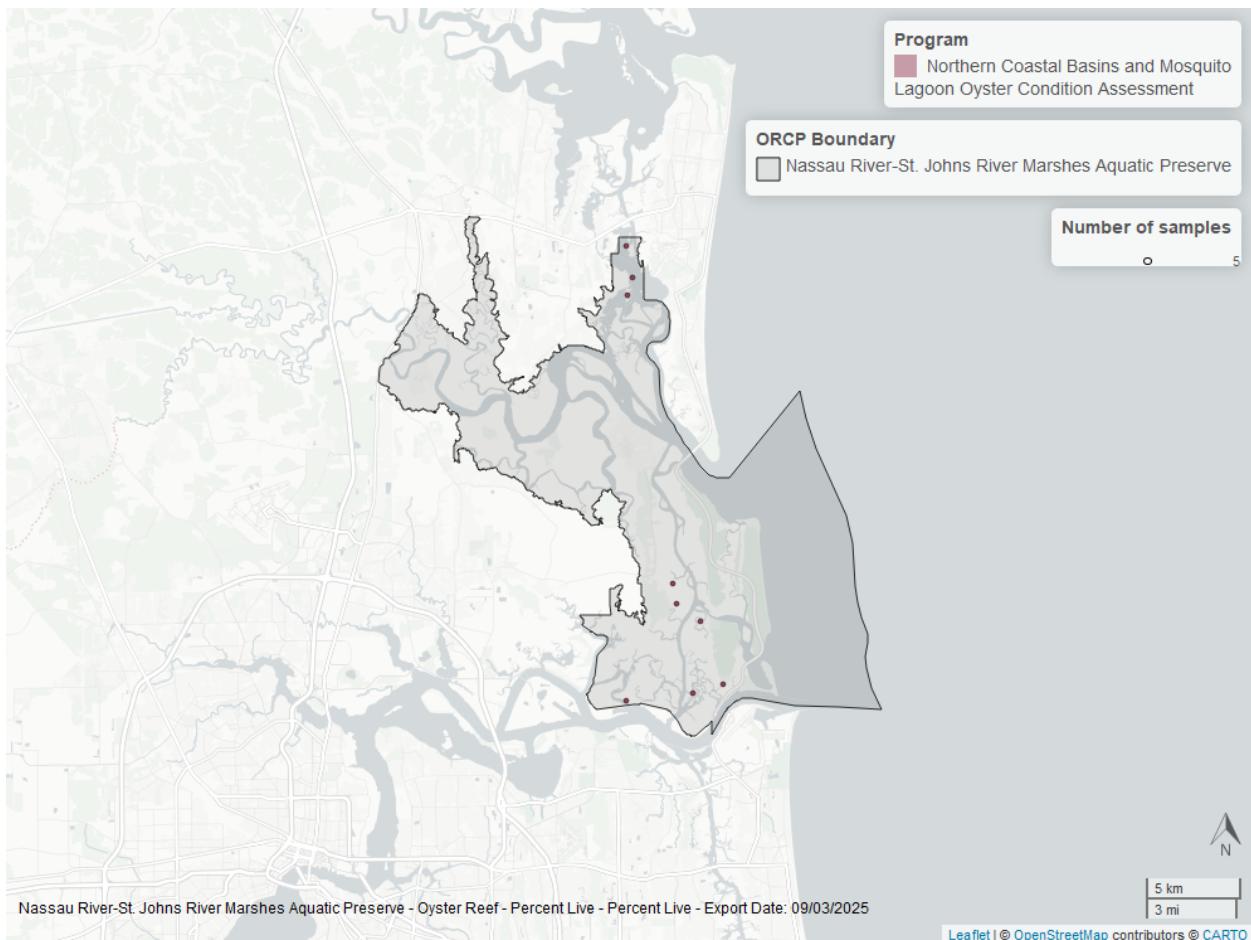


Figure 39: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 39: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Pine Island Sound Aquatic Preserve

Natural

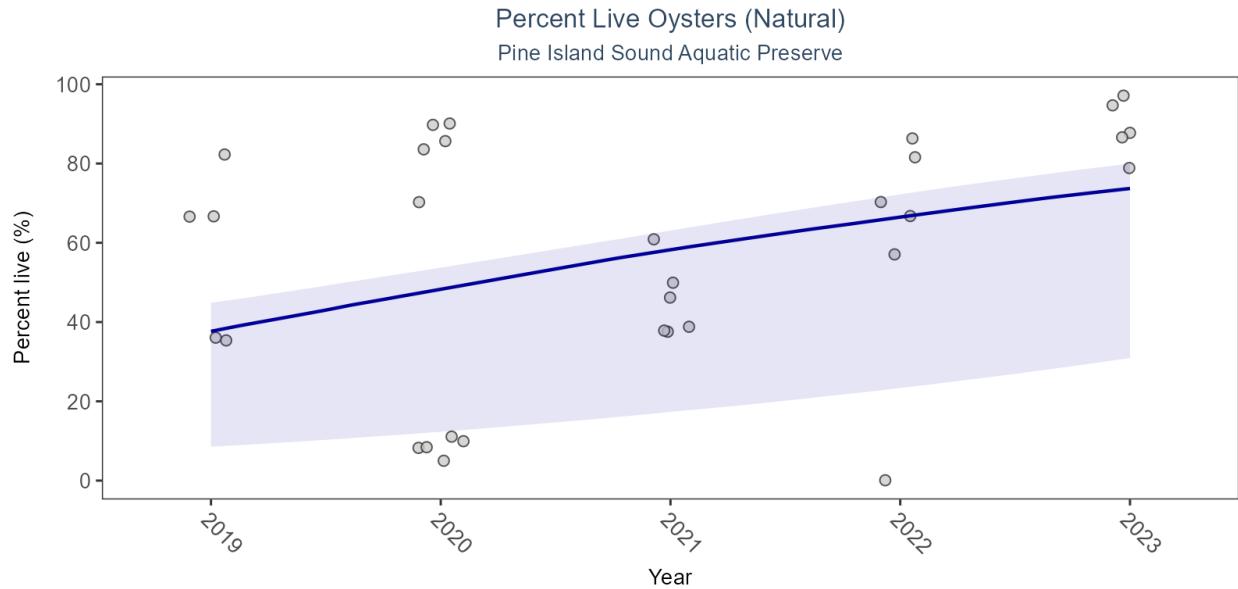


Figure 40: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 40: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	No significant change	7.6	17.27	5.65 to 8.82

For natural reefs, percent live cover increased by an average of 7.88% per year. For restored reefs, percent live cover increased by an average of 3.77% per year.

Restored

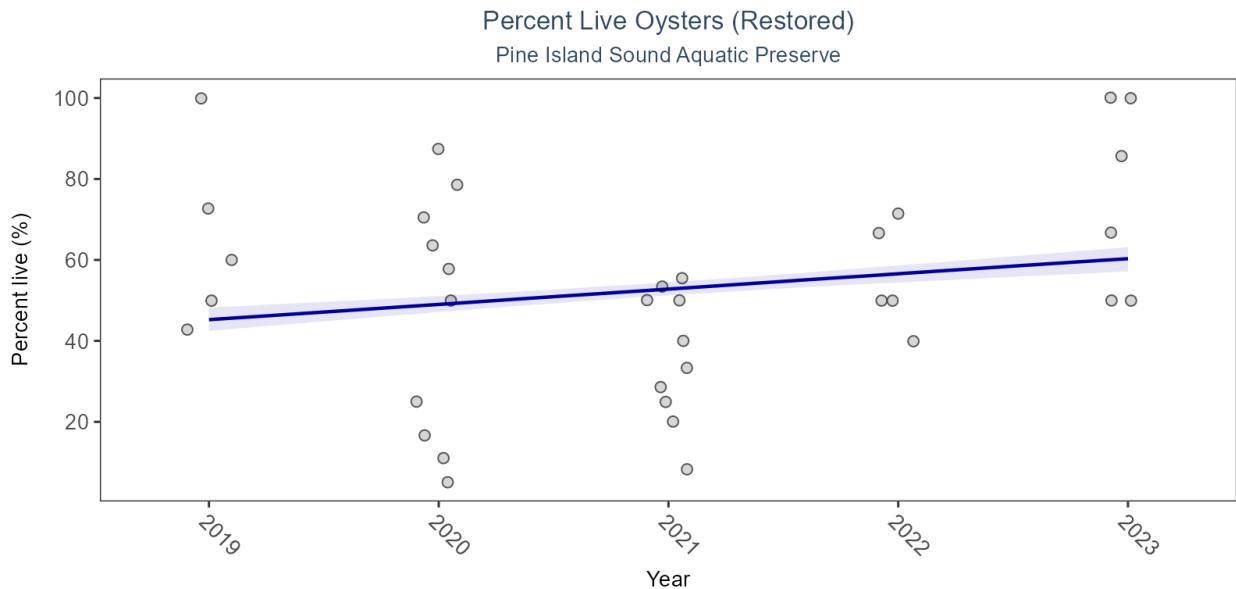
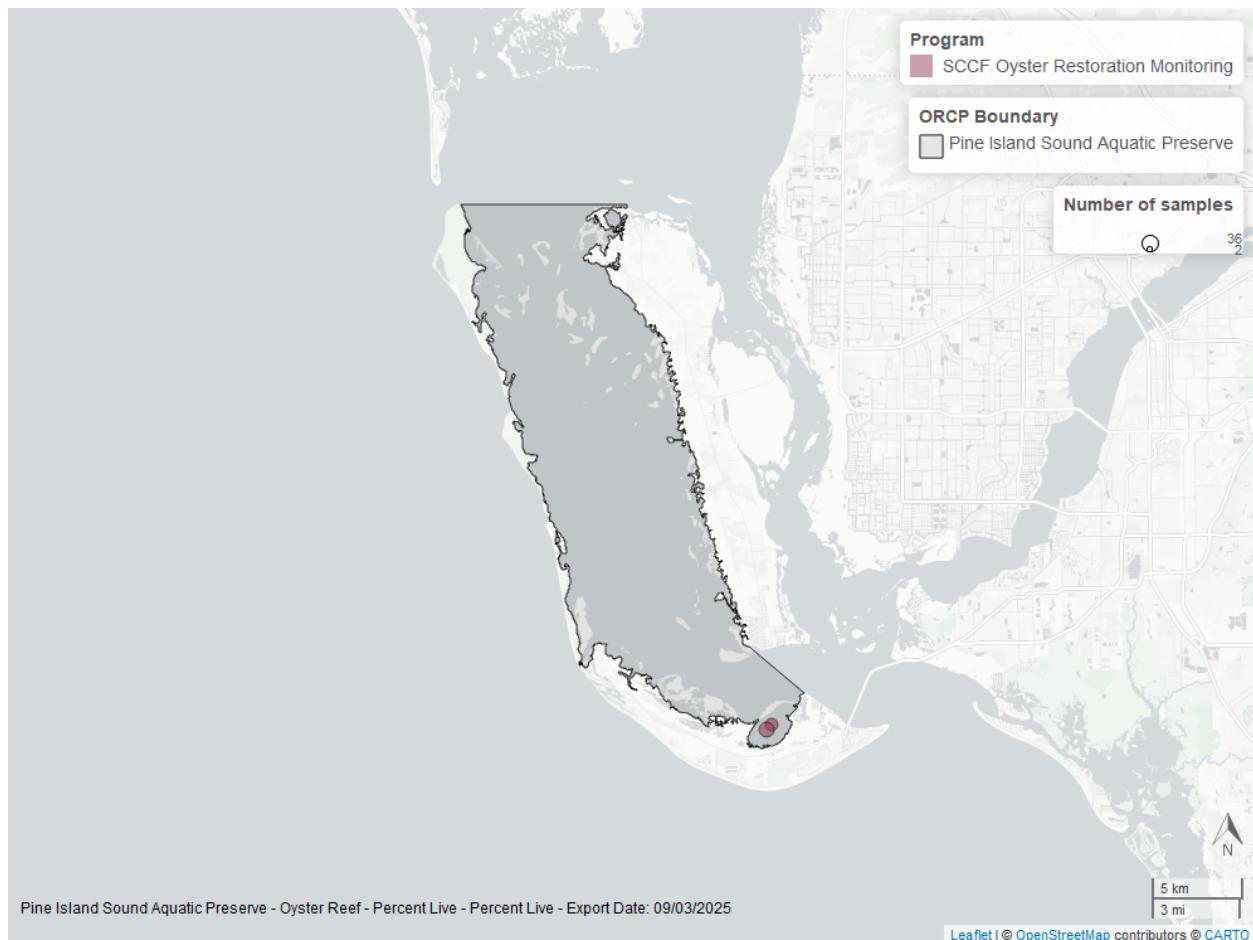


Figure 41: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 41: Model results for Oyster Percent Live - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	No significant change	3.77	1.1	3.67 to 3.72

For natural reefs, percent live cover increased by an average of 7.88% per year. For restored reefs, percent live cover increased by an average of 3.77% per year.



Pinellas County Aquatic Preserve

Natural

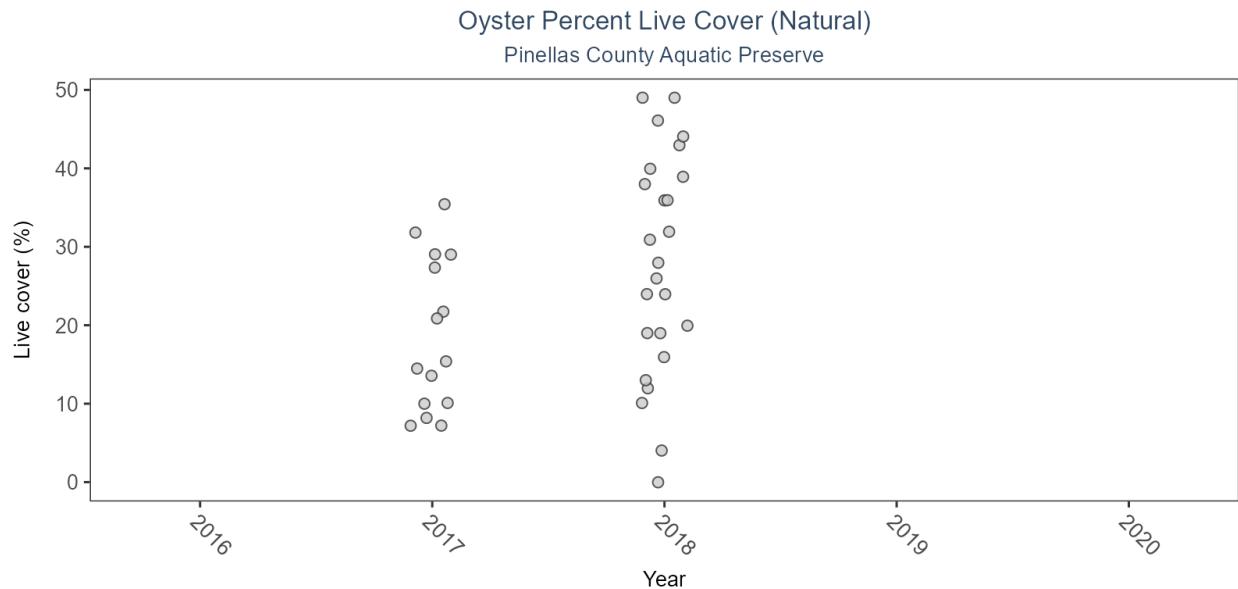
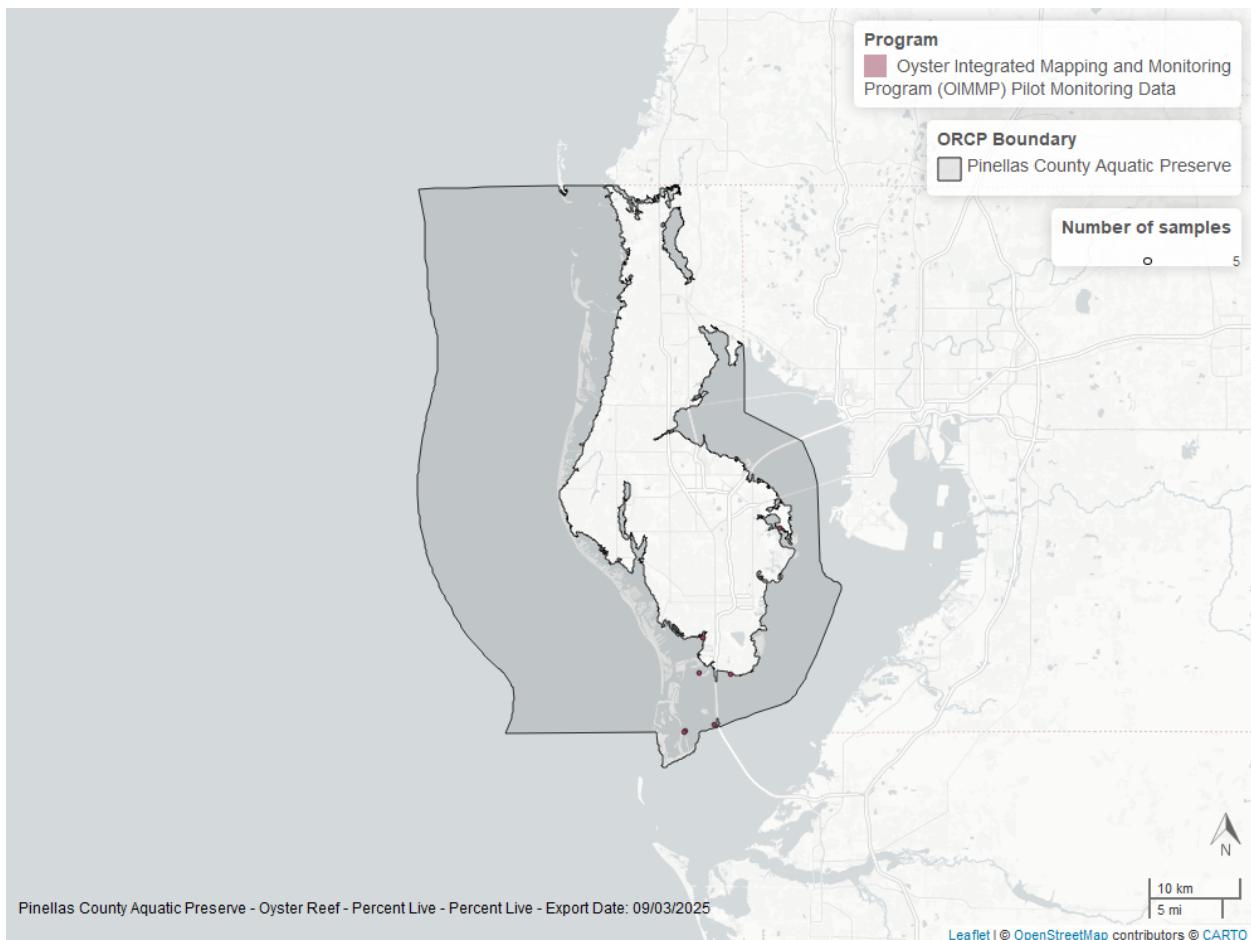


Figure 42: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 42: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



St. Martins Marsh Aquatic Preserve

Natural

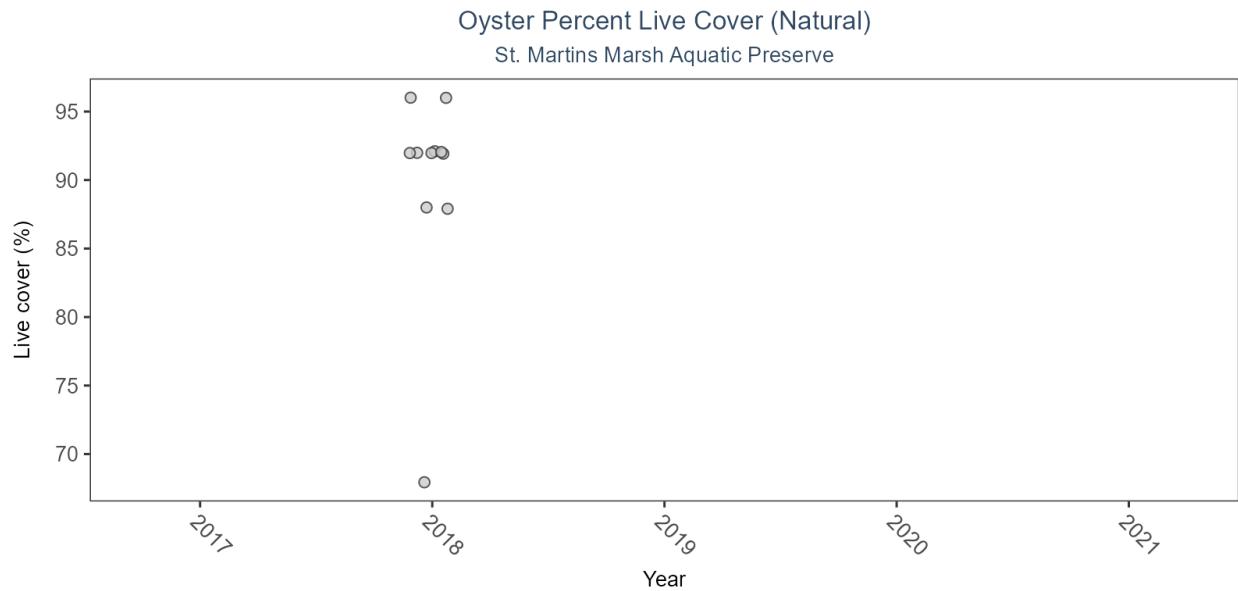
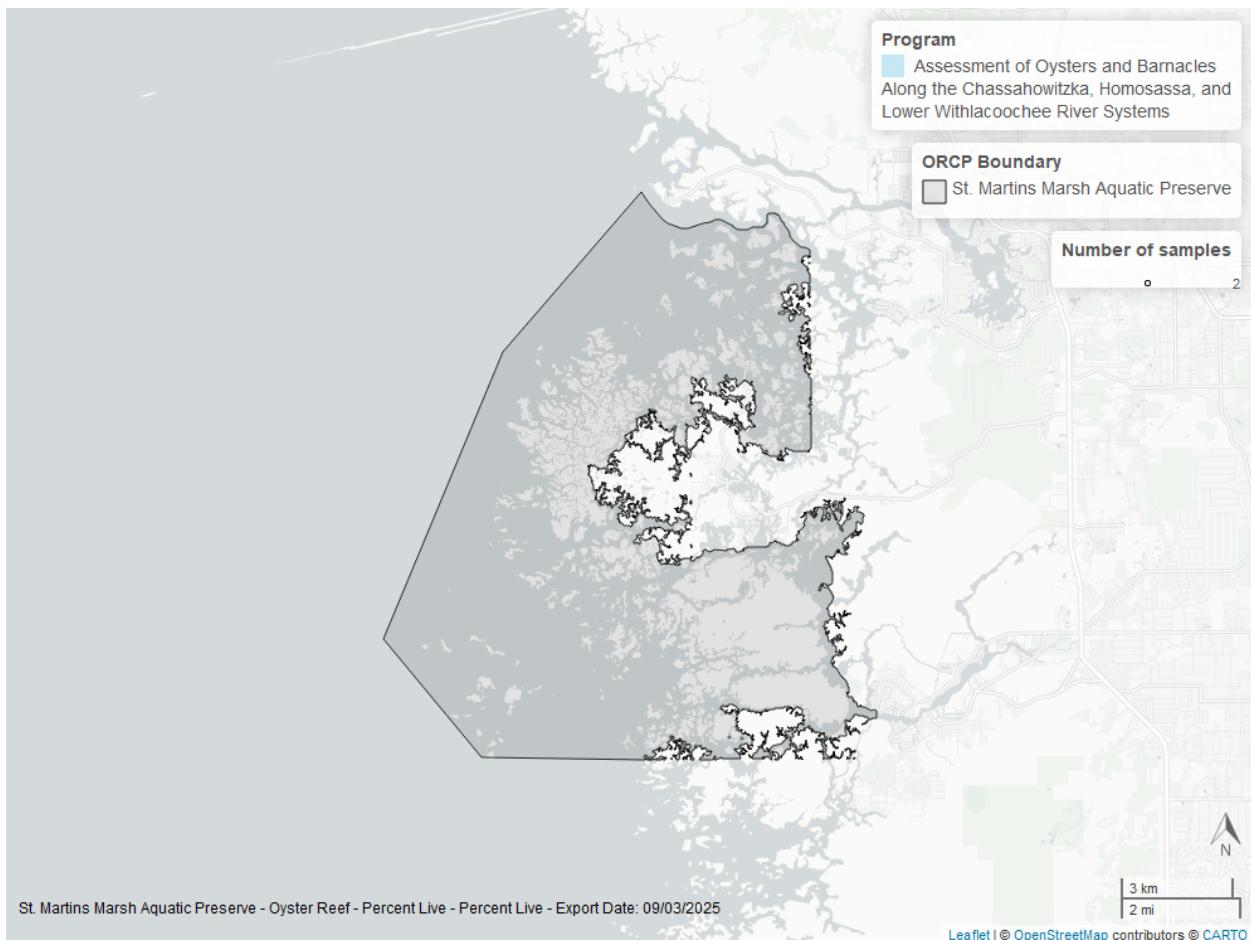


Figure 43: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 43: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Tomoka Marsh Aquatic Preserve

Natural

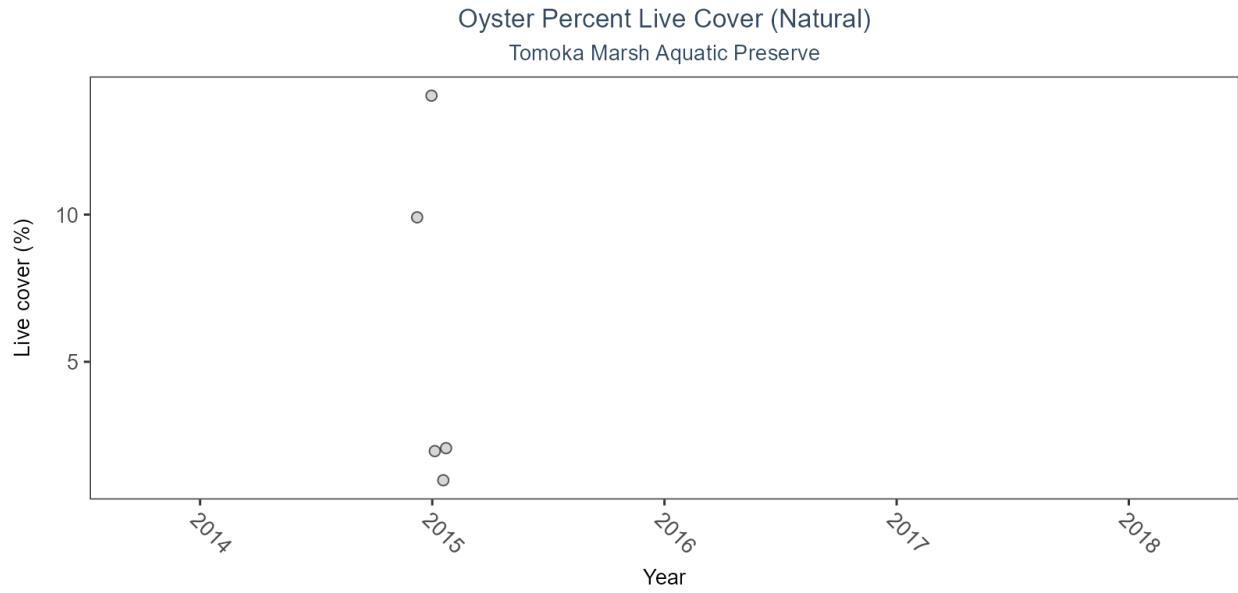
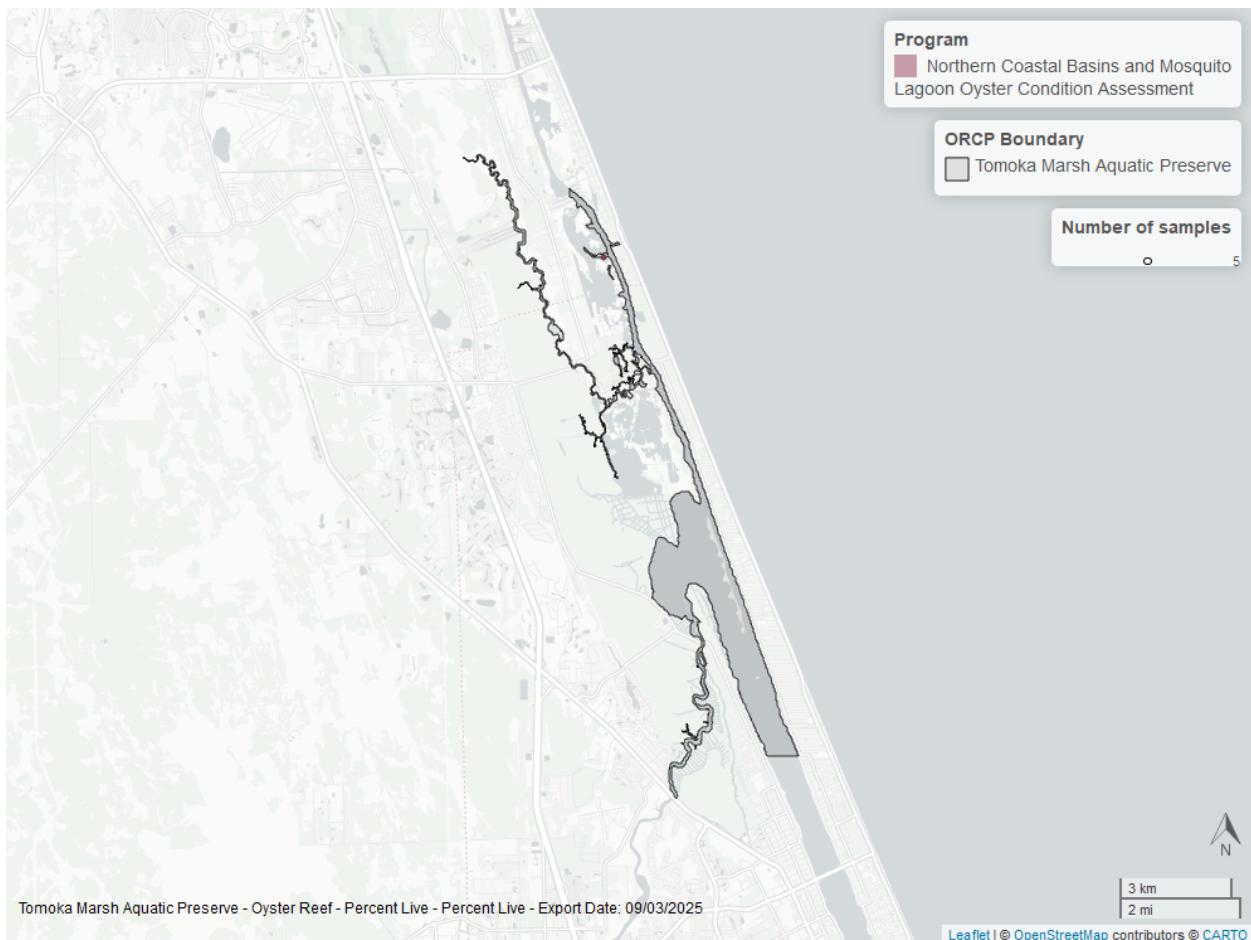


Figure 44: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 44: Model results for Oyster Percent Live - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for percent live cover.



Yellow River Marsh Aquatic Preserve

Restored

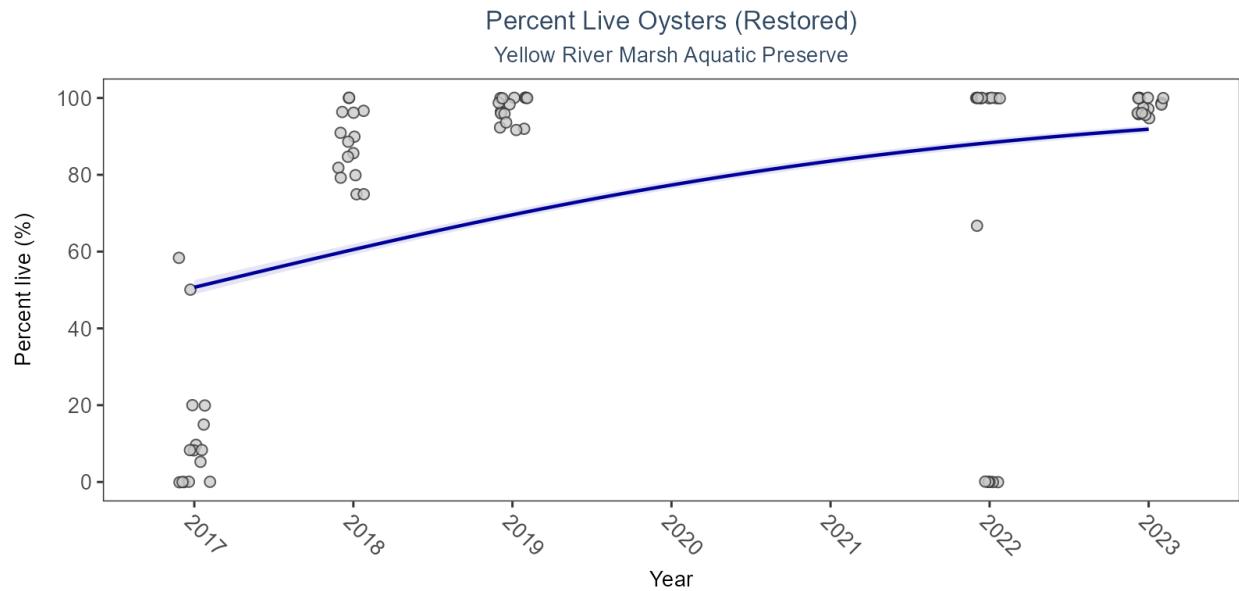
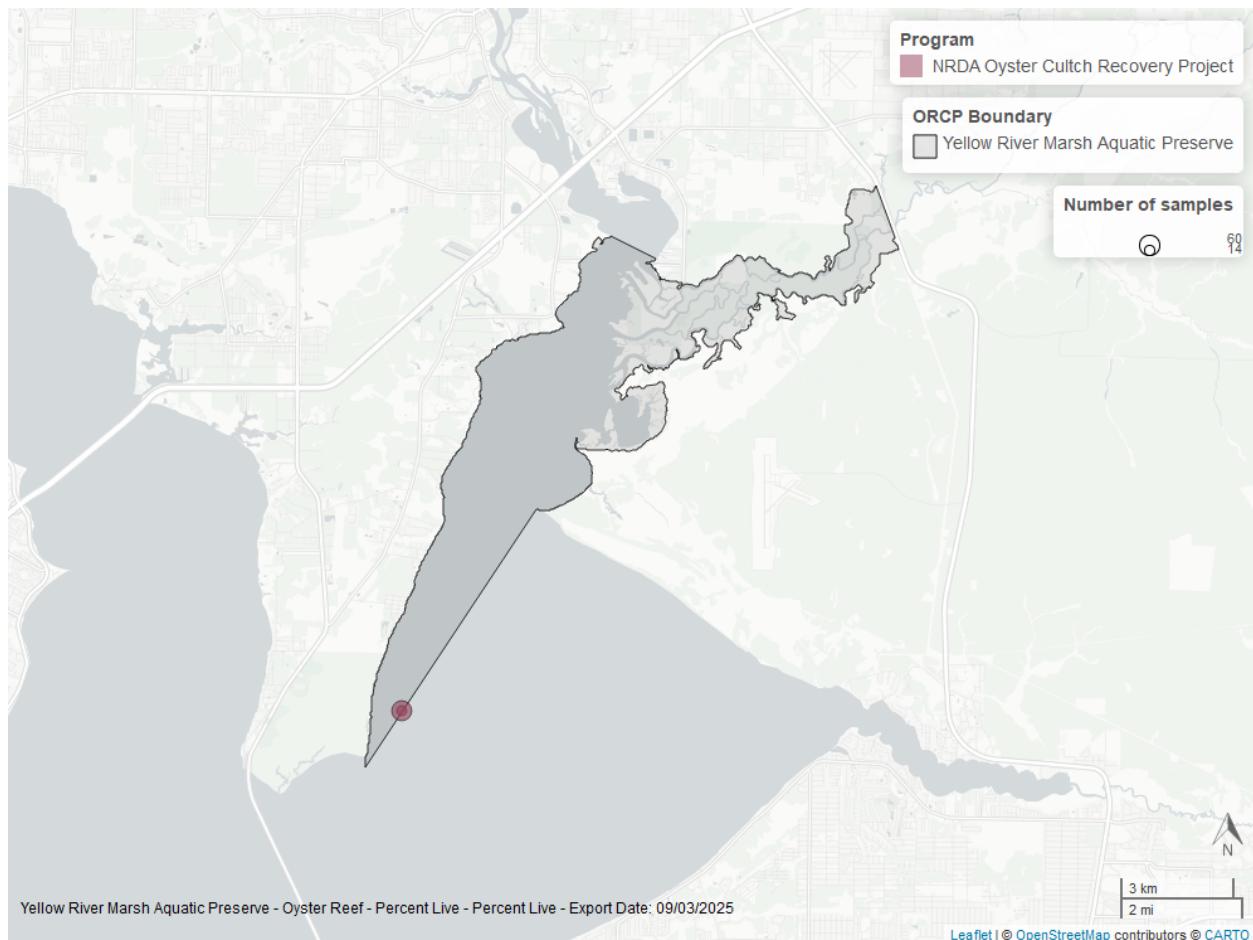


Figure 45: Scatter plot of percent live oysters over time. If the time series included five or more years with observations, an estimated trend (blue line) and a 95% credible interval (purple band) may also be plotted. Data points are jittered horizontally to reduce overlap.

Table 45: Model results for Oyster Percent Live - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	No significant change	6.86	0.6	7 to 6.7

For restored reefs, percent live cover increased by an average of 6.86% per year.



Shell Height

Apalachicola Bay Aquatic Preserve

Natural

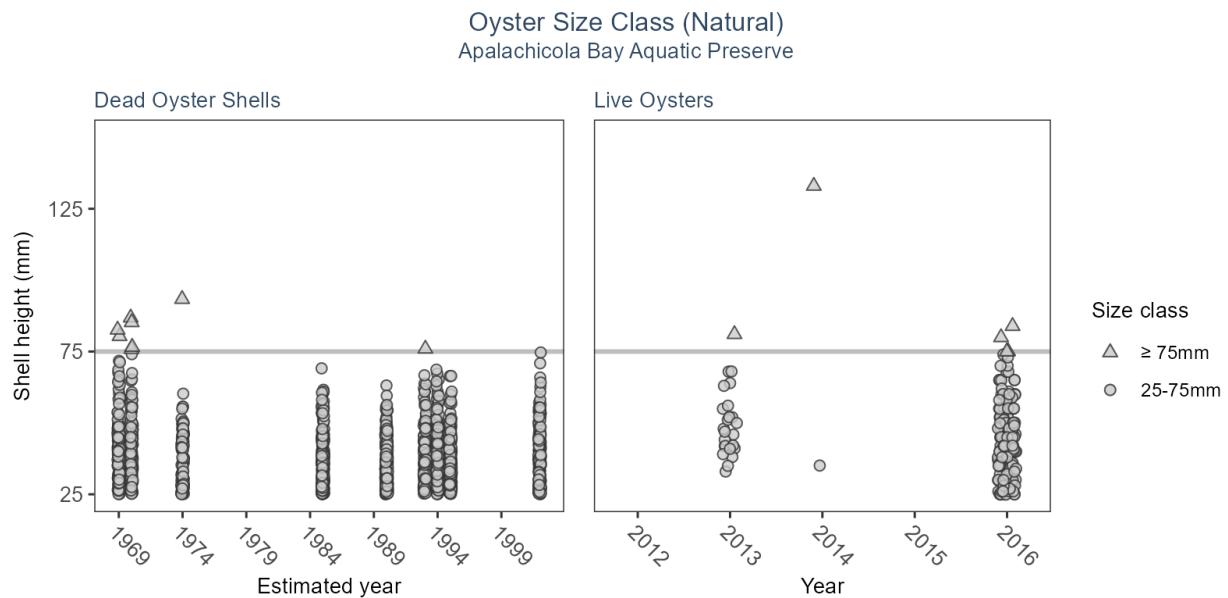
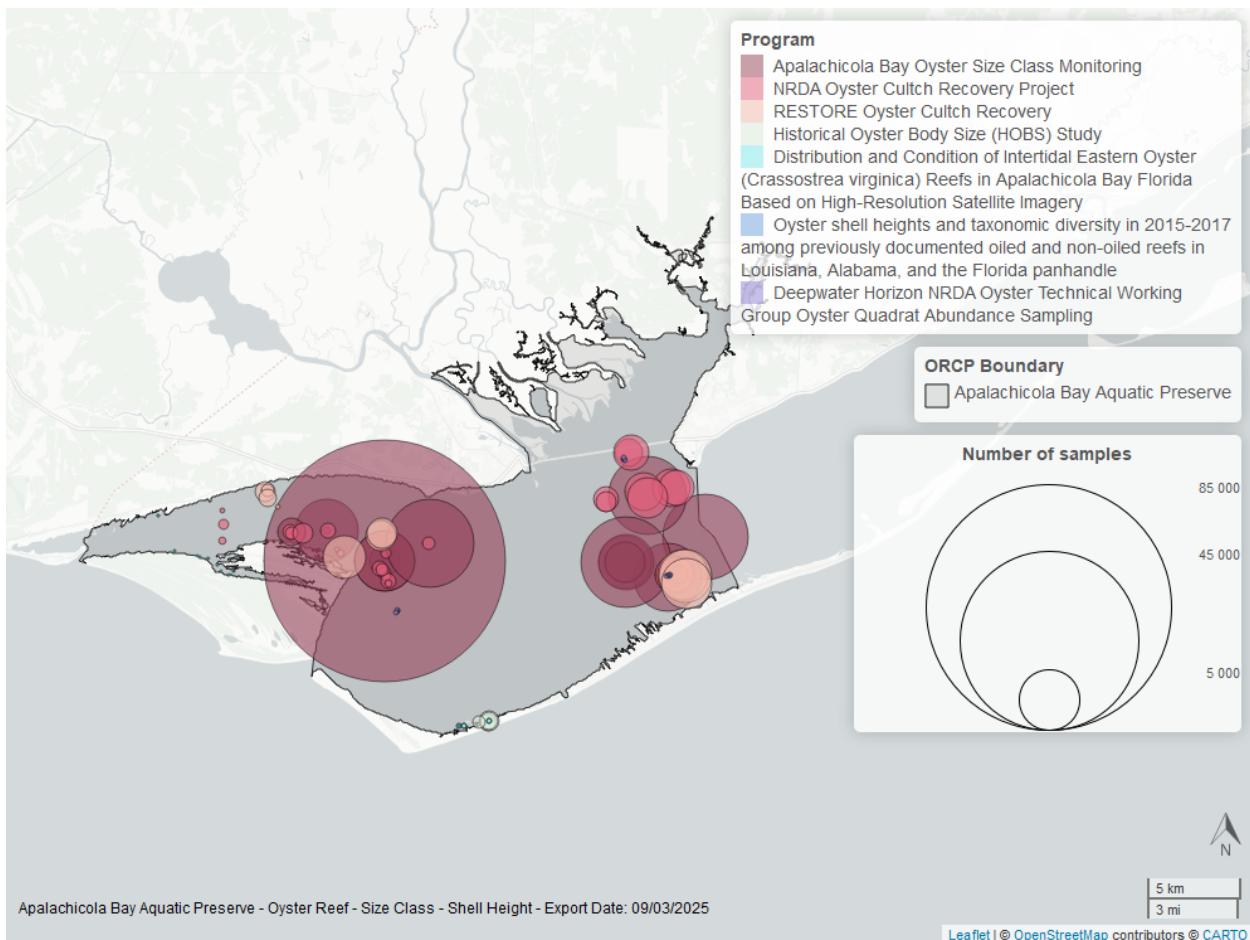


Table 46: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. For restored reefs, a model could not be fitted for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. Models are not run on dead oyster shell measurements.



Apalachicola National Estuarine Research Reserve

Natural

Oyster Size Class (Natural)
Apalachicola National Estuarine Research Reserve

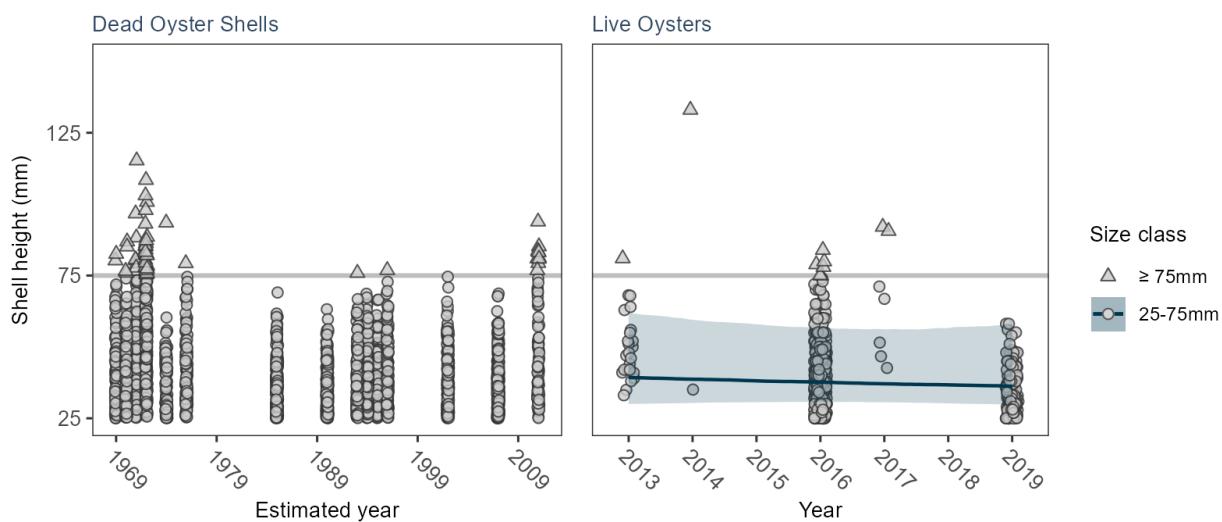
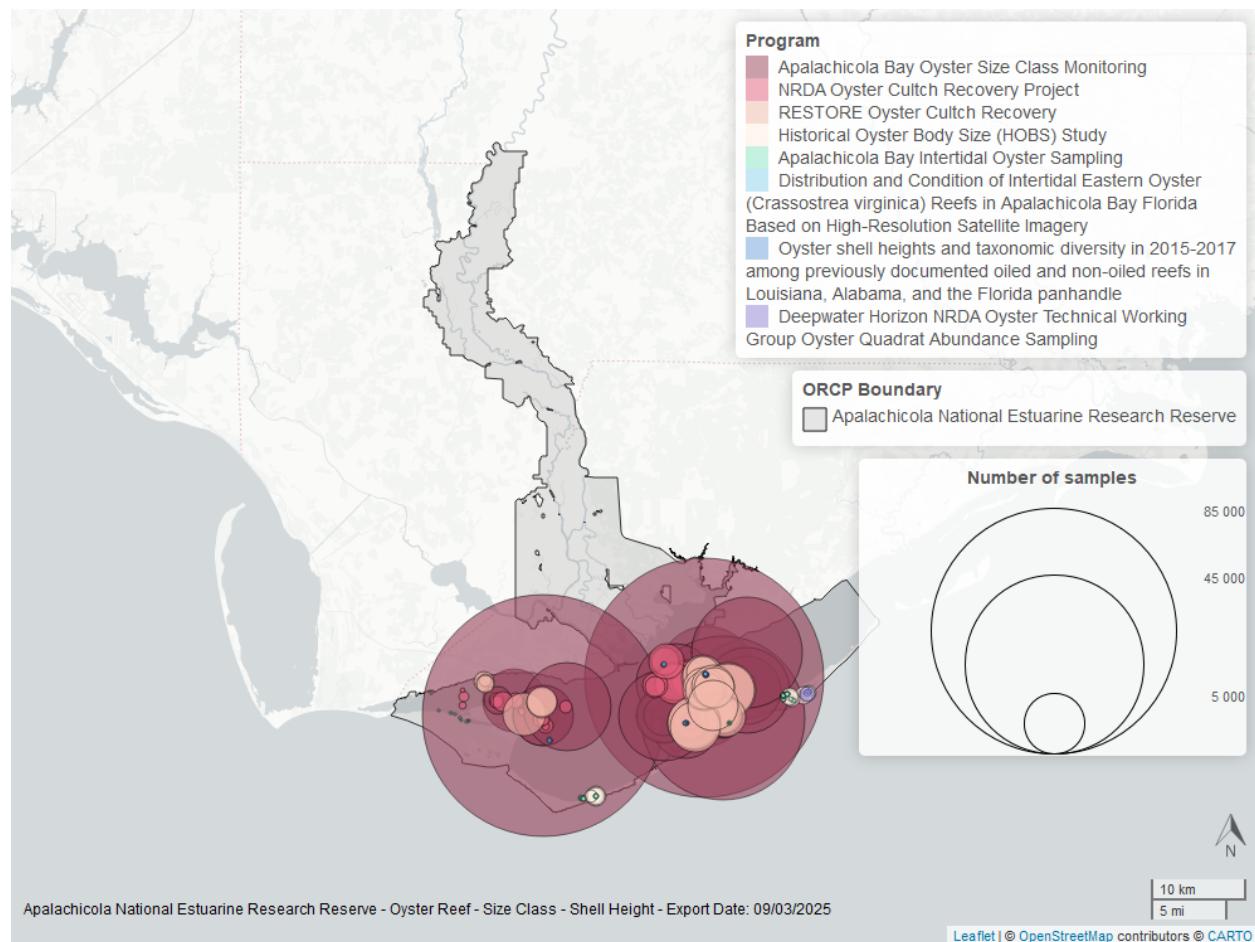


Table 47: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	-1.18	4.54	-10.28 to 7.86

For natural reefs, annual average live oyster shell height in the 25-75mm size class decreased by 1.18 mm per year, and there was insufficient data to calculate a trend for live oysters in the $\geq 75\text{mm}$ size class. For restored reefs, a model could not be fitted for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. Models are not run on dead oyster shell measurements.



Big Bend Seagrasses Aquatic Preserve

Natural

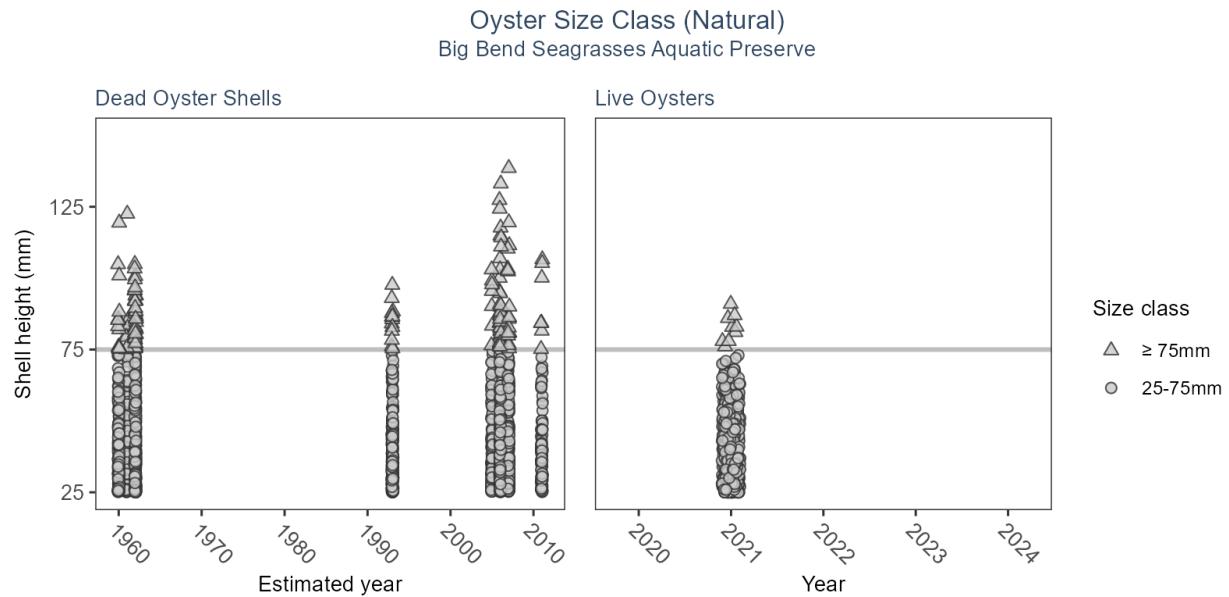
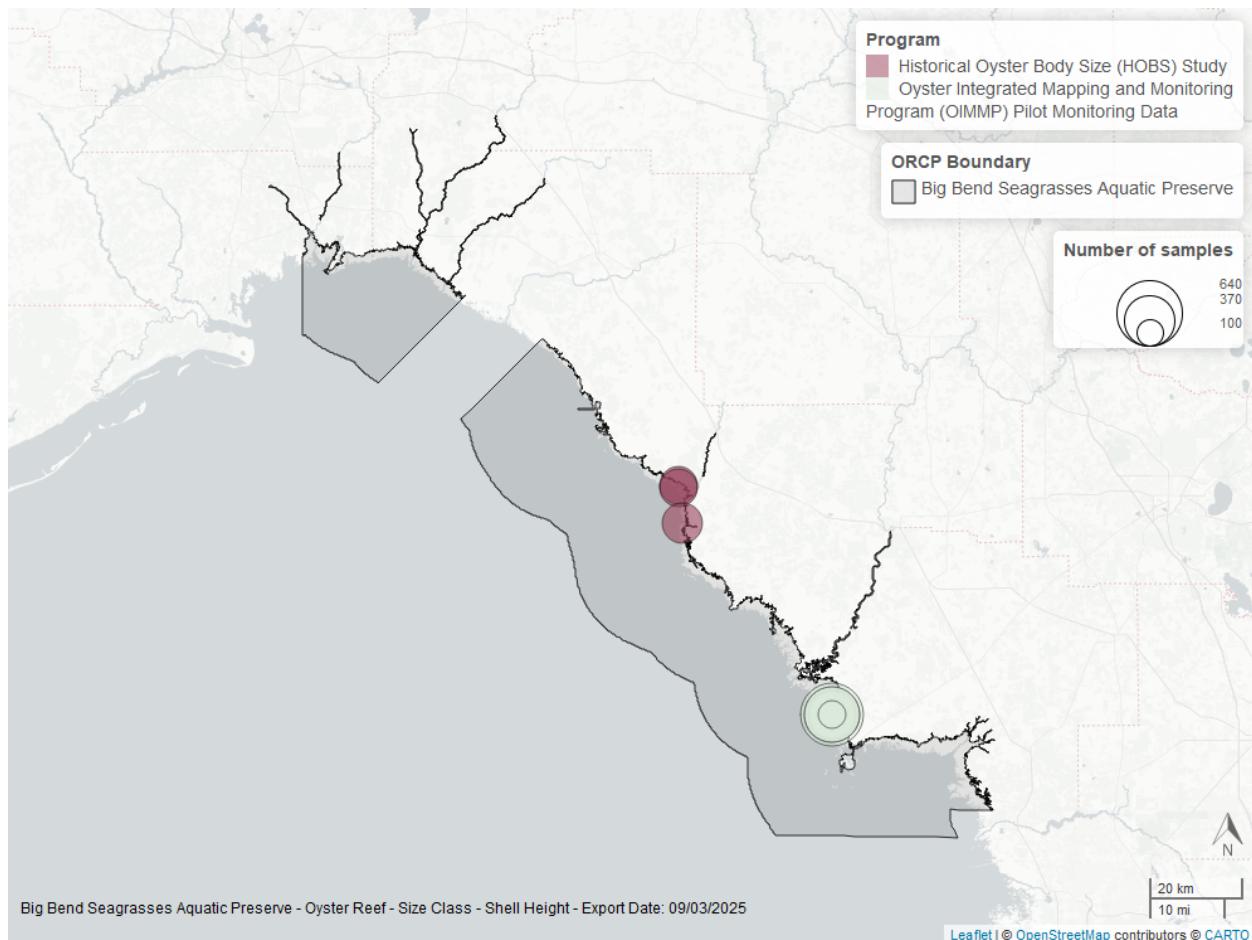


Table 48: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. Models are not run on dead oyster shell measurements.



Boca Ciega Bay Aquatic Preserve

Natural

Oyster Size Class (Natural)
Boca Ciega Bay Aquatic Preserve

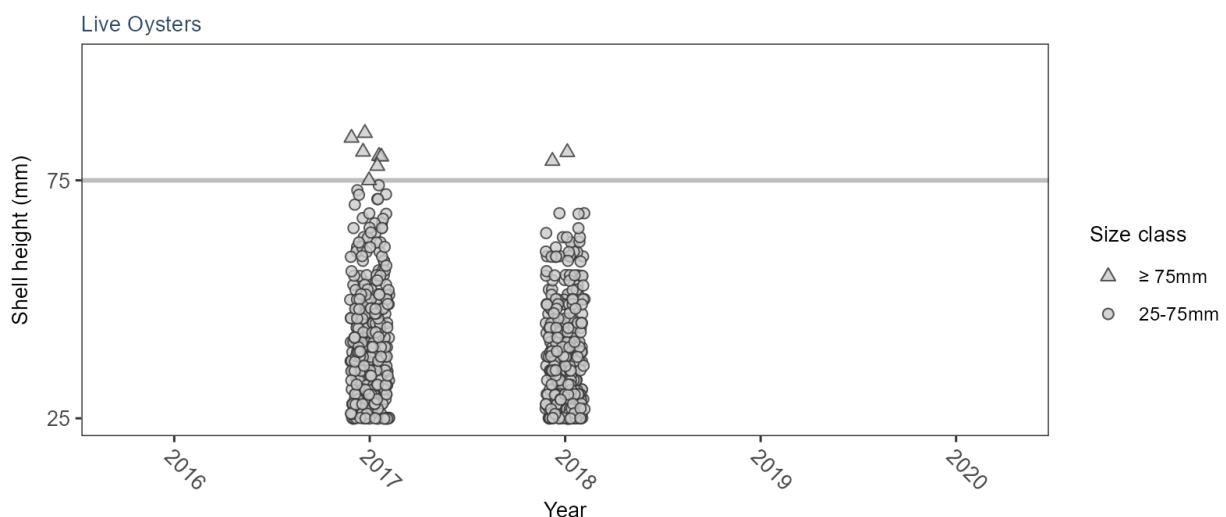


Table 49: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Estero Bay Aquatic Preserve

Restored

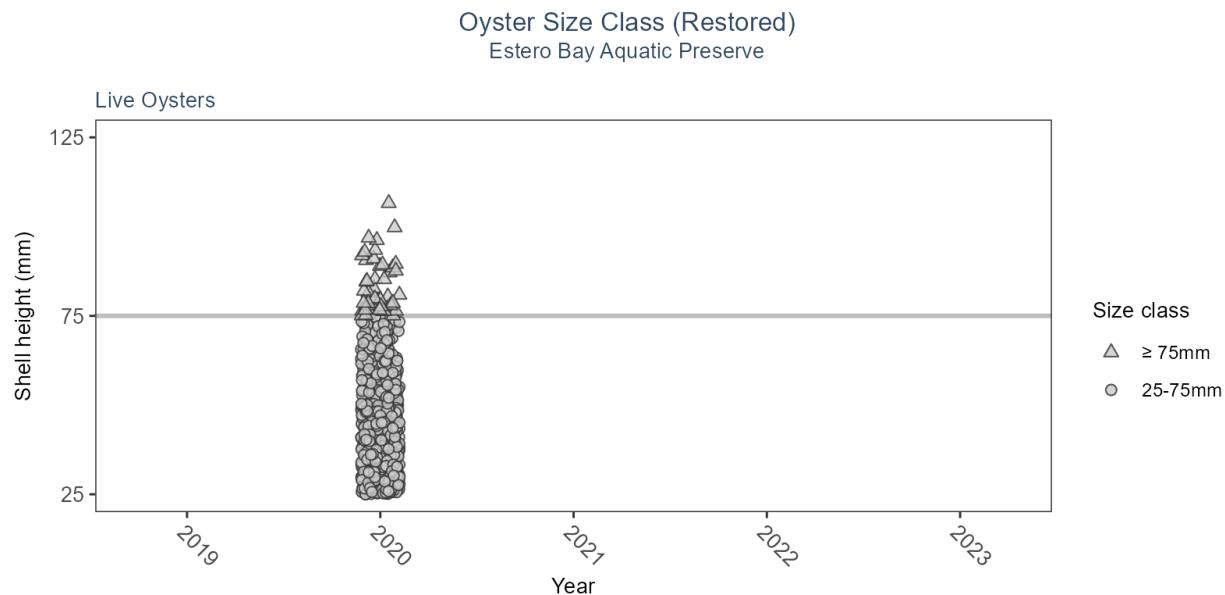


Table 50: Model results for Oyster Shell Height - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	-	-	-	-
Live Oysters	Restored	-	-	-	-
Live Oysters	Restored	-	-	-	-

For natural reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 0.65 mm per year, and it decreased by 1.08mm per year in the 25-75mm size class. For restored reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. Models are not run on dead oyster shell measurements.

Natural

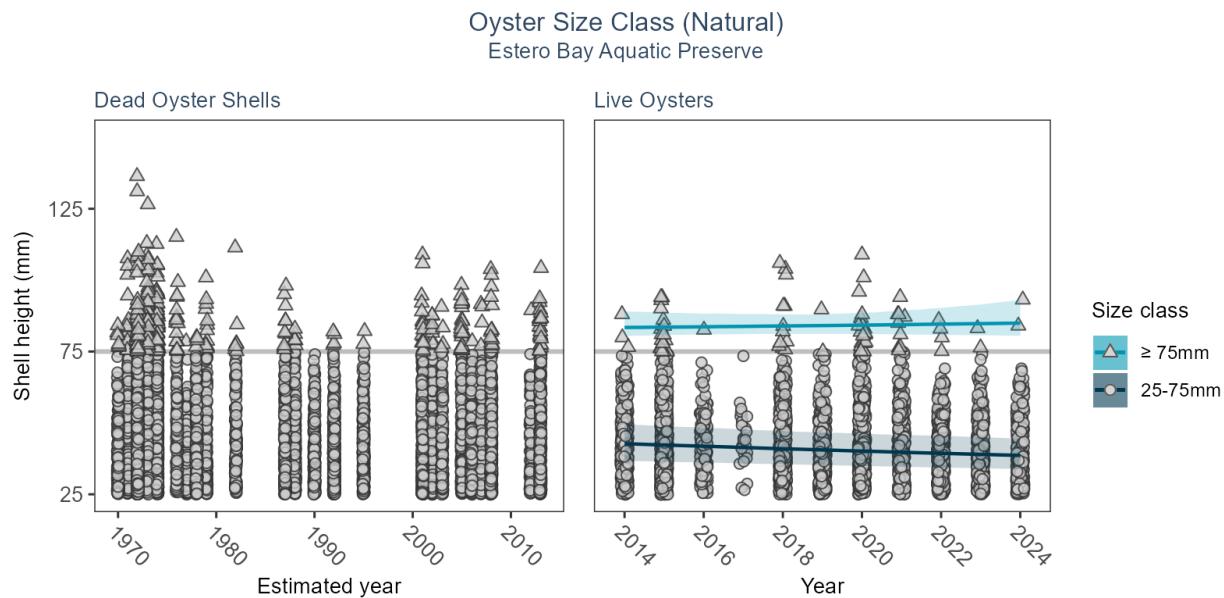
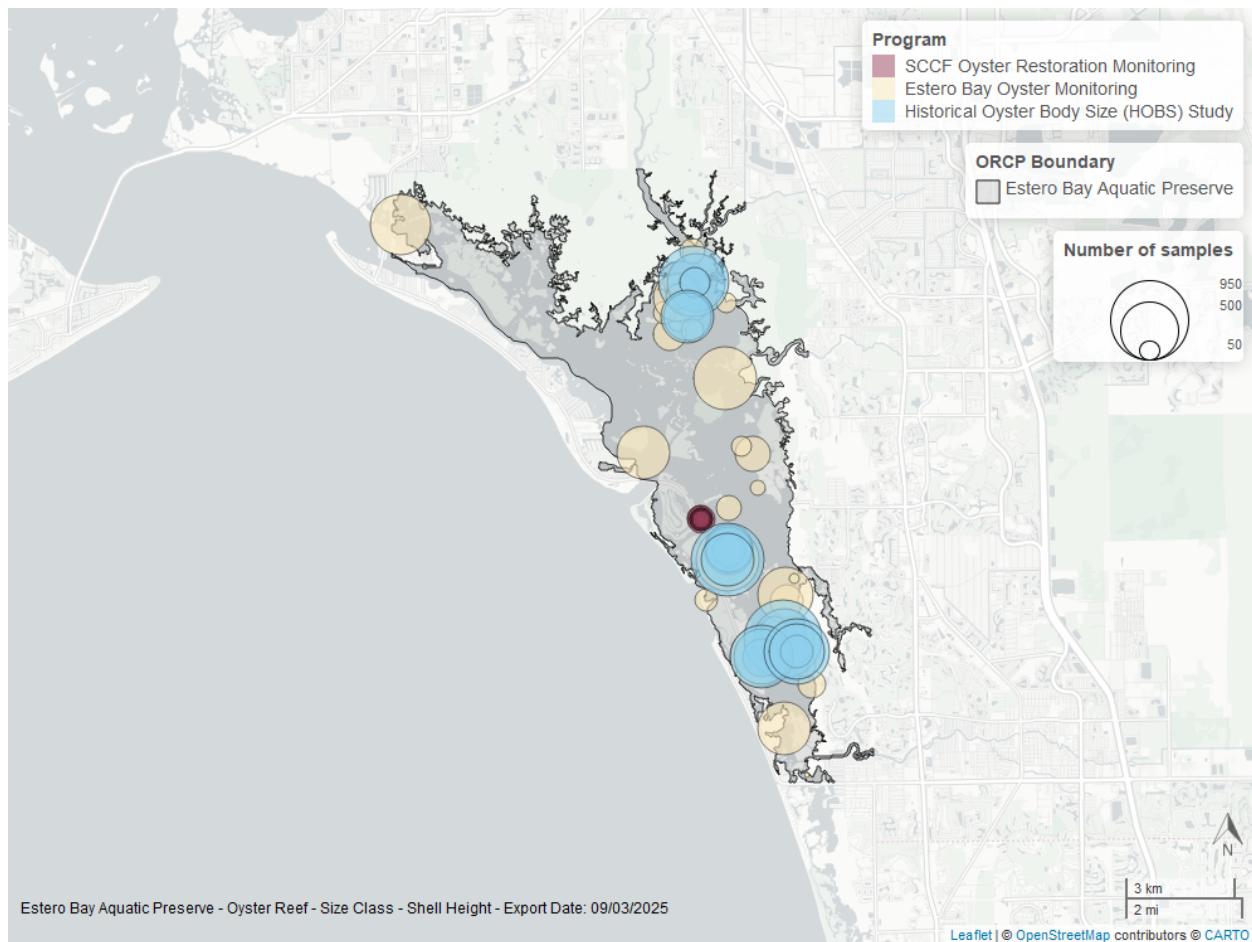


Table 51: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	0.65	1.81	-2.76 to 4.53
Live Oysters	Natural	No significant change	-1.08	0.28	-1.65 to -0.55

For natural reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 0.65 mm per year, and it decreased by 1.08mm per year in the 25-75mm size class. For restored reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class. Models are not run on dead oyster shell measurements.



Guana River Marsh Aquatic Preserve

Natural

Oyster Size Class (Natural)
Guana River Marsh Aquatic Preserve

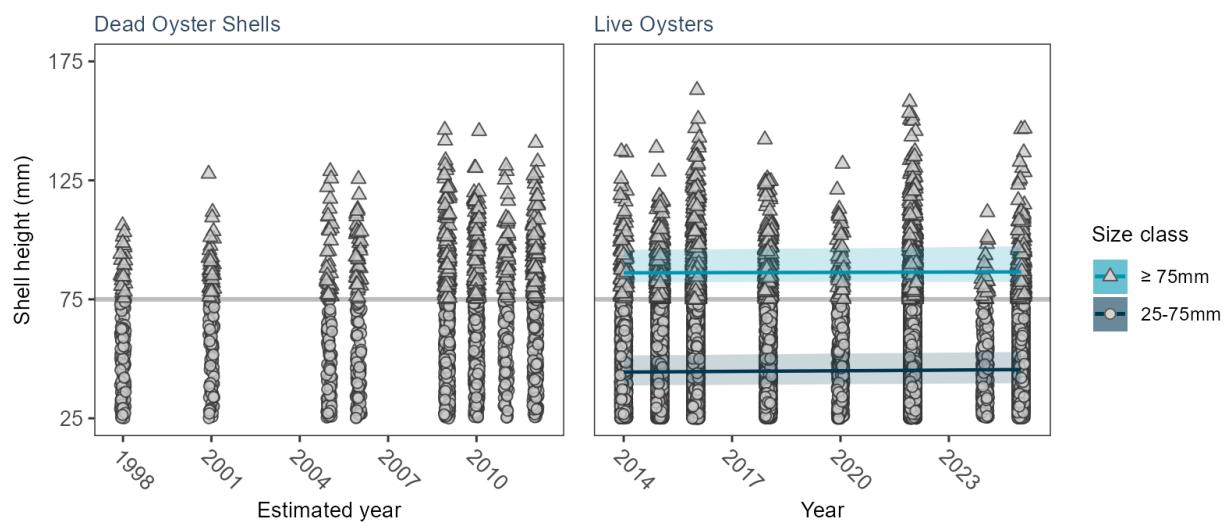
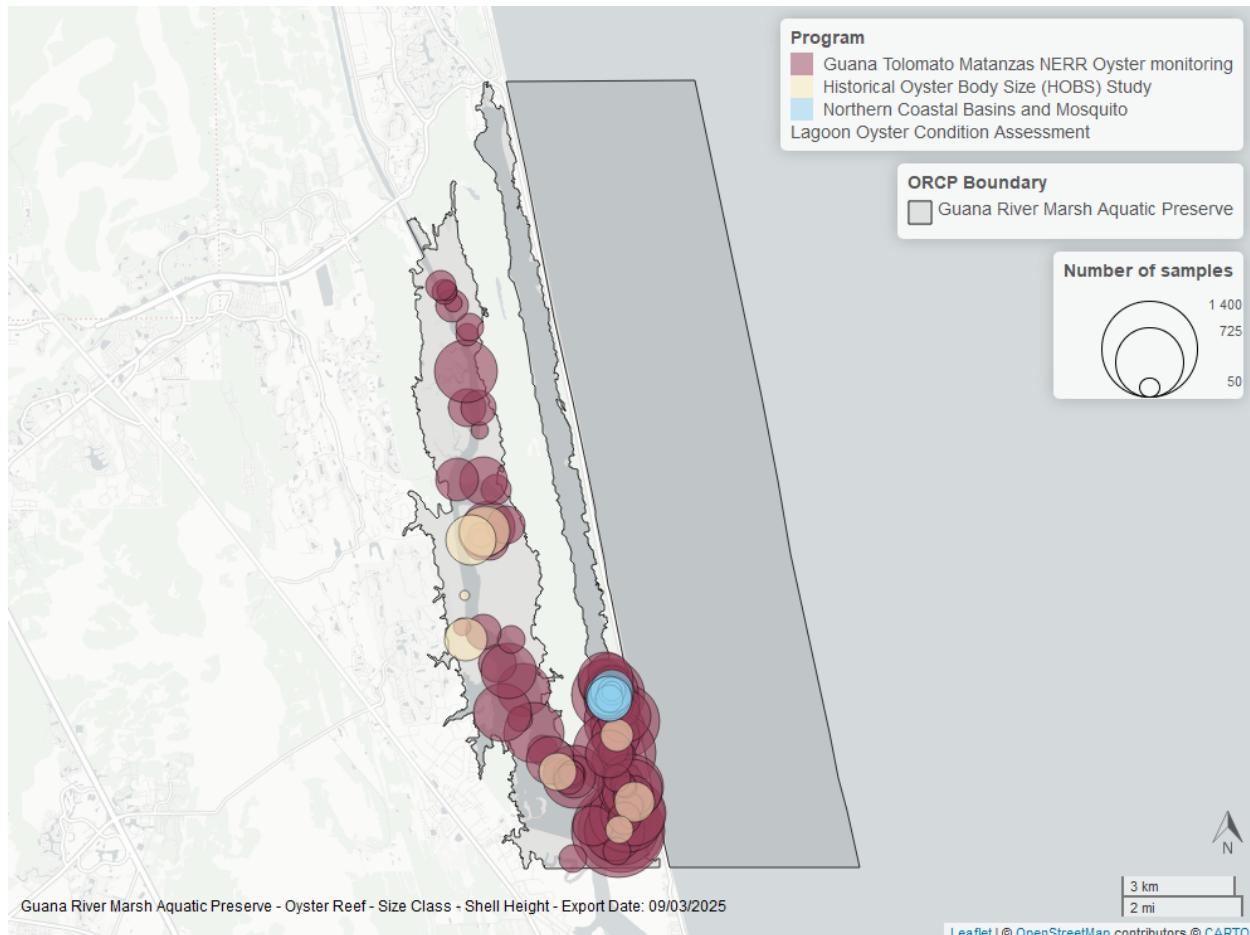


Table 52: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	0.29	0.46	-0.66 to 1.18
Live Oysters	Natural	No significant change	0.69	0.35	0 to 1.4

For natural reefs, annual average live oyster shell height in the 25-75mm and ≥ 75 mm size classes increased by 0.69 mm per year and 0.29 mm per year, respectively. Models are not run on dead oyster shell measurements.



Guana Tolomato Matanzas National Estuarine Research Reserve

Natural

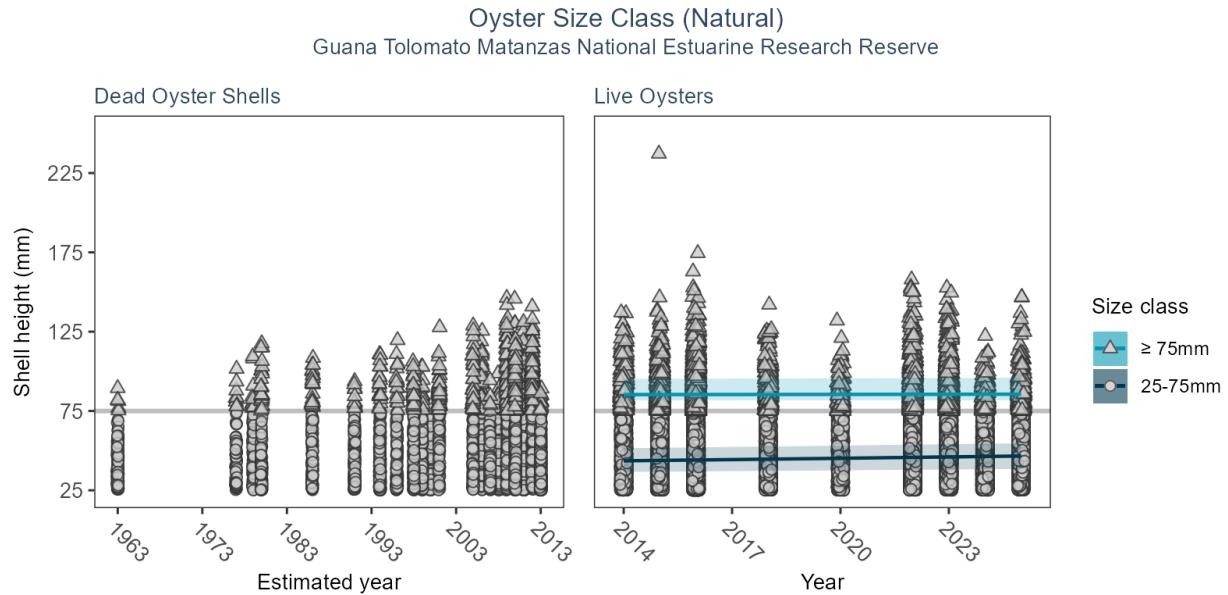
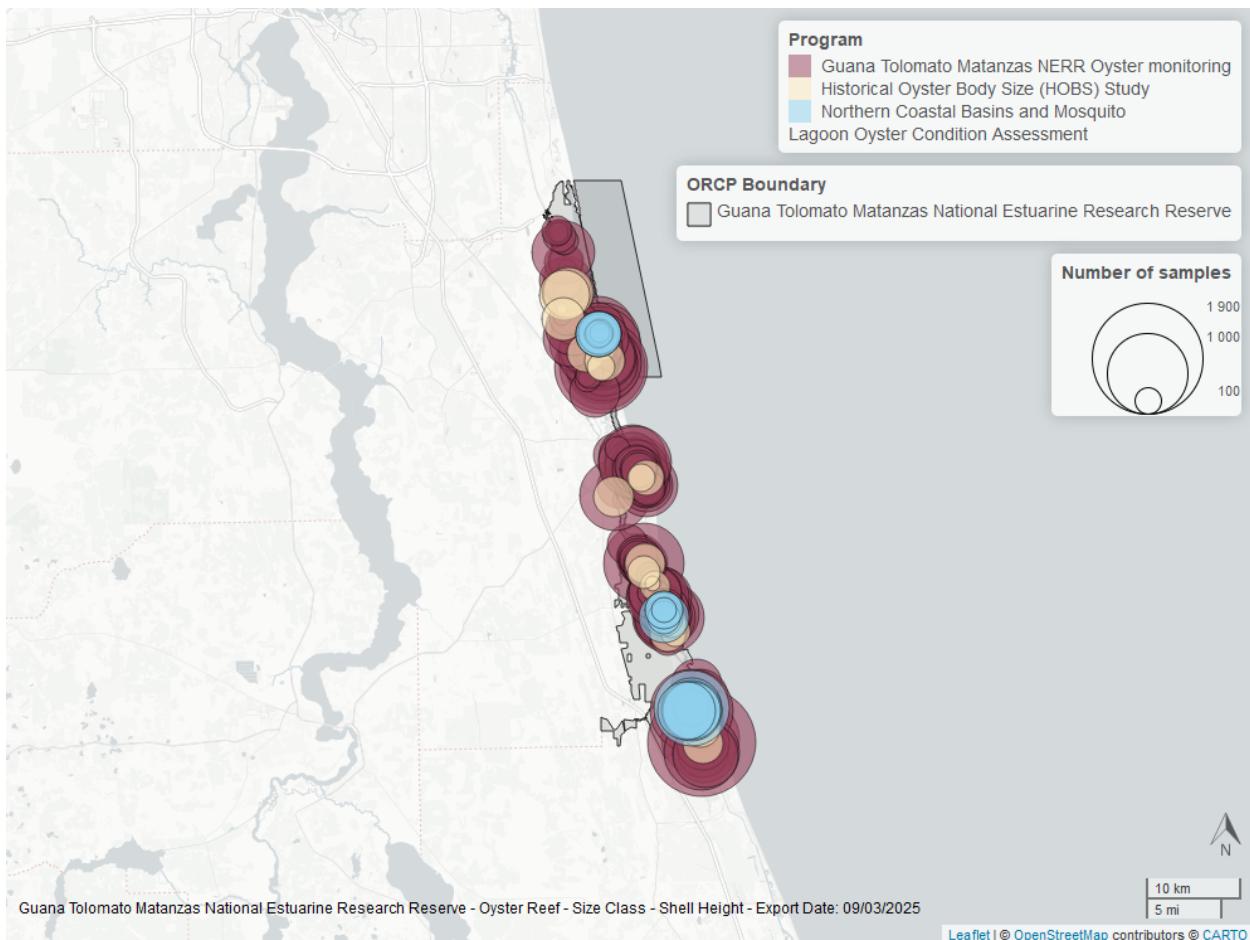


Table 53: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	0.20	0.35	-0.52 to 0.87
Live Oysters	Natural	No significant change	1.66	0.21	1.27 to 2.09

For natural reefs, annual average live oyster shell height in the 25-75mm and $\geq 75\text{mm}$ size classes increased by 1.66 mm per year and 0.2 mm per year, respectively. Models are not run on dead oyster shell measurements.



Indian River-Malabar to Vero Beach Aquatic Preserve Natural

Oyster Size Class (Natural)
Indian River-Malabar to Vero Beach Aquatic Preserve

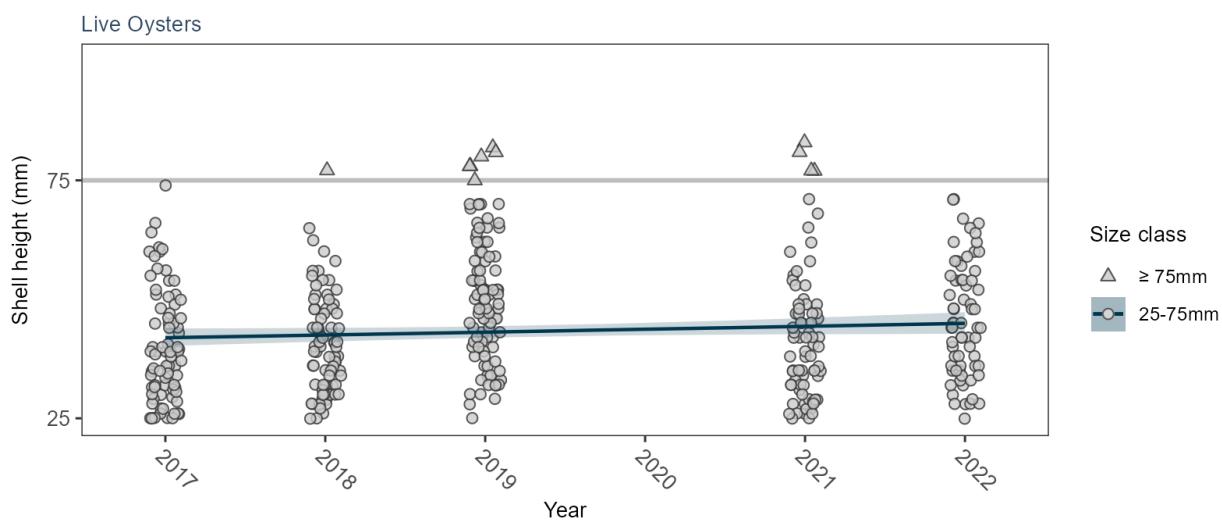
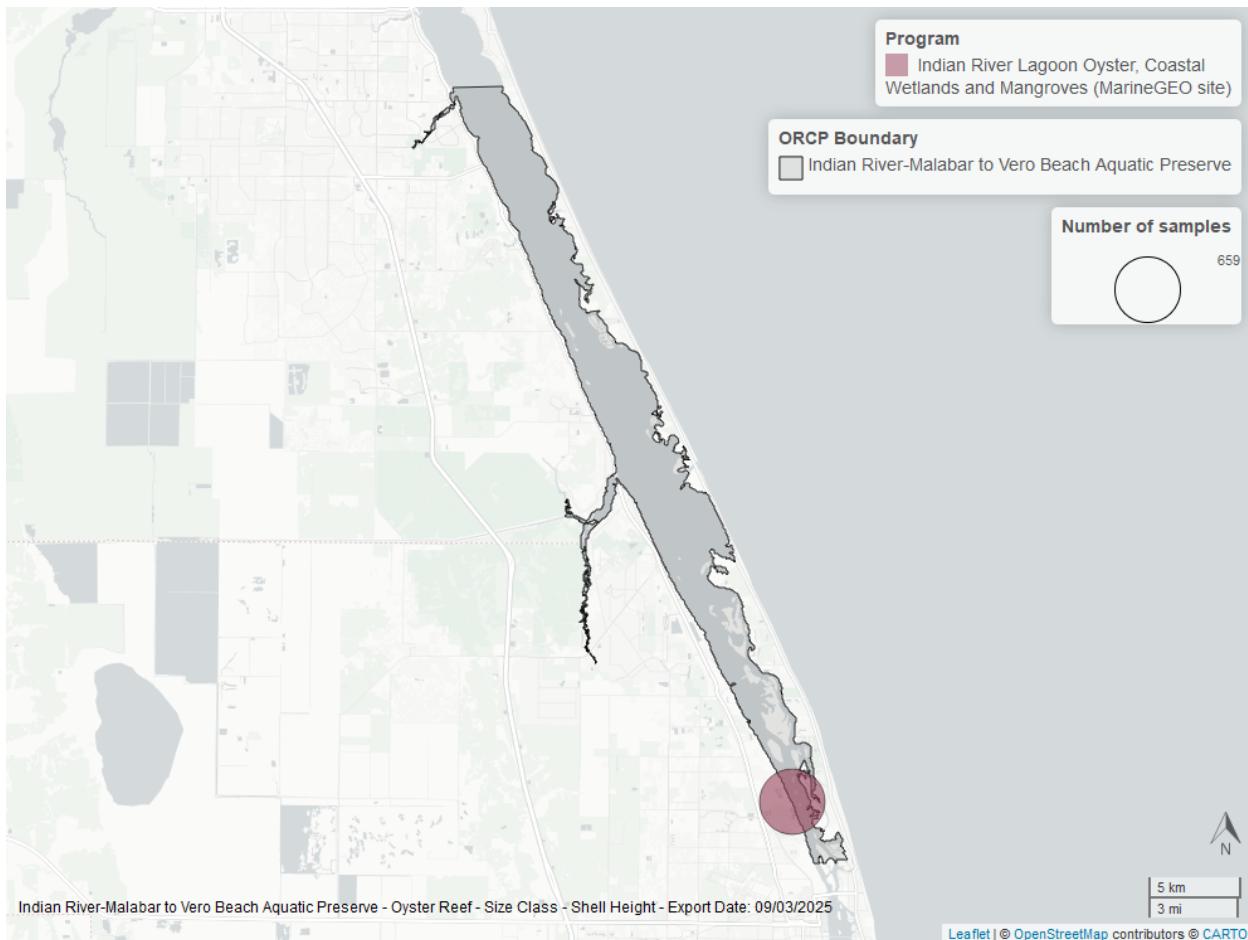


Table 54: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	1.52	0.95	-0.16 to 3.54

For natural reefs, annual average live oyster shell height in the 25-75mm size class increased by 1.52 mm per year, and there was insufficient data to calculate a trend for live oysters in the >=75mm size class.



Indian River-Vero Beach to Ft. Pierce Aquatic Preserve

Natural

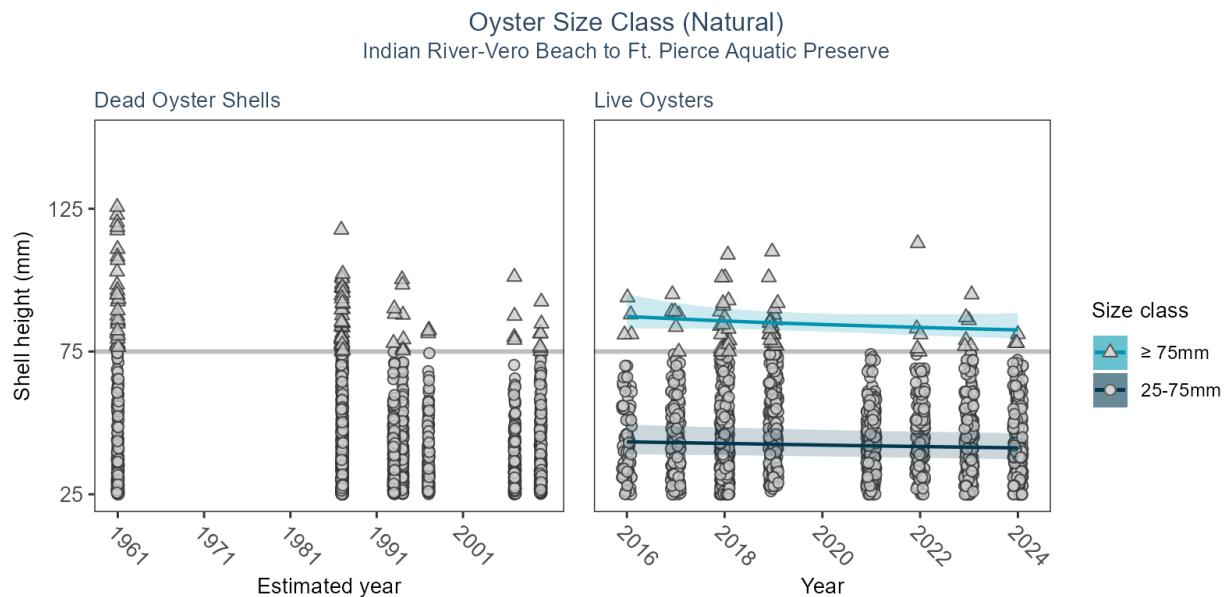
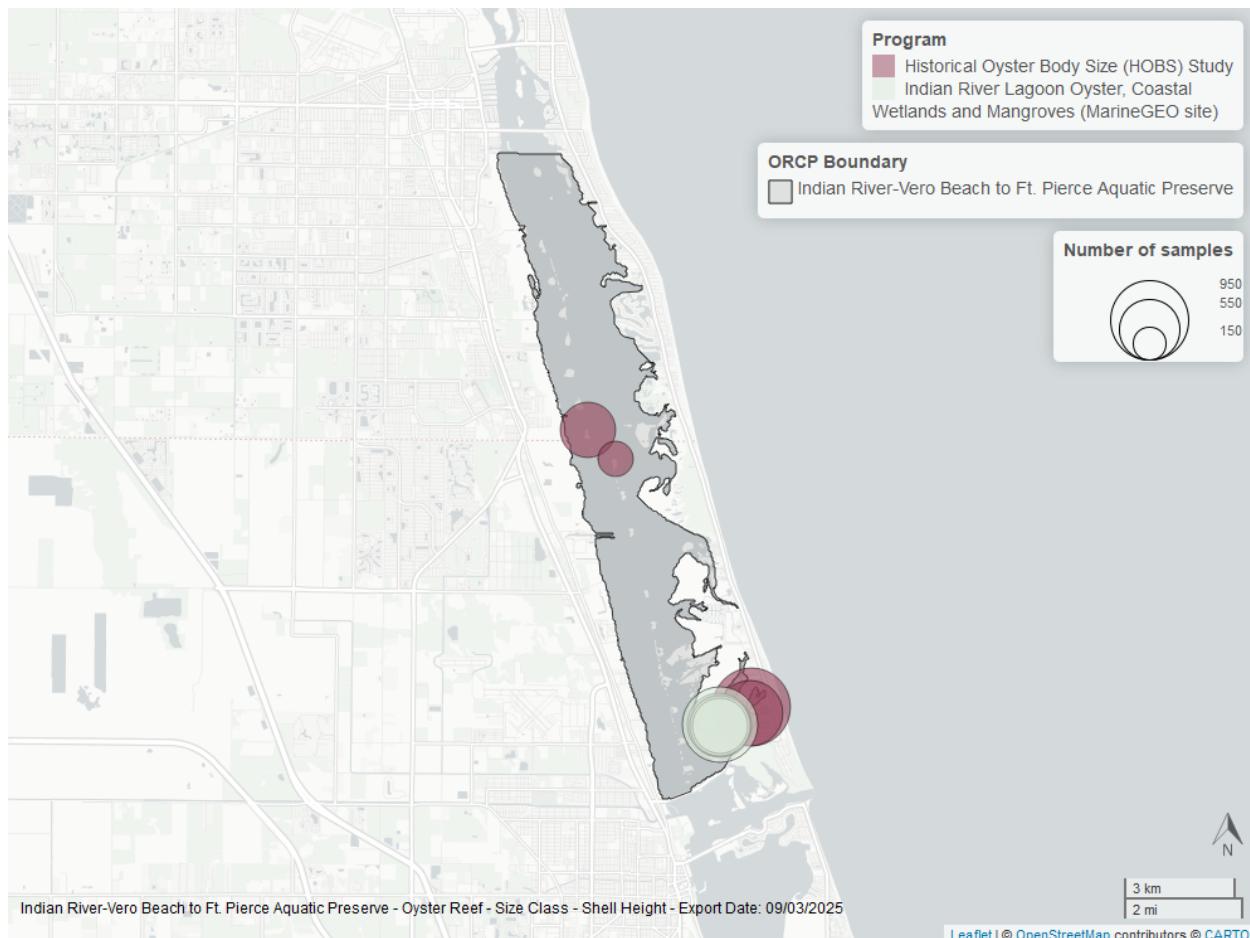


Table 55: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	-2.71	2.59	-8.62 to 1.51
Live Oysters	Natural	No significant change	-0.75	0.34	-1.44 to -0.07

For natural reefs, annual average live oyster shell height in the 25-75mm and $\geq 75\text{mm}$ size classes decreased by 0.75 mm per year and 2.71 mm per year, respectively. Models are not run on dead oyster shell measurements.



Jensen Beach to Jupiter Inlet Aquatic Preserve

Natural

Oyster Size Class (Natural)
Jensen Beach to Jupiter Inlet Aquatic Preserve

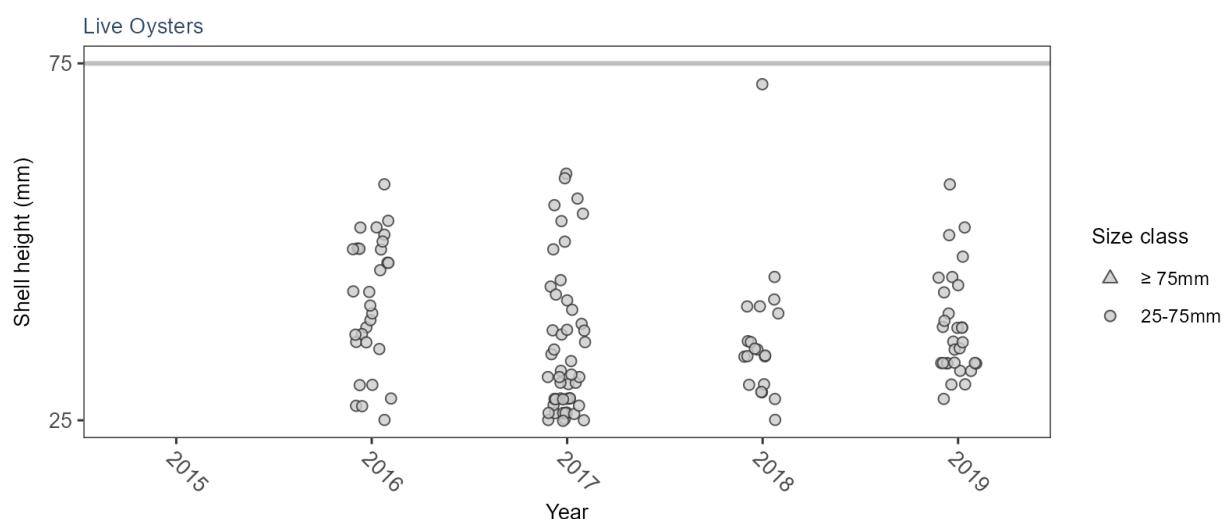
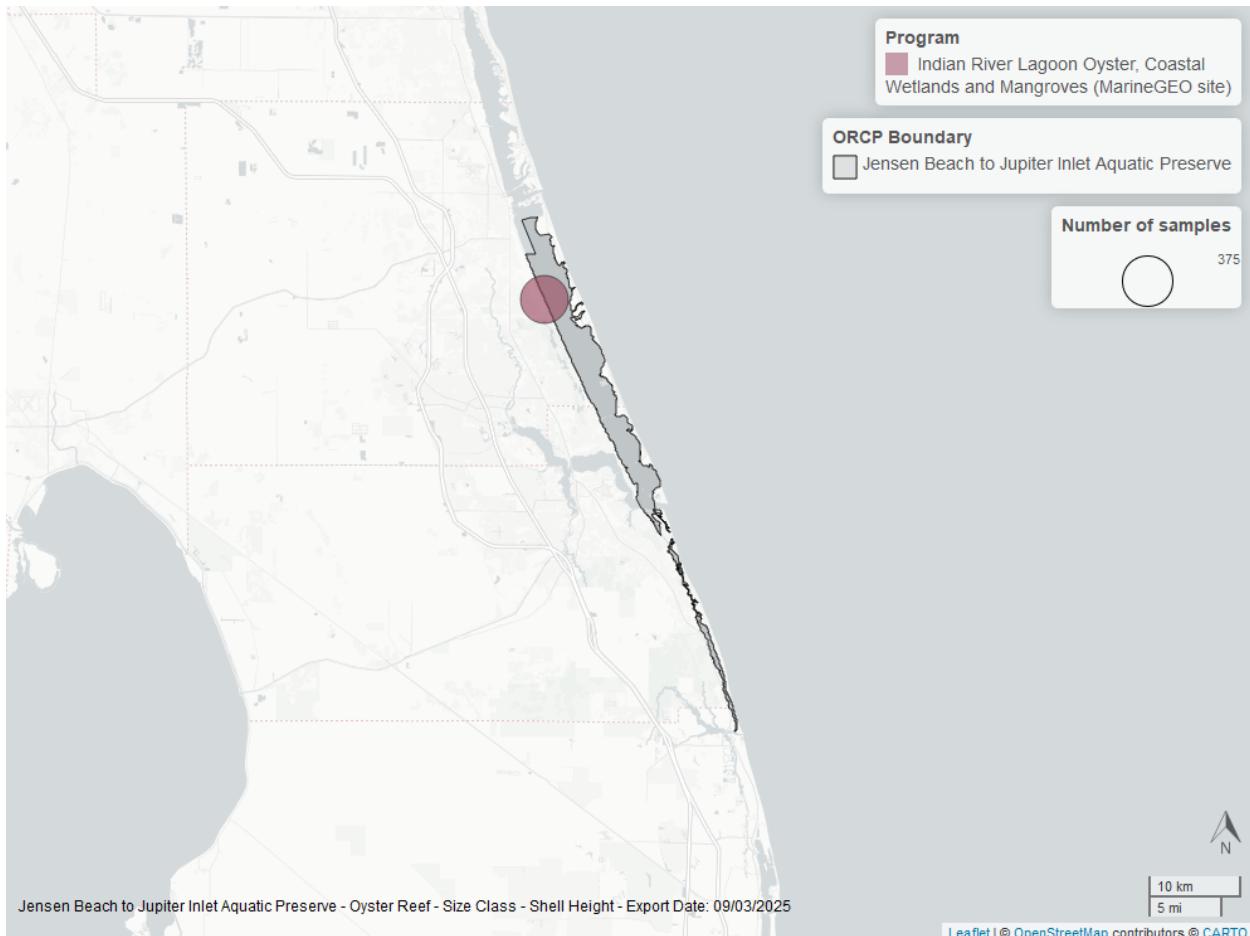


Table 56: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in the 25-75mm size class.



Lemon Bay Aquatic Preserve

Natural

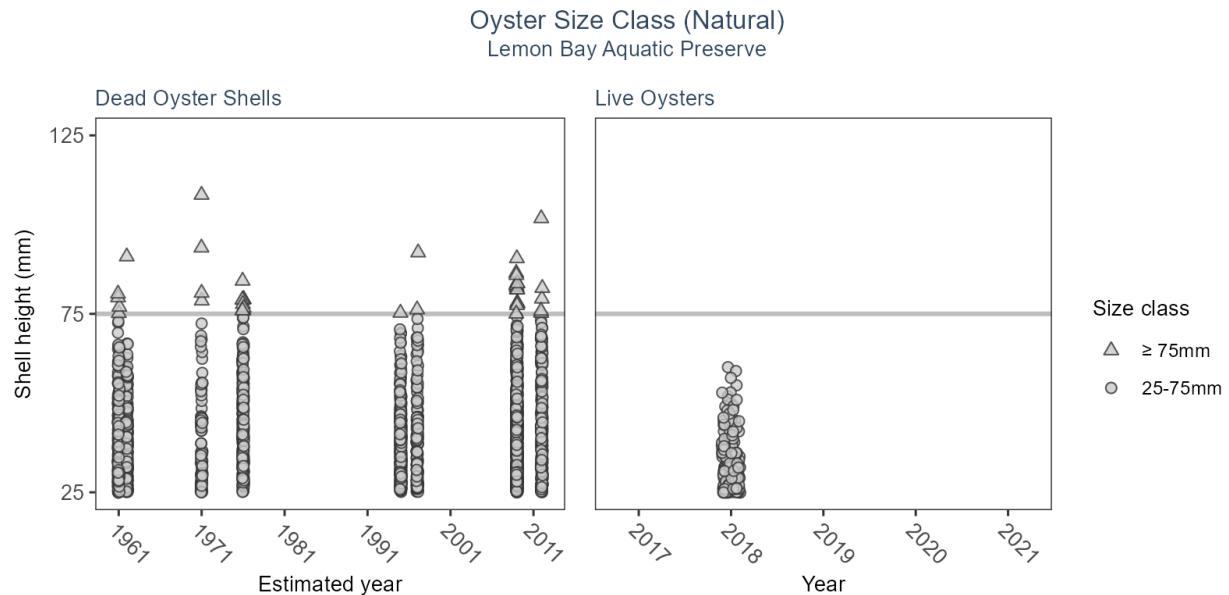
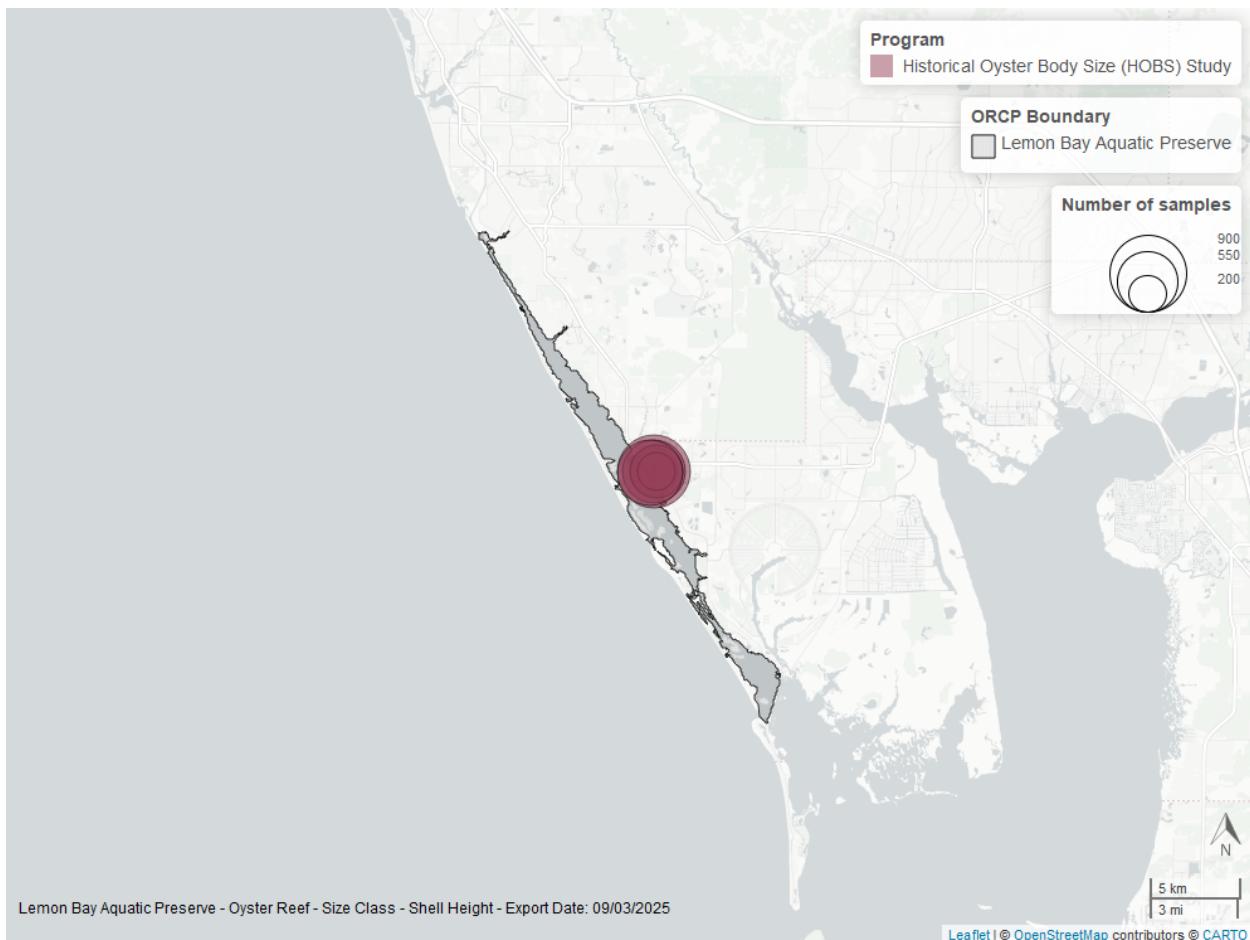


Table 57: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in the 25-75mm size class. Models are not run on dead oyster shell measurements.



Loxahatchee River-Lake Worth Creek Aquatic Preserve Natural

Oyster Size Class (Natural)
Loxahatchee River-Lake Worth Creek Aquatic Preserve

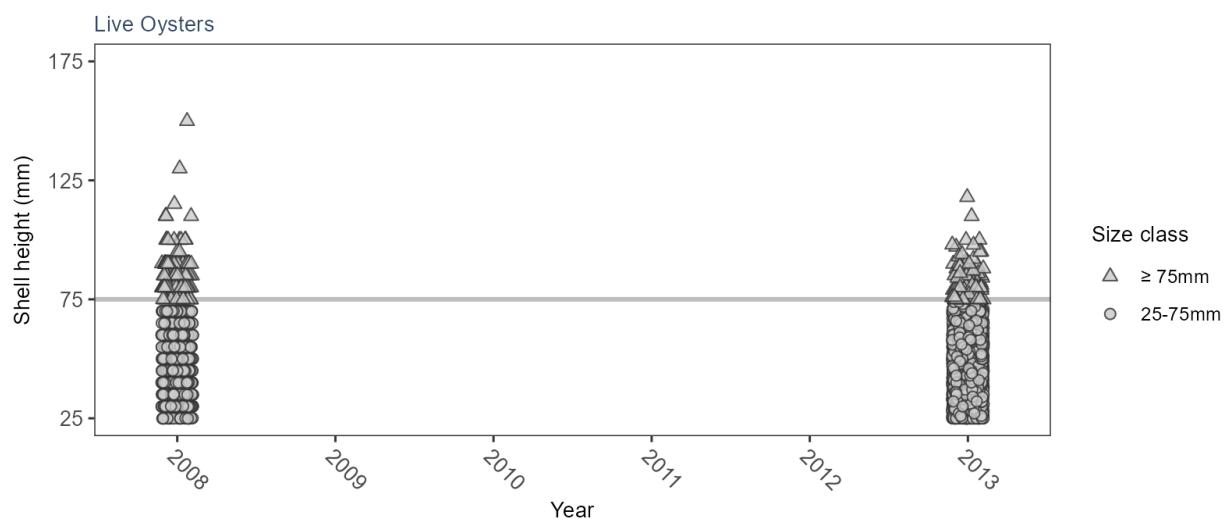
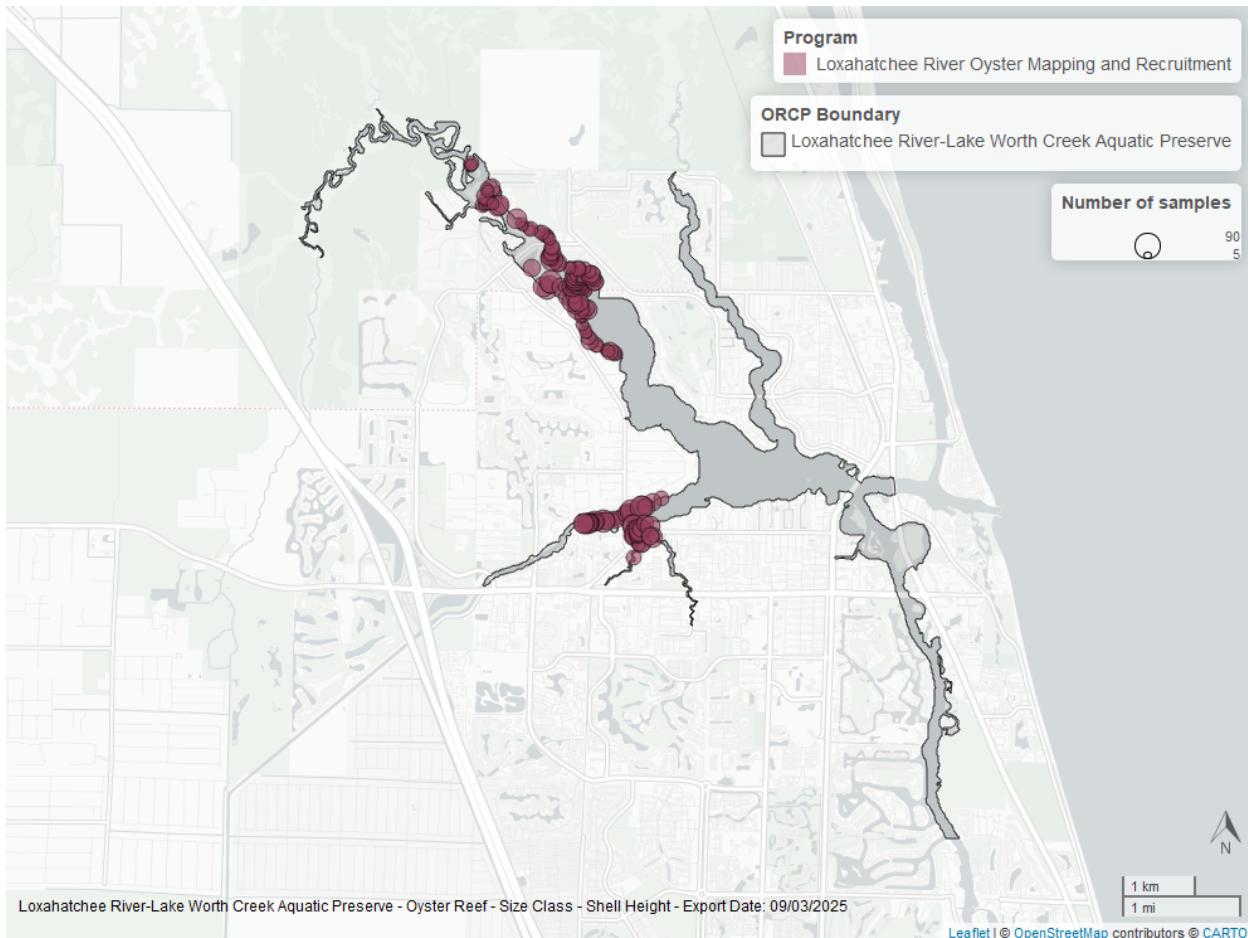


Table 58: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Mosquito Lagoon Aquatic Preserve

Natural

Oyster Size Class (Natural)
Mosquito Lagoon Aquatic Preserve

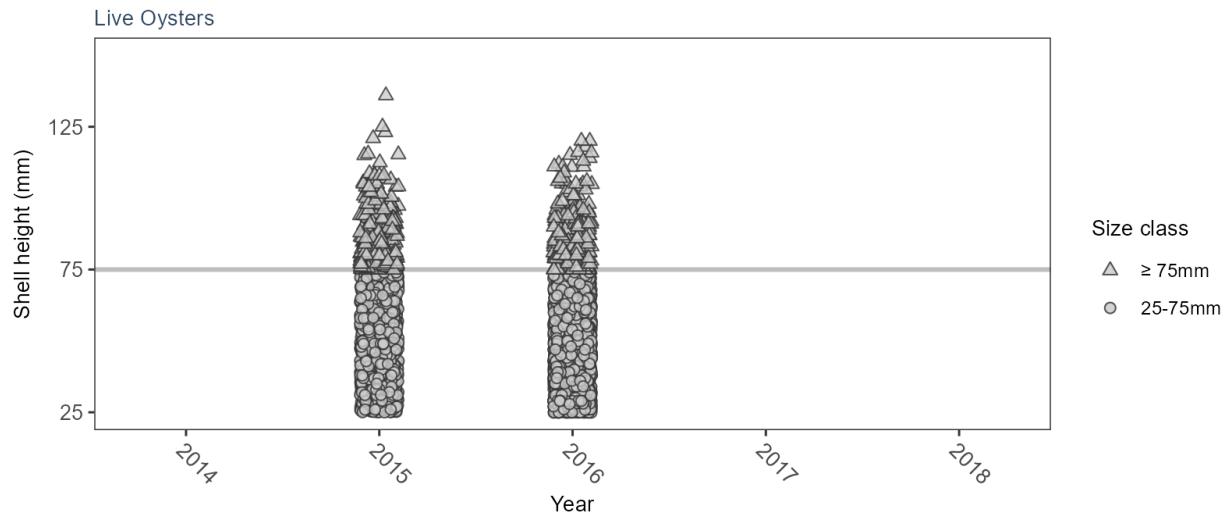
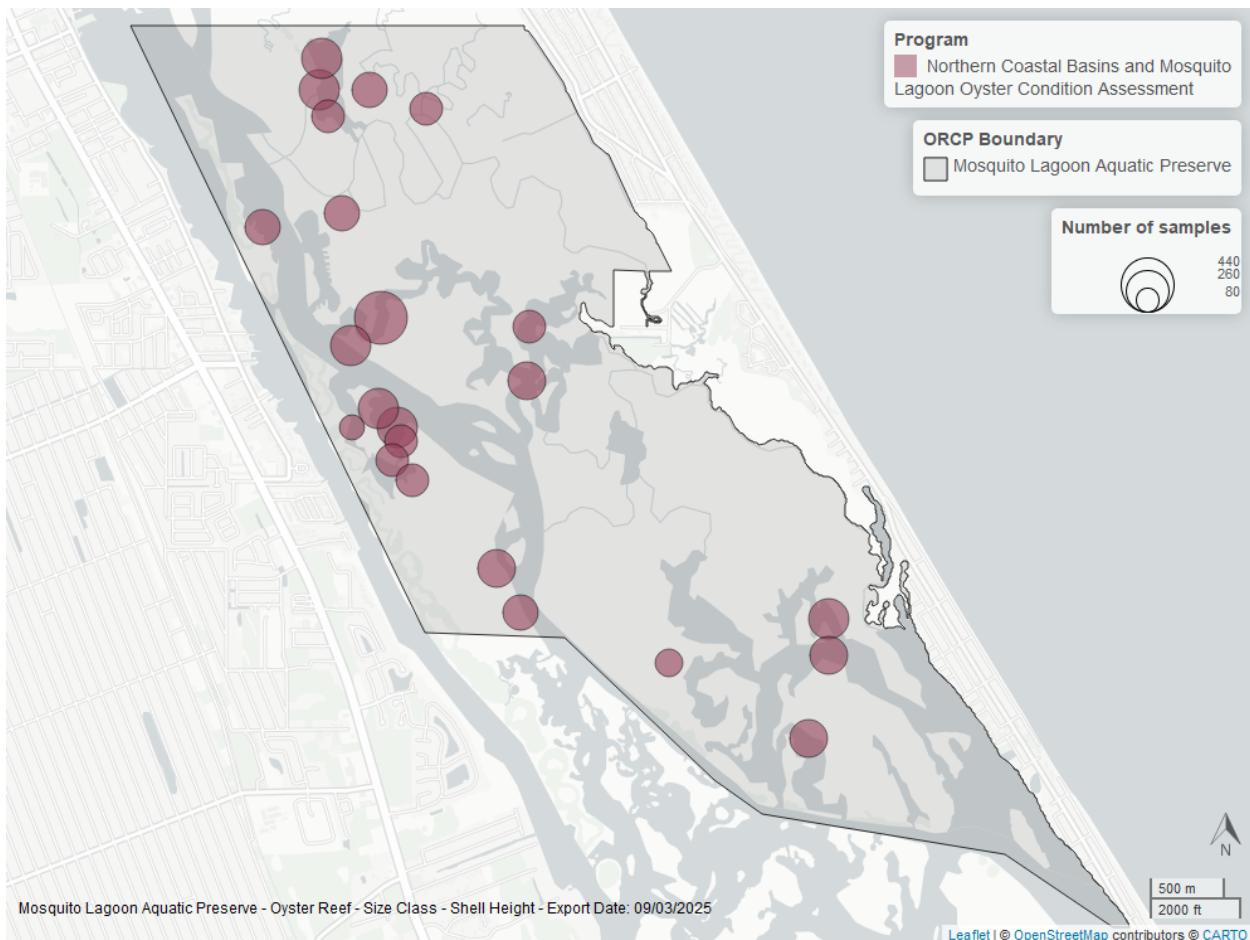


Table 59: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Nassau River-St. Johns River Marshes Aquatic Preserve

Natural

Oyster Size Class (Natural)
Nassau River-St. Johns River Marshes Aquatic Preserve

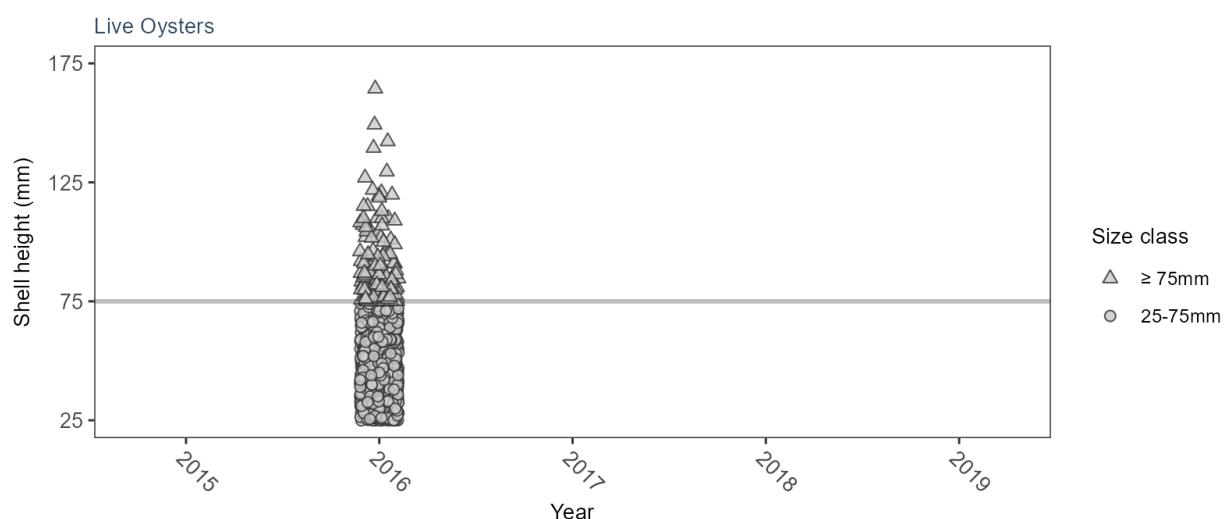
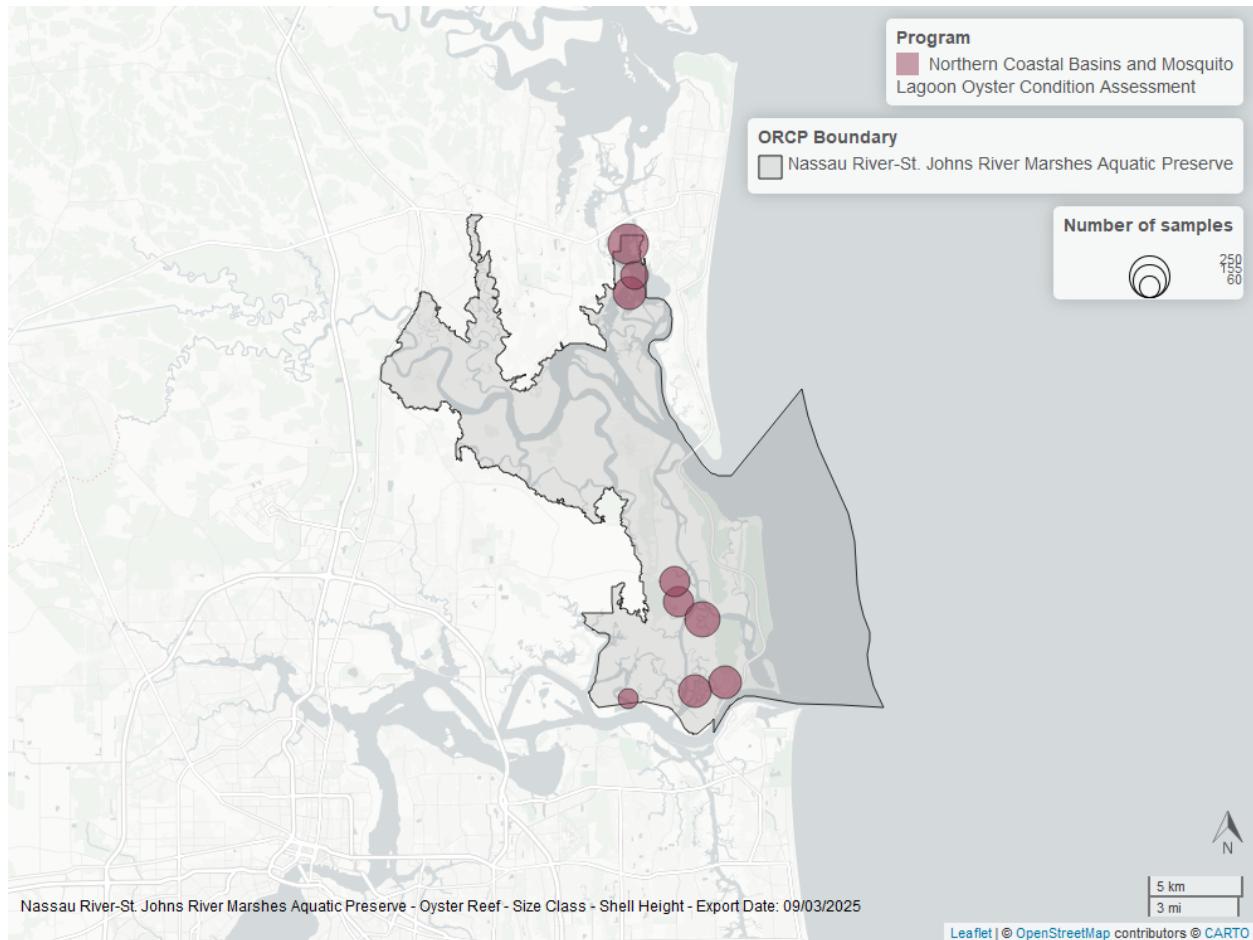


Table 60: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Nature Coast Aquatic Preserve

Natural

Oyster Size Class (Natural)
Nature Coast Aquatic Preserve

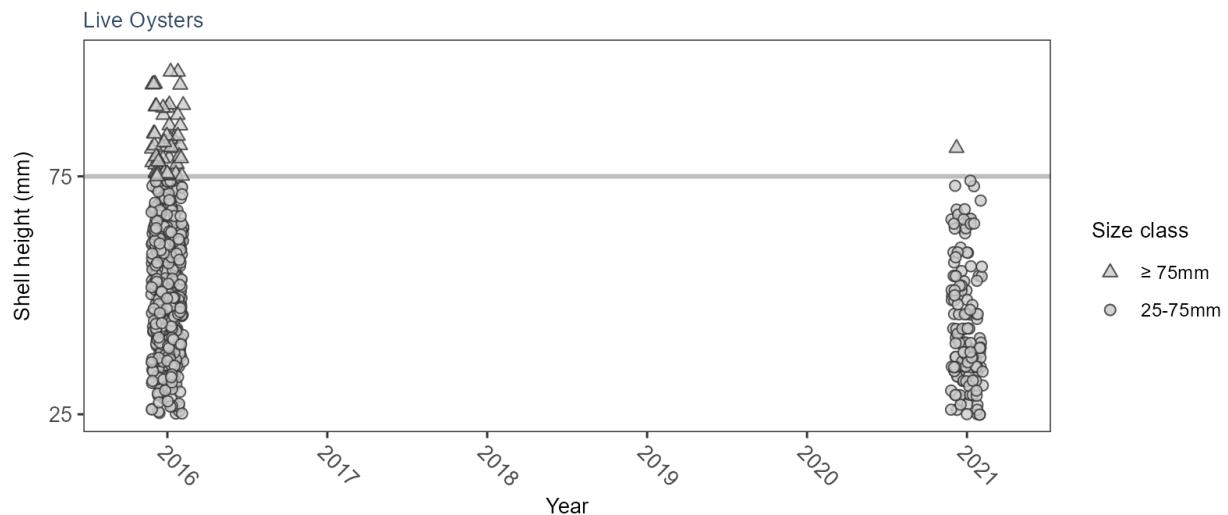
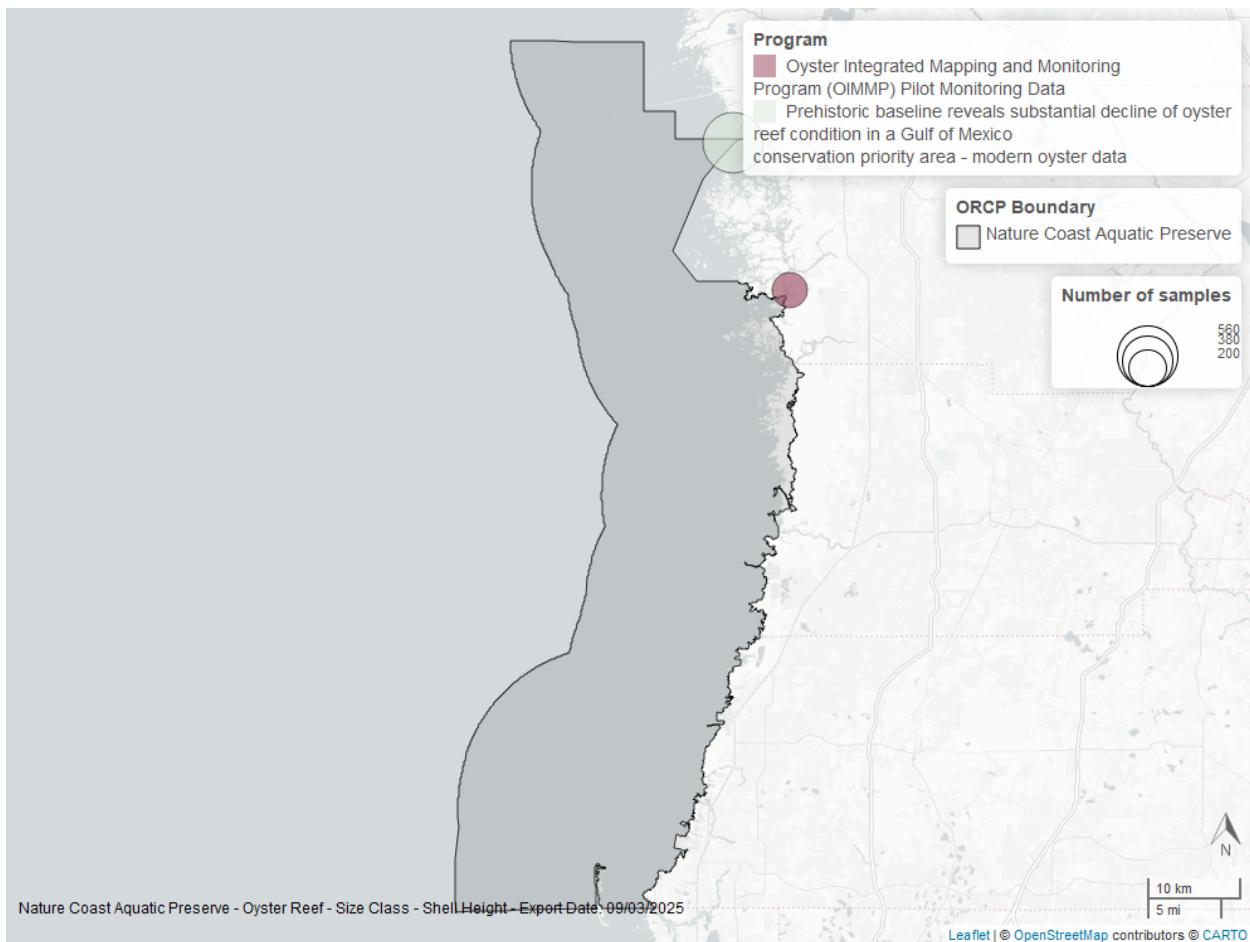


Table 61: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Pine Island Sound Aquatic Preserve

Natural

Oyster Size Class (Natural)
Pine Island Sound Aquatic Preserve

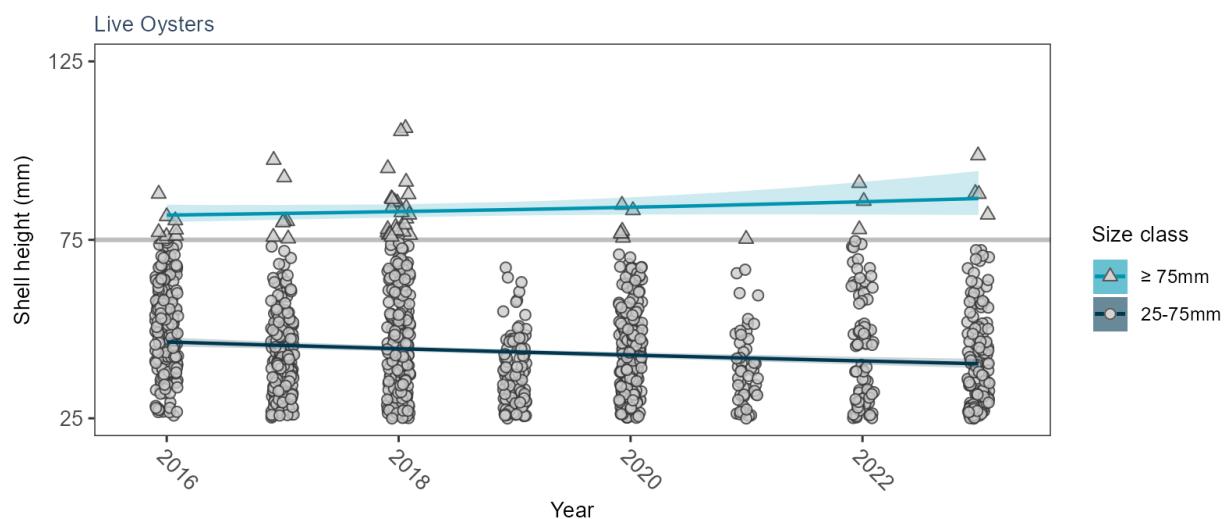


Table 62: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	No significant change	2.37	2.01	-1.18 to 6.83
Live Oysters	Natural	No significant change	-3.88	1.12	-6.54 to -2.14

For natural reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 2.37 mm per year, and it decreased by 3.88mm per year in the 25-75mm size class. For restored reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 10.38 mm per year, and it decreased by 8.81mm per year in the 25-75mm size class.

Restored

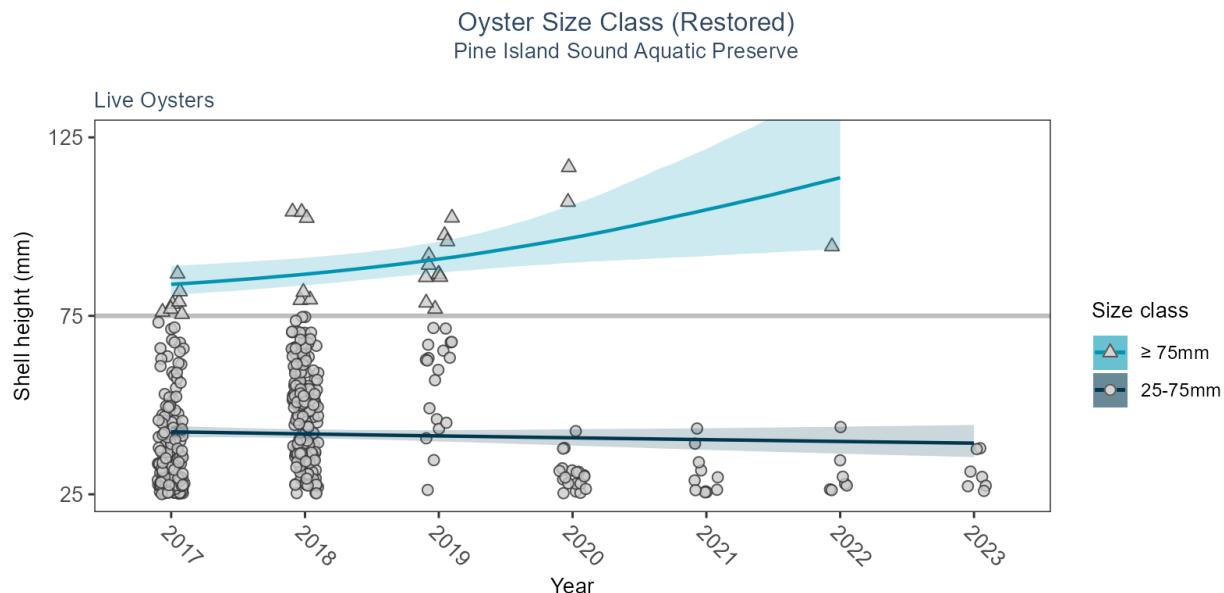
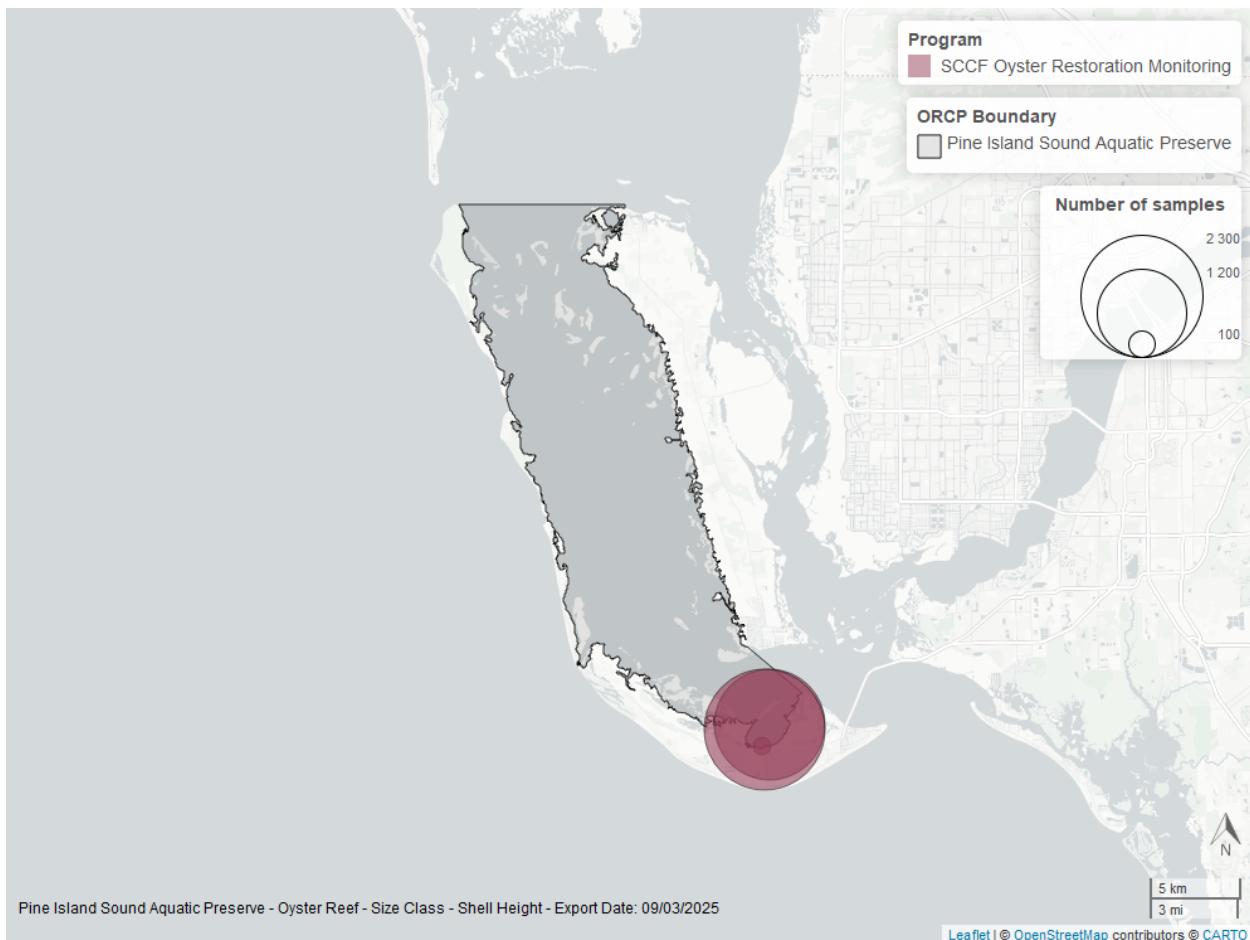


Table 63: Model results for Oyster Shell Height - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	-	-	-	-
Live Oysters	Restored	No significant change	10.38	4.62	3.05 to 21.32
Live Oysters	Restored	No significant change	-8.81	10.55	-34.64 to 6.17

For natural reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 2.37 mm per year, and it decreased by 3.88mm per year in the 25-75mm size class. For restored reefs, annual average live oyster shell height in the $\geq 75\text{mm}$ size class increased by 10.38 mm per year, and it decreased by 8.81mm per year in the 25-75mm size class.



Pinellas County Aquatic Preserve

Natural

Oyster Size Class (Natural)
Pinellas County Aquatic Preserve

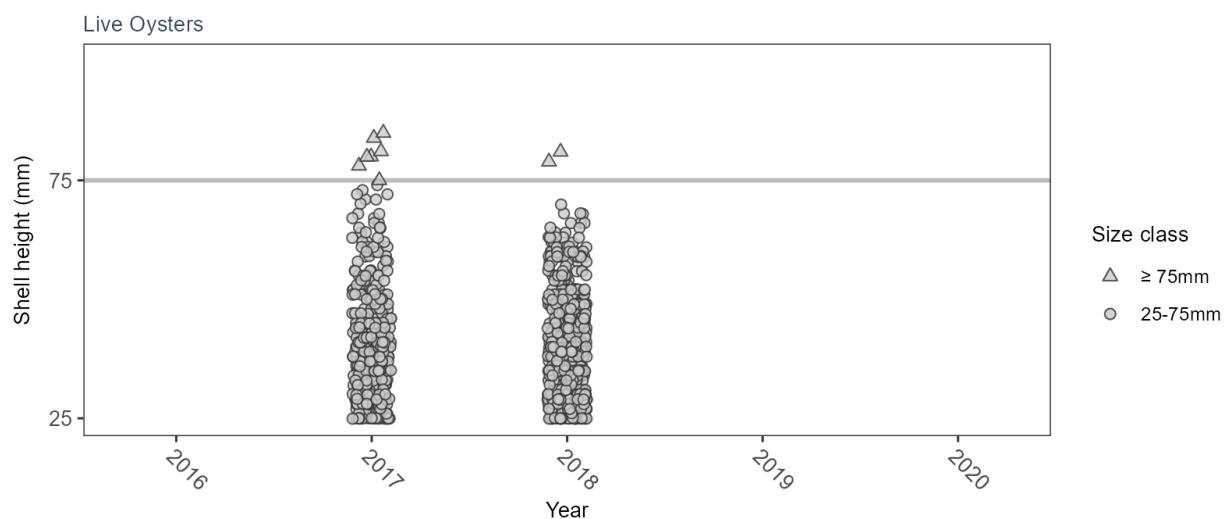
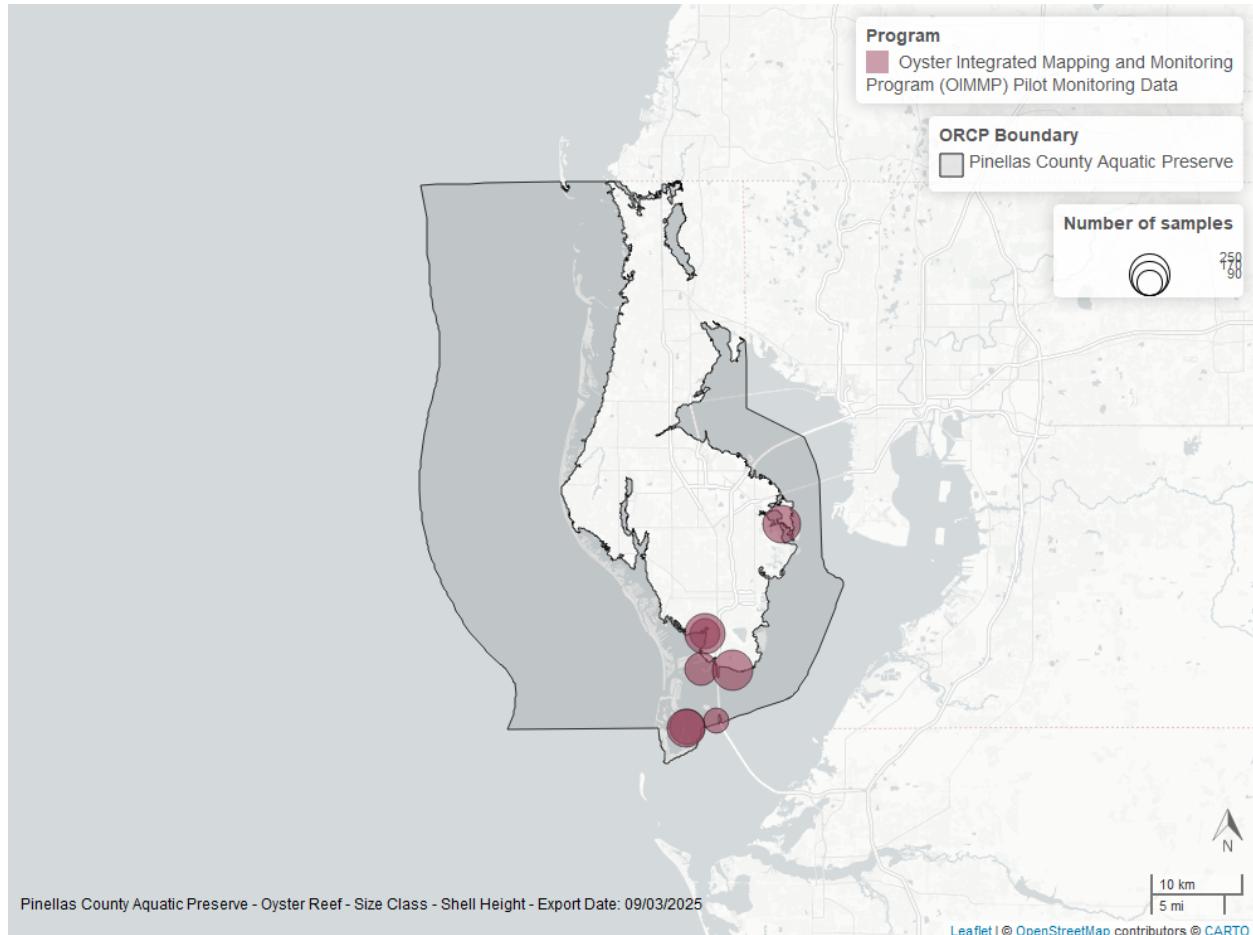


Table 64: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



St. Martins Marsh Aquatic Preserve

Natural

Oyster Size Class (Natural)
St. Martins Marsh Aquatic Preserve

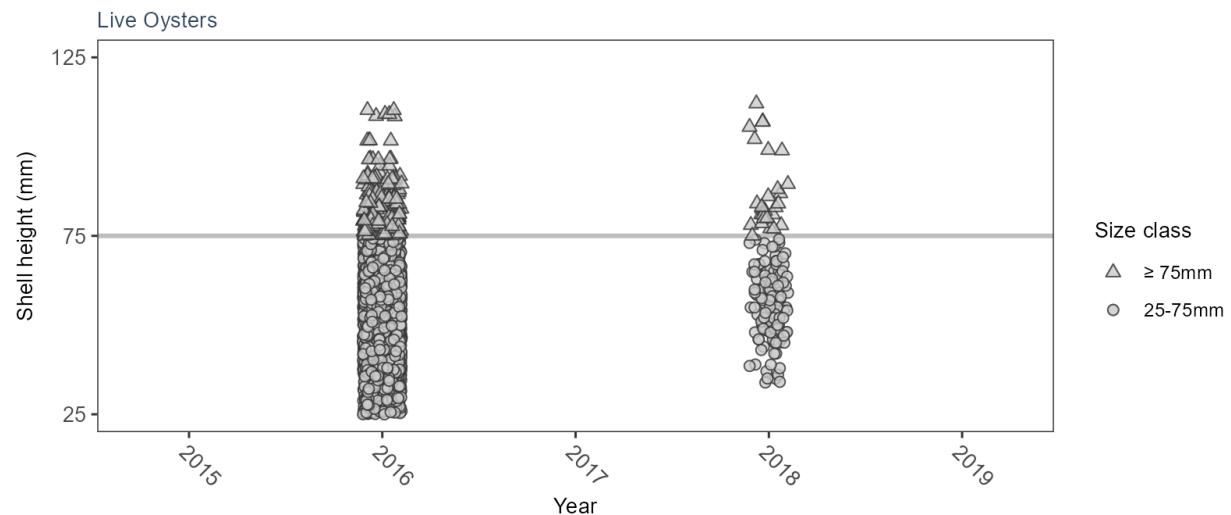
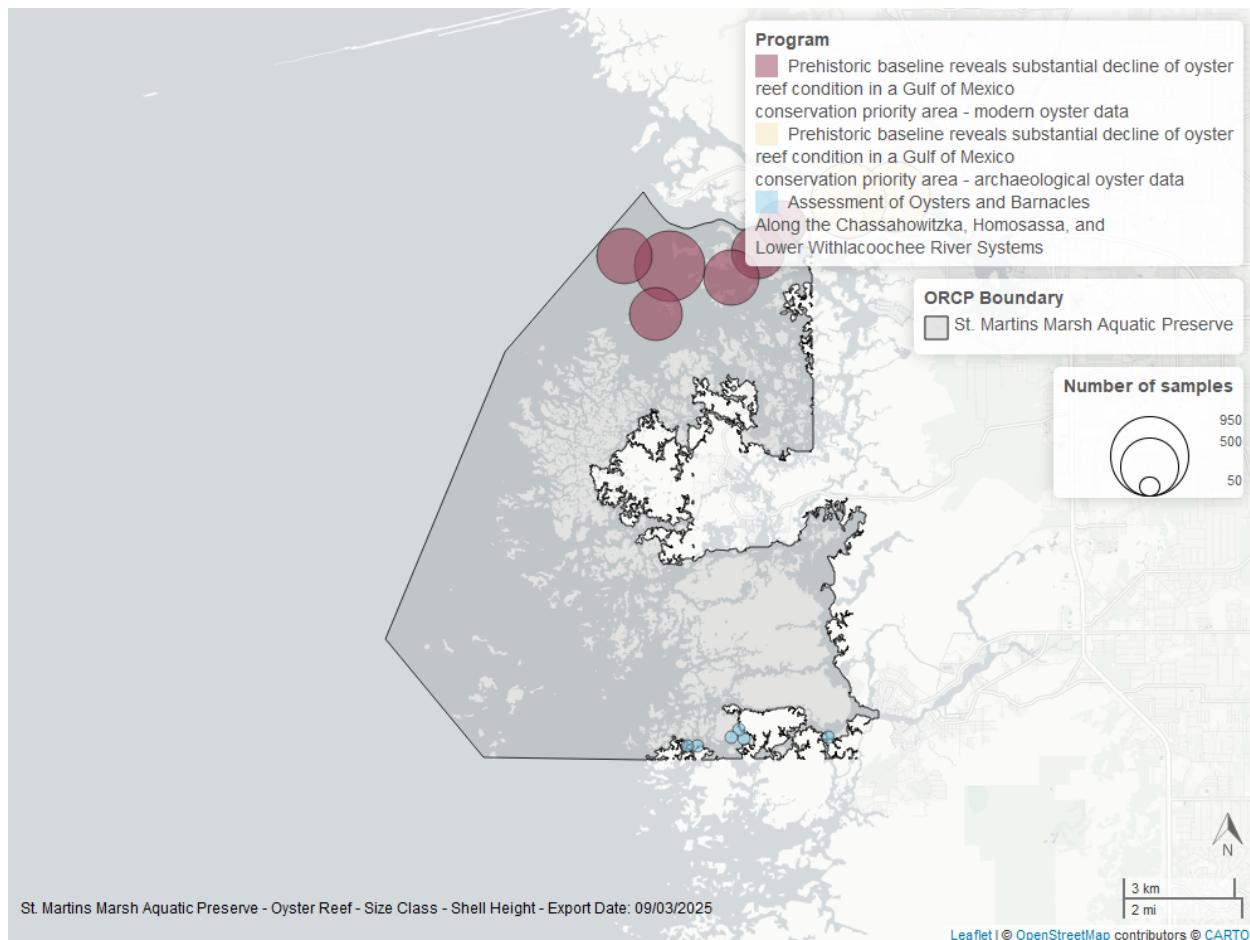


Table 65: Model results for Oyster Shell Height - Natural

<i>Shell Type</i>	<i>Habitat Type</i>	<i>Trend Status</i>	<i>Estimate</i>	<i>Standard Error</i>	<i>Credible Interval</i>
Dead Oyster Shells	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the $\geq 75\text{mm}$ size class.



Yellow River Marsh Aquatic Preserve

Natural

Oyster Size Class (Natural)
Yellow River Marsh Aquatic Preserve

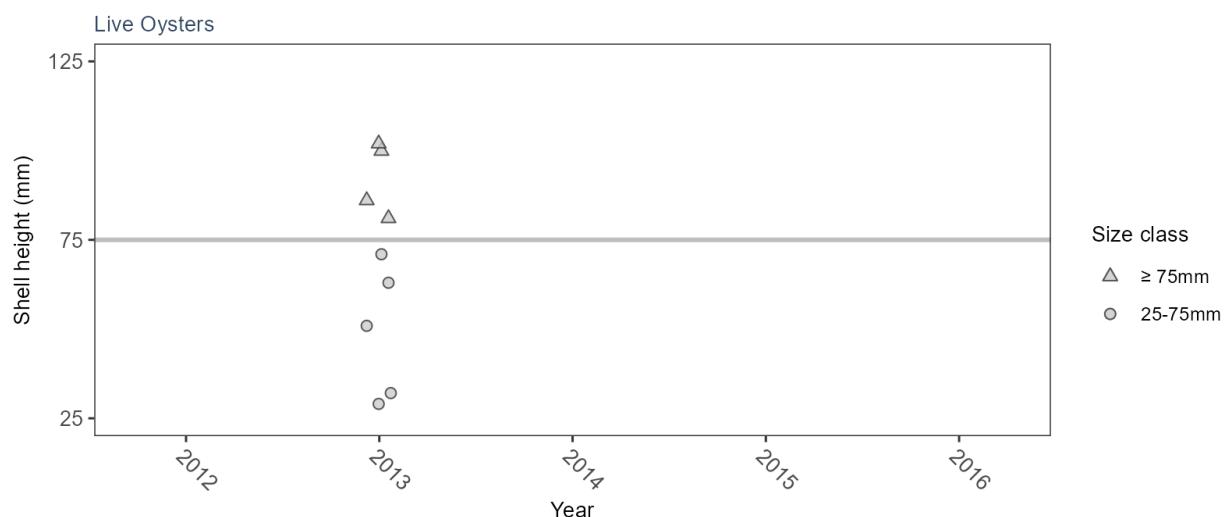


Table 66: Model results for Oyster Shell Height - Natural

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-
Live Oysters	Natural	-	-	-	-

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the ≥ 75 mm size class. For restored reefs, annual average live oyster shell height in the 25-75mm size class increased by 3.19 mm per year, and there was insufficient data to calculate a trend for live oysters in the ≥ 75 mm size class.

Restored

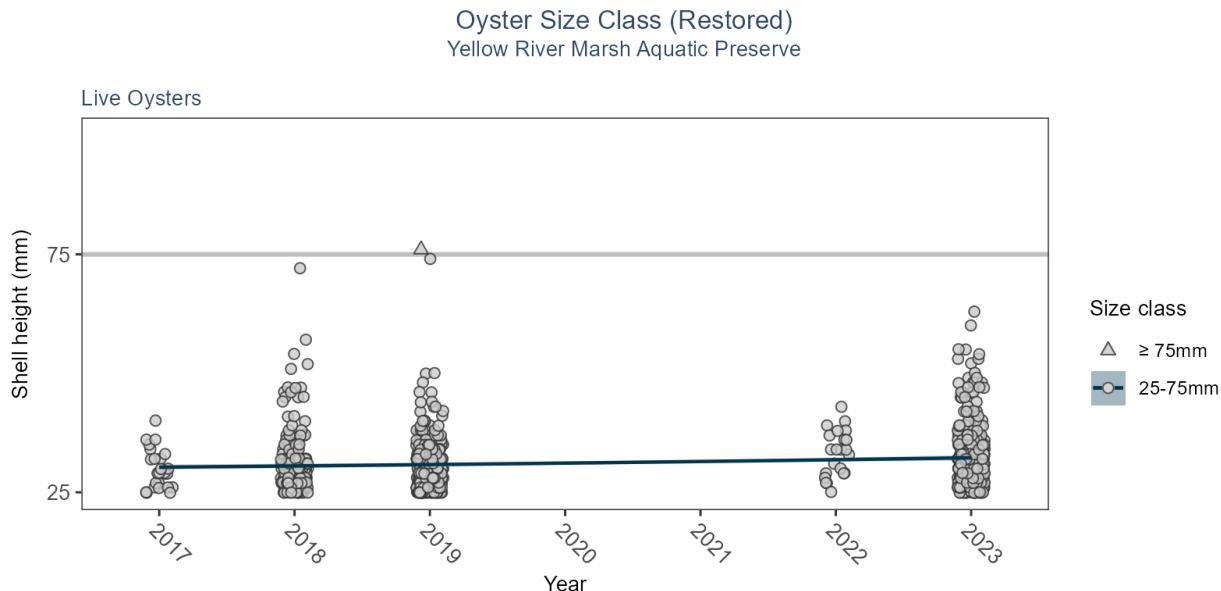
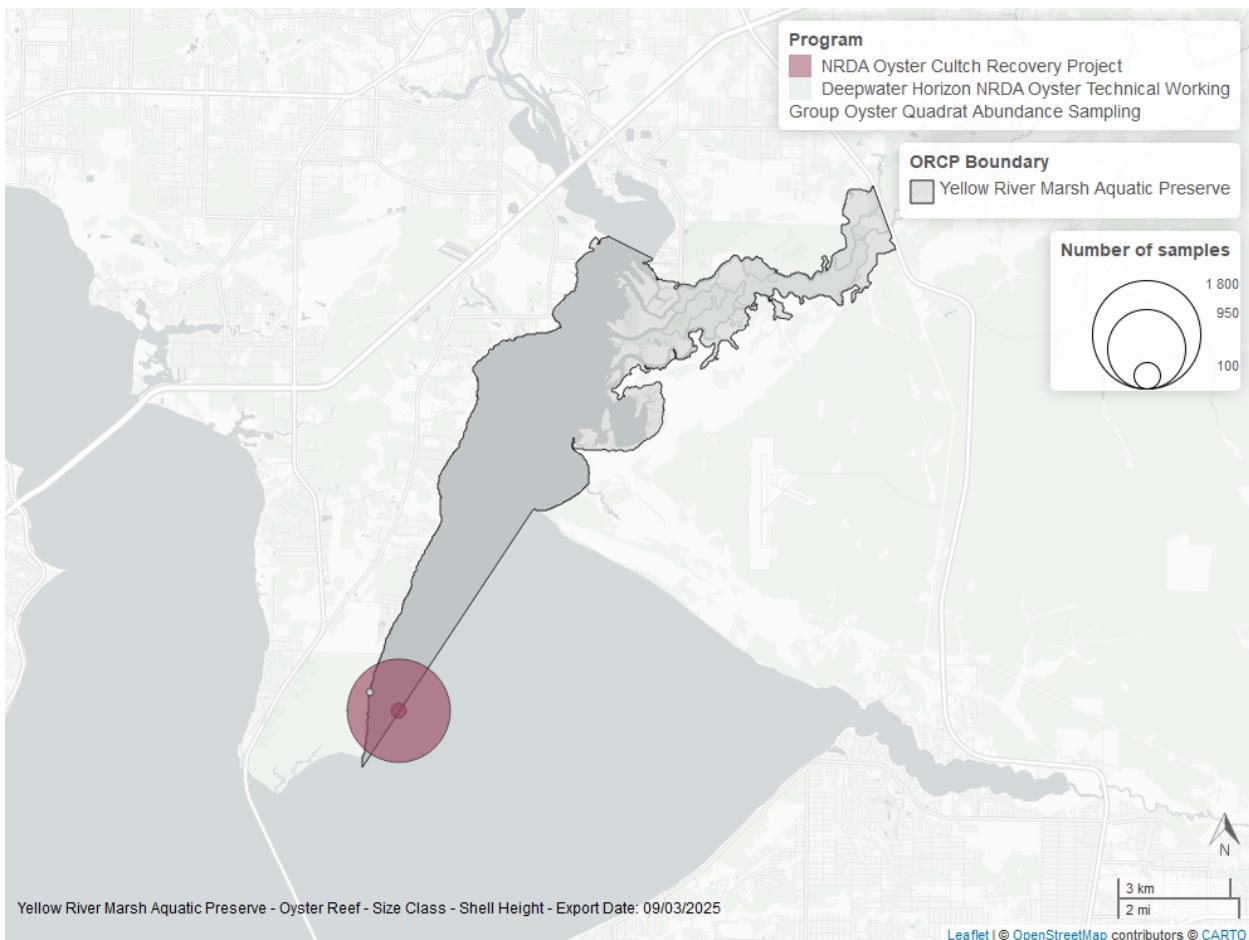


Table 67: Model results for Oyster Shell Height - Restored

Shell Type	Habitat Type	Trend Status	Estimate	Standard Error	Credible Interval
Live Oysters	Restored	-	-	-	-
Live Oysters	Restored	-	-	-	-
Live Oysters	Restored	No significant change	3.19	1.22	1.47 to 6.2

For natural reefs, there was insufficient data to calculate a trend for live oysters in either the 25-75mm or the ≥ 75 mm size class. For restored reefs, annual average live oyster shell height in the 25-75mm size class increased by 3.19 mm per year, and there was insufficient data to calculate a trend for live oysters in the ≥ 75 mm size class.



Libraries and Settings

Loads libraries used in the script. Loads the Segoe UI font for use in the figures. The inclusion of `scipen` option limits how frequently R defaults to scientific notation. Sets default settings for displaying warning and messages in created document, and sets figure dpi. Includes `knitr` and `kable` settings used to render this report. The code outlined below comes from `Oyster_Models_Clean_parallel.R` which performs data filtering and model fitting and can be run in parallel for more efficient results.

```
library(brms)
library(Rmisc)
library(rstan)
library(rstantools)
library(stringr)
library(data.table)
library(sf)
library(tidyverse)
library(doFuture)
library(tictoc)
library(doRNG)
library(rstudioapi)
library(ggpubr)
library(lubridate)
library(tidyverse)
```

```

library(data.table)
library(tictoc)
library(glue)
library(kableExtra)
library(future)
library(future.apply)
options(scipen=999)
knitr::opts_chunk$set(
  warning=FALSE,
  message=FALSE,
  dpi=200,
  fig.pos = 'H'
)
options(kableExtra.auto_format = FALSE)
options(knitr.kable.NA = '-')

```

File Import

Imports file that is determined in the Oyster_ReportRender.R script.

The command `fread` is used because of its improved speed while handling large data files. Updates column names.

The latest version of Oyster data is available at: <https://usf.box.com/s/6dtcerdkrte33fqw5kl5fjo6guo0anrw>

The file being used for the analysis is: **All_OYSTER_Parameters-2025-Sep-04.txt**

```

# If QAQCPlots is TRUE, plots will be created without running models
# Set to FALSE to run models
QAQCPlots <- FALSE

wd <- dirname(getActiveDocumentContext()$path)
setwd(wd)

# Set output directory
out_dir <- "output"

file_in <- str_subset(
  list.files("C:/SEACAR Data/SEACARdata/", full.names = TRUE), "OYSTER")

oysterraw <- fread(file_in, sep="|", na.strings=c("NULL"))
# New OIMMP updates include samples without MA associations, exclude for these analyses currently
oysterraw <- oysterraw[!is.na/AreaID], ]

oyster <- copy(oysterraw)
oysterraw2 <- tidyr::pivot_wider(oysterraw, names_from="ParameterName",
                                   values_from="ResultValue")
setDT(oysterraw2)
setnames(oysterraw2, c("Density", "Percent Live", "Shell Height",
                      "Number of Oysters Counted - Live",
                      "Number of Oysters Counted - Dead",
                      "Number of Oysters Counted - Total", "Reef Height"),
         c("Density_m2", "PercentLive_pct", "ShellHeight_mm",
           "Number_of_Oysters_Counted_Live_Count",
           "Number_of_Oysters_Counted_Dead_Count",

```

```

  "Number_of_Oysters_Counted_Total_Count",
  "ReefHeight_mm"))
oysterraw2[, ObsIndex := seq(1:nrow(oysterraw2))]

oysterraw <- oysterraw2
rm(oysterraw2)

```

Data Setup & Filtering

Documentation on database filtering is provided here: [SEACAR Documentation- Analysis Filters and Calculations.docx](#)

Identifies and removes outliers in the data and various programs.

```

#Make sure column formats are correct-I am still getting an "NAs introduced
#by coercion" warning on the LiveDate calculation,
#but I'm not sure what is going on because when I spot-check the output,
#it does not look like it is introducing NAs...

oysterraw[, `:=` (RowID=as.integer(RowID),
                  ProgramID=as.integer(ProgramID),
                  LocationID=as.integer(LocationID),
                  ProgramName=as.character(ProgramName),
                  ProgramLocationID=as.character(ProgramLocationID),
                  QuadIdentifier=as.character(QuadIdentifier),
                  ReefIdentifier=as.character(ReefIdentifier),
                  UniversalReefID=as.factor(UniversalReefID),
                  LiveDate=as.integer(ifelse(!is.na(LiveDate_Qualifier) &
                                              str_detect(LiveDate,
                                                         "....-...-.."),
                                              paste0(str_sub(LiveDate, 1, 4)),
                                              round(as.numeric(LiveDate)))),
                  LiveDate_Qualifier=as.character(LiveDate_Qualifier),
                  LiveDate_MinEstDate=as.numeric(LiveDate_MinEstDate),
                  LiveDate_MaxEstDate=as.numeric(LiveDate_MaxEstDate),
                  SampleAge_Stdev=as.numeric(SampleAge_Stdev),
                  #GISUniqueID=as.logical(GISUniqueID),
                  Year=as.integer(Year),
                  Month=as.integer(Month),
                  ManagedAreaName=as.character(ManagedAreaName),
                  SurveyMethod=as.character(SurveyMethod),
                  PercentLiveMethod=as.character(PercentLiveMethod),
                  HabitatClassification=as.character(HabitatClassification),
                  MinimumSizeMeasured_mm=as.character(MinimumSizeMeasured_mm),
                  NumberMeasured_n=as.character(NumberMeasured_n),
                  QuadSize_m2=as.numeric(QuadSize_m2),
                  MADUp=as.integer(MADUp),
                  Density_m2=as.numeric(Density_m2),
                  PercentLive_pct=as.numeric(PercentLive_pct),
                  ShellHeight_mm=as.numeric(ShellHeight_mm),
                  Number_of_Oysters_Counted_Total_Count =
                    as.integer(Number_of_Oysters_Counted_Total_Count),
                  Number_of_Oysters_Counted_Live_Count =
                    as.integer(Number_of_Oysters_Counted_Live_Count),
                  Number_of_Oysters_Counted_Dead_Count =

```

```

        as.integer(Number_of_Oysters_Counted_Dead_Count),
        ObsIndex=as.integer(ObsIndex))]

#Calculate Density_m2 values for ProgramID==4016 & 4042
oysterraw[ProgramID==4016, Density_m2 :=  

           Number_of_Oysters_Counted_Live_Count/as.numeric(QuadSize_m2)]
oysterraw[ProgramID==4042 & !is.na(Number_of_Oysters_Counted_Live_Count),  

           Density_m2 :=  

           Number_of_Oysters_Counted_Live_Count/as.numeric(QuadSize_m2)]

#Remove "25" values from total counts column, make all "PercentLiveMethod"  

#values the same, and calculate estimated live Density for ProgramID==5074 and
oysterraw <- oysterraw[RowID %in%
                        setdiff(  

                           oysterraw[, RowID],  

                           oysterraw[ProgramID ==5074 &  

                                       Number_of_Oysters_Counted_Total_Count==25, RowID]), ]  

oysterraw[ProgramID==5074, PercentLiveMethod := "Estimated percent"]  

oysterraw[ProgramID==5074, SampleDate :=  

           unique(oysterraw[ProgramID==5074 &  

                           !is.na(Number_of_Oysters_Counted_Total_Count),  

                           SampleDate])[1]]

#Some PercentLiveMethod values for ID4042 are NA
oysterraw[ProgramID==4042 | ProgramID==4016,  

           PercentLiveMethod := "Point-intercept"]

#make sure quadrat identifiers are unique
oysterraw[, QuadIdentifier_old := QuadIdentifier]
oysterraw[, QuadIdentifier := paste(UniversalReefID,
                                      LocationID, Year, Month,
                                      QuadIdentifier_old, sep="_")]

oysterraw[, MA_plotlab := paste0(ManagedAreaName, "_", HabitatClassification)]
subtidal <- c(4044, 5007, 5071, 5073)
oysterraw[, Subtidal := ifelse(ProgramID %in% subtidal, 1, 0)][, Subtidal := as.logical(Subtidal)]

#Create variables for relative year and size class category for data that
#should be included in analyses and counts of live oysters measured
for(i in unique(oysterraw$ManagedAreaName)){
  oysterraw[ManagedAreaName==i & !is.na(LiveDate), `:=`  

            (RelYear=(LiveDate-min(LiveDate))+1,  

             YearDiff=min(LiveDate)-1,  

             #adding 1 to each RelYear to avoid min(RelYear)==0,  

             #because it is used later as an index for plotting years so  

             #it needs to start from 1  

             SizeClass=fcase(ShellHeight_mm >= 25 &  

                             ShellHeight_mm < 75, "25to75mm",  

                             ShellHeight_mm >= 75, "o75mm",  

                             default=NA))]

  oysterraw[ManagedAreaName==i & !is.na(LiveDate),  

            counts := length(ShellHeight_mm), by=c("QuadIdentifier")]

```

```

}

#Remove unrealistically high shell heights from ID_5017
oysterraw <- setdiff(oysterraw, oysterraw[ProgramID==5017 & ShellHeight_mm >= 165, ])

#Create data table to save model results
oysterresults <- data.table(indicator=character(),
                           managed_area=character(),
                           habitat_class=character(),
                           size_class=character(),
                           live_date_qual=character(),
                           n_programs=integer(),
                           programs=list(),
                           filename=character(),
                           effect=character(),
                           component=character(),
                           group=character(),
                           term=character(),
                           estimate=numeric(),
                           std.error=numeric(),
                           conf.low=numeric(),
                           conf.high=numeric())

#How many years of data for each managed area/habitat class/indicator combination?
setDT(oysterraw)
oysterraw[!is.na(Density_m2), `:=` (nyrpar="Density_m2",
                                      nyyears=length(unique(Year))),
         by=MA_plotlab]
oysterraw[!is.na(PercentLive_pct), `:=` (nyrpar="PercentLive_pct",
                                           nyyears=length(unique(Year))),
         by=MA_plotlab]
oysterraw[!is.na(ShellHeight_mm), `:=` (nyrpar="ShellHeight_mm",
                                           nyyears=length(unique(Year))),
         by=MA_plotlab]
MAinclude <- distinct(oysterraw[, .(MA_plotlab, nyrpar, nyyears)])
# View(MAinclude[!is.na(nyrpar) & nyyears >= 5, ])

oysterraw[str_detect(MA_plotlab, "Pine Island Sound"),
          `:=` (MA_plotlab=ifelse(str_detect(ProgramLocationID,
                                             "Reference") |
                                             str_detect(ProgramLocationID,
                                                       "Control"),
                                             "Pine Island Sound Aquatic Preserve_Natural",
                                             "Pine Island Sound Aquatic Preserve_Restored"),
                 HabitatClassification=ifelse(str_detect(ProgramLocationID,
                                                       "Reference") |
                                             str_detect(ProgramLocationID,
                                                       "Control"),
                                                       "Natural", "Restored")))]

```

Managed Area Statistics

Gets summary statistics for each managed area. Uses piping from dplyr package to feed into subsequent steps. Sets of summary statistics are performed for Density, Shell Height, and Percent Living. The following steps are performed:

1. Group data that have the same ManagedAreaName, Year, Month, nyrrpar, LiveDate_Qualifier, SizeClass, and HabitatClassification.
 - Second summary statistics do not use the Month grouping and are only for ManagedAreaName, Year, and the other oyster parameters.
 - Third summary statistics do not use Year grouping and are only for ManagedAreaName, Month, and the other oyster parameters.
 - Fourth summary statistics are only grouped based on ManagedAreaName and the other oyster parameters
 - Determines the years that the minimum and maximum species richness occurred
2. For each group, provide the following information: Parameter Name (ParameterName), Number of Entries (N_Data), Lowest Value (Min), Largest Value (Max), Median, Mean, Standard Deviation, and a list of all Programs included in these measurements.
3. Sort the data in ascending (A to Z and 0 to 9) order based on ManagedAreaName then Year then Month
4. Write summary stats to a pipe-delimited .txt file in the output directory
 - [Oyster Output Files in SEACAR GitHub \(\[https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Oyster/output\]\(https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Oyster/output\)\)](https://github.com/FloridaSEACAR/SEACAR_Trend_Analyses/tree/main/Oyster/output)

Density

```
oysterraw$SizeClass[oysterraw$SizeClass=="25to75mm"] <- "25-75mm"
oysterraw$SizeClass[oysterraw$SizeClass=="35to75mm"] <- "35-75mm"
oysterraw$SizeClass[oysterraw$SizeClass==">75mm"] <- ">75mm"

# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrrpar=="Density_m2",] %>%
  group_by(AreaID, ManagedAreaName, Year, Month, nyrrpar,
           LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
                   Min=min(Density_m2[!is.na(Density_m2)]),
                   Max=max(Density_m2[!is.na(Density_m2)]),
                   Median=median(Density_m2[!is.na(Density_m2)]),
                   Mean=mean(Density_m2[!is.na(Density_m2)]),
                   StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
  setnames(MA_YM_Stats, c("nyrrpar", "LiveDate_Qualifier",
                         "HabitatClassification"),
           c("ParameterName", "ShellType", "HabitatType"))
  MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
  MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
  # Puts the data in order based on ManagedAreaName, Year, then Month
  MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                                 MA_YM_Stats$Year,
                                                 MA_YM_Stats$Month,
                                                 MA_YM_Stats$ShellType,
```

```

MA_YM_Stats$SizeClass,
MA_YM_Stats$HabitatType), ])

# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir, "/Density/Oyster_Dens_MA_MMYY_Stats.txt"),
sep="|")

# Save stats file to directory
ma_stats[["Density"]][["MA_YM_Stats"]] <- MA_YM_Stats
# Removes variable storing data to improve computer memory
rm(MA_YM_Stats)

# Create summary statistics for each managed area based on Year intervals
MA_Y_Stats <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
  group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
    SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
    Min=min(Density_m2[!is.na(Density_m2)]),
    Max=max(Density_m2[!is.na(Density_m2)]),
    Median=median(Density_m2[!is.na(Density_m2)]),
    Mean=mean(Density_m2[!is.na(Density_m2)]),
    StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
      collapse=', '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
      collapse=', '))
setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
  "HabitatClassification"),
  c("ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Year
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
  MA_Y_Stats$Year,
  MA_Y_Stats$ShellType,
  MA_Y_Stats$SizeClass,
  MA_Y_Stats$HabitatType), ])

# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir, "/Density/Oyster_Dens_MA_Yr_Stats.txt"),
sep="|")

# Save stats file to directory
ma_stats[["Density"]][["MA_Y_Stats"]] <- MA_Y_Stats
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
  group_by(AreaID, ManagedAreaName, Month, nyrpar,
    LiveDate_Qualifier, SizeClass,
    HabitatClassification) %>%
  dplyr::summarize(N_Data=length(Density_m2[!is.na(Density_m2)]),
    Min=min(Density_m2[!is.na(Density_m2)]),
    Max=max(Density_m2[!is.na(Density_m2)]),
    Median=median(Density_m2[!is.na(Density_m2)]),
    Mean=mean(Density_m2[!is.na(Density_m2)]),
    StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
      collapse=', '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
      collapse=', '))

```

```

    StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                   collapse=', '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                      collapse=', '))
setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                      "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Month
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                              MA_M_Stats$Month,
                                              MA_M_Stats$ShellType,
                                              MA_M_Stats$SizeClass,
                                              MA_M_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir, "/Density/Oyster_Dens_MA_Mo_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Density"]][["MA_M_Stats"]] <- MA_M_Stats
# Removes variable storing data to improve computer memory
rm(MA_M_Stats)

# Create summary overall statistics for each managed area.
MA_Ov_Stats <- oysterraw[oysterraw$nyrpar=="Density_m2",] %>%
  group_by(AreaID, ManagedAreaName, nyrpar,
            LiveDate_Qualifier, SizeClass,
            HabitatClassification) %>%
  dplyr::summarize(N_Years=length(unique(
    LiveDate[!is.na(LiveDate) & !is.na(Density_m2)])),
    SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
    EarliestLiveData=min(LiveDate[!is.na(Density_m2)]),
    LatestLiveData=max(LiveDate[!is.na(Density_m2)]),
    LastSampleDate=max(SampleDate),
    N_Data=length(Density_m2[!is.na(Density_m2)]),
    Min=min(Density_m2[!is.na(Density_m2)]),
    Max=max(Density_m2[!is.na(Density_m2)]),
    Median=median(Density_m2[!is.na(Density_m2)]),
    Mean=mean(Density_m2[!is.na(Density_m2)]),
    StandardDeviation=sd(Density_m2[!is.na(Density_m2)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                  collapse=', '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                      collapse=', '))
  setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification"),
           c("ParameterName", "ShellType", "HabitatType"))
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                              MA_Ov_Stats$ShellType,

```

```

        MA_Ov_Stats$SizeClass,
        MA_Ov_Stats$HabitatType), ])

# Replaces blank ProgramIDs with NA (missing values)
MA_Ov_Stats$ProgramIDs <- replace(MA_Ov_Stats$ProgramIDs,
                                   MA_Ov_Stats$ProgramIDs=="", NA)
MA_Ov_Stats$Programs <- replace(MA_Ov_Stats$Programs,
                                 MA_Ov_Stats$Programs=="", NA)
# Write overall statistics to file
fwrite(MA_Ov_Stats, paste0(out_dir,"/Density/Oyster_Dens_MA_Overall_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Density"]][["MA_Ov_Stats"]] <- MA_Ov_Stats
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)

```

Shell Height

```

# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
  group_by(AreaID, ManagedAreaName, Year, Month, nyrpar,
           LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
  setnames(MA_YM_Stats, c("nyrpar", "LiveDate_Qualifier",
                         "HabitatClassification"),
           c("ParameterName", "ShellType", "HabitatType"))
  MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
  MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName, Year, then Month
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                                MA_YM_Stats$Year,
                                                MA_YM_Stats$Month,
                                                MA_YM_Stats$ShellType,
                                                MA_YM_Stats$SizeClass,
                                                MA_YM_Stats$HabitatType), ])

# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir,"/Shell_Height/Oyster_SH_MA_MMYY_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Shell Height"]][["MA_YM_Stats"]] <- MA_YM_Stats
# Removes variable storing data to improve computer memory
rm(MA_YM_Stats)

```

```

# Create summary statistics for each managed area based on Year intervals
MA_Y_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
  group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
           SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
  setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification",
                        "ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Year
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
                                              MA_Y_Stats$Year,
                                              MA_Y_Stats$ShellType,
                                              MA_Y_Stats$SizeClass,
                                              MA_Y_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir,"/Shell_Height/Oyster_SH_MA_Yr_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Shell Height"]][["MA_Y_Stats"]] <- MA_Y_Stats
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
  group_by(AreaID, ManagedAreaName, Month, nyrpar,
           LiveDate_Qualifier, SizeClass,
           HabitatClassification) %>%
  dplyr::summarize(N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
  setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                        "HabitatClassification",
                        "ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"

```

```

# Puts the data in order based on ManagedAreaName then Month
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                              MA_M_Stats$Month,
                                              MA_M_Stats$ShellType,
                                              MA_M_Stats$SizeClass,
                                              MA_M_Stats$HabitatType), ])

# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir, "/Shell_Height/Oyster_SH_MA_Mo_Stats.txt"),
       sep="|")

# Save stats file to directory
ma_stats[["Shell Height"]][["MA_M_Stats"]] <- MA_M_Stats
# Removes variable storing data to improve computer memory
rm(MA_M_Stats)

# Create summary overall statistics for each managed area.
MA_Ov_Stats <- oysterraw[oysterraw$nyrpar=="ShellHeight_mm",] %>%
  group_by(AreaID, ManagedAreaName, nyrpar,
           LiveDate_Qualifier, SizeClass,
           HabitatClassification) %>%
  dplyr::summarize(N_Years=length(unique(
    LiveDate[!is.na(LiveDate) & !is.na(ShellHeight_mm)])),
    SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
    EarliestLiveDate=min(LiveDate[!is.na(ShellHeight_mm)]),
    LatestLiveDate=max(LiveDate[!is.na(ShellHeight_mm)]),
    LastSampleDate=max(SampleDate),
    N_Data=length(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    Min=min(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    Max=max(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    Median=median(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    Mean=mean(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    StandardDeviation=sd(ShellHeight_mm[!is.na(ShellHeight_mm)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                  collapse=' '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                      collapse=' '))
  setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                         "HabitatClassification"),
           c("ParameterName", "ShellType", "HabitatType"))
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                              MA_Ov_Stats$ShellType,
                                              MA_Ov_Stats$SizeClass,
                                              MA_Ov_Stats$HabitatType), ])

# Replaces blank ProgramIDs with NA (missing values)
MA_Ov_Stats$ProgramIDs <- replace(MA_Ov_Stats$ProgramIDs,
                                    MA_Ov_Stats$ProgramIDs=="", NA)
MA_Ov_Stats$Programs <- replace(MA_Ov_Stats$Programs,
                                 MA_Ov_Stats$Programs=="", NA)
# Write overall statistics to file
fwrite(MA_Ov_Stats, paste0(out_dir, "/Shell_Height/Oyster_SH_MA_Overall_Stats.txt"),
       sep="|")

```

```

sep=" | ")
# Save stats file to directory
ma_stats[["Shell Height"]][["MA_Ov_Stats"]] <- MA_Ov_Stats
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)

```

Percent Live

```

# Create summary statistics for each managed area based on Year and Month
# intervals.
MA_YM_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
  group_by(AreaID, ManagedAreaName, Year, Month, nyrpar,
           LiveDate_Qualifier, SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
                   StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
setnames(MA_YM_Stats, c("nyrpar", "LiveDate_Qualifier",
                      "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_YM_Stats$ShellType[MA_YM_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName, Year, then Month
MA_YM_Stats <- as.data.table(MA_YM_Stats[order(MA_YM_Stats$ManagedAreaName,
                                                MA_YM_Stats$Year,
                                                MA_YM_Stats$Month,
                                                MA_YM_Stats$ShellType,
                                                MA_YM_Stats$SizeClass,
                                                MA_YM_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_YM_Stats, paste0(out_dir, "/Percent_Live/Oyster_PrcLive_MA_MMYY_Stats.txt"),
       sep=" | ")
# Save stats file to directory
ma_stats[["Percent Live"]][["MA_YM_Stats"]] <- MA_YM_Stats
# Removes variable storing data to improve computer memory
rm(MA_YM_Stats)

# Create summary statistics for each managed area based on Year intervals
MA_Y_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
  group_by(AreaID, ManagedAreaName, Year, nyrpar, LiveDate_Qualifier,
           SizeClass, HabitatClassification) %>%
  dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
                   StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)])),

```

```

Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
               collapse=', '),
ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                  collapse=', '))
setnames(MA_Y_Stats, c("nyrpar", "LiveDate_Qualifier",
                      "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_Y_Stats$ShellType[MA_Y_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Year
MA_Y_Stats <- as.data.table(MA_Y_Stats[order(MA_Y_Stats$ManagedAreaName,
                                              MA_Y_Stats$Year,
                                              MA_Y_Stats$ShellType,
                                              MA_Y_Stats$SizeClass,
                                              MA_Y_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_Y_Stats, paste0(out_dir,"/Percent_Live/Oyster_PrcLive_MA_Yr_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Percent Live"]][["MA_Y_Stats"]] <- MA_Y_Stats
# Removes variable storing data to improve computer memory
rm(MA_Y_Stats)

# Create summary statistics for each managed area based on Month intervals.
MA_M_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
  group_by(AreaID, ManagedAreaName, Month, nyrpar,
           LiveDate_Qualifier, SizeClass,
           HabitatClassification) %>%
  dplyr::summarize(N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
                   StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
                   Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                                  collapse=', '),
                   ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                                     collapse=', '))
setnames(MA_M_Stats, c("nyrpar", "LiveDate_Qualifier",
                      "HabitatClassification"),
         c("ParameterName", "ShellType", "HabitatType"))
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
MA_M_Stats$ShellType[MA_M_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName then Month
MA_M_Stats <- as.data.table(MA_M_Stats[order(MA_M_Stats$ManagedAreaName,
                                              MA_M_Stats$Month,
                                              MA_M_Stats$ShellType,
                                              MA_M_Stats$SizeClass,
                                              MA_M_Stats$HabitatType), ])
# Writes summary statistics to file
fwrite(MA_M_Stats, paste0(out_dir,"/Percent_Live/Oyster_PrcLive_MA_Mo_Stats.txt"),
       sep="|")
# Save stats file to directory

```

```

ma_stats[["Percent Live"]][["MA_M_Stats"]] <- MA_M_Stats
# Removes variable storing data to improve computer memory
rm(MA_M_Stats)

# Create summary overall statistics for each managed area.
MA_Ov_Stats <- oysterraw[oysterraw$nyrpar=="PercentLive_pct",] %>%
  group_by(AreaID, ManagedAreaName, nyrpar,
           LiveDate_Qualifier, SizeClass,
           HabitatClassification) %>%
  dplyr::summarize(N_Years=length(unique(
    LiveDate[!is.na(LiveDate) & !is.na(PercentLive_pct)])),
    SufficientData=ifelse(N_Years>=5, TRUE, FALSE),
    EarliestLiveDate=min(LiveDate[!is.na(PercentLive_pct)]),
    LatestLiveDate=max(LiveDate[!is.na(PercentLive_pct)]),
    LastSampleDate=max(SampleDate),
    N_Data=length(PercentLive_pct[!is.na(PercentLive_pct)]),
    Min=min(PercentLive_pct[!is.na(PercentLive_pct)]),
    Max=max(PercentLive_pct[!is.na(PercentLive_pct)]),
    Median=median(PercentLive_pct[!is.na(PercentLive_pct)]),
    Mean=mean(PercentLive_pct[!is.na(PercentLive_pct)]),
    StandardDeviation=sd(PercentLive_pct[!is.na(PercentLive_pct)]),
    Programs=paste(sort(unique(ProgramName), decreasing=FALSE),
                  collapse=', '),
    ProgramIDs=paste(sort(unique(ProgramID), decreasing=FALSE),
                      collapse=', '))
  setnames(MA_Ov_Stats, c("nyrpar", "LiveDate_Qualifier",
                         "HabitatClassification"),
           c("ParameterName", "ShellType", "HabitatType"))
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Exact"] <- "Live Oyster Shells"
  MA_Ov_Stats$ShellType[MA_Ov_Stats$ShellType=="Estimate"] <- "Dead Oyster Shells"
# Puts the data in order based on ManagedAreaName
MA_Ov_Stats <- as.data.table(MA_Ov_Stats[order(MA_Ov_Stats$ManagedAreaName,
                                                MA_Ov_Stats$ShellType,
                                                MA_Ov_Stats$SizeClass,
                                                MA_Ov_Stats$HabitatType), ])

# Replaces blank ProgramIDs with NA (missing values)
MA_Ov_Stats$ProgramIDs <- replace(MA_Ov_Stats$ProgramIDs,
                                    MA_Ov_Stats$ProgramIDs=="", NA)
MA_Ov_Stats$Programs <- replace(MA_Ov_Stats$Programs,
                                 MA_Ov_Stats$Programs=="", NA)
# Write overall statistics to file
fwrite(MA_Ov_Stats, paste0(out_dir, "/Percent_Live/Oyster_PrcLive_MA_Overall_Stats.txt"),
       sep="|")
# Save stats file to directory
ma_stats[["Percent Live"]][["MA_Ov_Stats"]] <- MA_Ov_Stats
# Removes variable storing data to improve computer memory
rm(MA_Ov_Stats)

```

Plotting setup

```
# LiveData Threshold -----
oysterraw <- oysterraw[oysterraw$LiveData>=1960,]
for(i in unique(oysterraw$ManagedAreaName)){
  oysterraw[ManagedAreaName==i & !is.na(LiveDate), `:=`  

    (RelYear=(LiveDate-min(LiveDate))+1,  

     YearDiff=min(LiveDate)-1)]
}

# Plot theme and setup -----
plot_theme <- theme_bw() +  

  theme(panel.grid.major = element_blank(),  

        panel.grid.minor = element_blank(),  

        text=element_text(family="Arial"),  

        plot.title=element_text(hjust=0.5, size=12, color="#314963"),  

        plot.subtitle=element_text(hjust=0.5, size=10, color="#314963"),  

        legend.title=element_text(size=10),  

        legend.text = element_text(hjust = 0),  

        axis.title.x = element_text(size=10, margin = margin(t = 5, r = 0,  

                                                       b = 10, l = 0)),  

        axis.title.y = element_text(size=10, margin = margin(t = 0, r = 10,  

                                                       b = 0, l = 0)),  

        axis.text=element_text(size=10),  

        axis.text.x=element_text(angle = -45, hjust = 0))

plot_jitter <- position_jitter(width = 0.1, height = 0.1, seed=42)
```

Oyster Shell Height Analysis

Subsets data for that which is related to shell height. The data is further subset for each managed area and shell size class. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. `shell_height_models_par` function also creates figures and saves data.

```
# Find out which MAs should receive models for SH
sh_stats <- ma_stats[["Shell Height"]][["MA_Ov_Stats"]] %>%
  filter(!is.na(SizeClass)) %>%
  select(ManagedAreaName, ParameterName, ShellType, SizeClass, HabitatType) %>%
  as.data.table()

task_list <- sh_stats[, .(HabitatType = unique(HabitatType)), by = ManagedAreaName]
task_list <- task_list[!(ManagedAreaName %in% c("Apalachicola Bay Aquatic Preserve",
                                              "Apalachicola National Estuarine Research Reserve")) & H]

# Add MA abbreviation
task_list[, ma_abrev := MA_All[.SD, on = "ManagedAreaName", Abbreviation]]
task_list <- as.data.frame(task_list)

# Function to subset data and run models where possible
shell_height_models_par <- function(ma, ma_abrev, habitat_type, oysterraw){
  library(future)
  library(future.apply)
  library(tidyverse)
  library(data.table)
```

```

library(ggplot2)
library(brms)
library(Rmisc)
library(rstan)
library(rstantools)
library(stringr)
library(sf)
library(tictoc)
library(rstudioapi)
library(ggpubr)

# Combined MA name with habitat type
ma_plotlabel <- paste0(ma, "_", str_to_title(habitat_type))
# At least 5 years of data are required in order to run model analyses
# Function checks N years of data, returns T or F
suff_years <- function(data){length(unique(data$Year))>=5}

if(ma_abrev %in% c("ABAP", "ANERR")){
  #Exclude the five samples that don't have counts less than the "NumberMeasured"
  #value for the corresponding program (see variable exploration graphs in the
  #25to75mm section for the rationale and graphs for this step.)
  numValves <- unique(oysterraw[, c("ProgramID", "RelYear", "counts",
                                     "QuadIdentifier", "Subtidal", "QuadSize_m2",
                                     "LiveDate_Qualifier", "NumberMeasured_n")])

  exclude_samps <- subset(numValves, numValves$NumberMeasured_n=="20" &
                           numValves$counts > 19)$QuadIdentifier

  sho25 <- oysterraw[!is.na(ShellHeight_mm) & ShellHeight_mm >= 25 &
                     MA_plotlab==ma_plotlabel & QuadIdentifier %in% setdiff(
                       oysterraw[!is.na(ShellHeight_mm) & ManagedAreaName==ma, QuadIdentifier], exclude_samps)]
} else {
  sho25 <- oysterraw[!is.na(ShellHeight_mm) & ShellHeight_mm >= 25 & MA_plotlab==ma_plotlabel, ]
}
# Save shell height data > 25mm
saveRDS(sho25, paste0("output/model_results/data/", ma_abrev, "_sho25_", Sys.Date(), "_", habitat_type))

# Subset and save for shell height data >25 & <75
sh25to75 <- sho25[ShellHeight_mm < 75, ]
saveRDS(sh25to75, paste0("output/model_results/data/", ma_abrev, "_sh25to75_", Sys.Date(), "_", habitat_type))

# Subset for model data (where LiveDate_Qualifier is "Exact" NOT "Estimate")
sh25to75_mod_data <- subset(sh25to75, sh25to75$LiveDate_Qualifier!="Estimate")

if(suff_years(sh25to75_mod_data) & !QAQCPlots){
  cat("---- Sufficient years of data for SH 25mm to 75mm. Running model. \n")
  # Set formula to account for multiple quadsizes
  if(length(unique(sh25to75_mod_data$QuadSize_m2))>1){
    f <- brms::brmsformula(ShellHeight_mm | trunc(lb=25, ub=75) ~ RelYear + QuadSize_m2 + (1 | UniversalReefID))
  } else {
    f <- brms::brmsformula(ShellHeight_mm | trunc(lb=25, ub=75) ~ RelYear + (1 | UniversalReefID))
  }
  # Failed convergence in PISAP due to low number of UniversalReefID, try simpler model
}

```

```

if(ma_abrev=="PISAP"){
  f <- brms::brmsformula(ShellHeight_mm | trunc(lb=25, ub=75) ~ RelYear)
}
sh25to75_glmm <- brm(
  # formula=ShellHeight_mm / trunc(lb=25, ub=75) ~ RelYear+QuadSize_m2+(1 | UniversalReefID),
  formula = f,
  data=sh25to75_mod_data,
  family=gaussian, cores=ncores,
  control=list(adapt_delta=0.995, max_treedepth=20),
  iter=iter, warmup=warmup, chains=nchains, thin=3, seed=5699,
  backend="rstan",
  file=paste0("output/model_results/GLMMs/", ma_abrev, "_sh25to75_glmm_", habitat_type, ".rds")
)
models1 <- list(sh25to75_glmm)
} else {models1 <- NULL}

# Set variables for use within plots
data1 <- sh25to75

# Subset and save for shell height data >=75
sho75 <- sho25[ShellHeight_mm >= 75, ]
saveRDS(sho75, paste0("output/model_results/data/", ma_abrev, "_sho75_", Sys.Date(), "_", habitat_type))

# Subset for model data (where LiveDate_Qualifier is "Exact" NOT "Estimate")
sho75_mod_data <- subset(sho75, sho75$LiveDate_Qualifier!="Estimate")

if(suff_years(sho75_mod_data) & !QAQCPlots){
  cat("---- Sufficient years of data for SH over 75mm. Running model. \n")
  # Set formula to account for multiple quadsizes
  if(length(unique(sho75_mod_data$QuadSize_m2))>1){
    f <- brms::brmsformula(ShellHeight_mm | trunc(lb=75, ub=250) ~ RelYear + QuadSize_m2 + (1 | UniversalReefID))
  } else {
    f <- brms::brmsformula(ShellHeight_mm | trunc(lb=75, ub=250) ~ RelYear + (1 | UniversalReefID))
  }
  # Failed convergence in PISAP due to low number of UniversalReefID, try simpler model
  if(ma_abrev=="PISAP"){
    f <- brms::brmsformula(ShellHeight_mm | trunc(lb=75, ub=250) ~ RelYear)
  }
  sho75_glmm <- brm(
    formula = f,
    data=sho75_mod_data,
    family=gaussian, cores=ncores,
    control= list(adapt_delta=0.995, max_treedepth=20),
    iter=iter, warmup=warmup, chains=nchains, thin=3, seed=3639,
    backend="rstan",
    file=paste0("output/model_results/GLMMs/", ma_abrev, "_sho75_glmm_", habitat_type, ".rds")
)
  models2 <- list(sho75_glmm)
} else {models2 <- NULL}
# Set variables for use within plots
data2 <- sho75

##### modresultssh_par function #####

```

```

datafile1 <- data1
datafile2 <- data2
indicator <- "Size class"
meplotzoom <- FALSE

oysterresults_temp <- data.frame()
datafile1$SizeClass[datafile1$SizeClass=="25to75mm" &
                    datafile1$MA_plotlab==
                    "St. Martins Marsh Aquatic Preserve_Natural"] <-
                    "35-75mm"
sizeclass1 <- unique(datafile1$SizeClass)
for(m in seq_along(models1)){
  modelobj <- models1[[m]]
  oyres_i <- setDT(broom.mixed::tidy(modelobj))
  #tidy() does not like that parameter values have underscores
  #for some reason, so the resulting table is incomplete

  if(nrow(oyres_i[effect=="fixed", ]) - nrow(summary(modelobj)$fixed) == -1){
    missingrow <- data.table(effect="fixed",
                               component="cond",
                               #not sure what "cond" means in the tidy summary.
                               group=NA,
                               term=rownames(summary(modelobj)$fixed)[2],
                               estimate=summary(modelobj)$fixed$Estimate[2],
                               std.error=summary(modelobj)$fixed$Est.Error[2],
                               conf.low=summary(modelobj)$fixed$`1-95% CI`[2],
                               conf.high=summary(modelobj)$fixed$`u-95% CI`[2])
    oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
  }

  setDT(oyres_i)
  oyres_i[, `:=` (indicator=indicator,
                  managed_area=unique(datafile1$ManagedAreaName),
                  habitat_class=unique(datafile1$HabitatClassification),
                  size_class=sizeclass1,
                  live_date_qual=ifelse(
                    str_detect(
                      modelobj$file, "_hist"), "Estimate",
                    "Exact"),
                  n_programs=if(class(
                    try(datafile1$LiveDate_Qualifier)) != "try-error"){
                    length(unique(
                      datafile1[LiveDate_Qualifier==
                        ifelse(str_detect(
                          modelobj$file, "_hist"),
                          "Estimate", "Exact"),
                      ProgramID]))})
                } else{length(unique(datafile1[, ProgramID]))},
                programs=if(class(try(
                  datafile1$LiveDate_Qualifier)) != "try-error"){
                  list(unique(
                    datafile1[LiveDate_Qualifier==
                      ifelse(

```

```

        str_detect(
            modelobj$file,
            "_hist"),
        "Estimate",
        "Exact"),
        ProgramID)))
} else{list(unique(datafile1[, ProgramID]))},
filename=modelobj$file)]}

oysterresults_temp <- rbind(oysterresults_temp, oyres_i)
}

datafile2$SizeClass[datafile2$SizeClass=="25to75mm" &
                    datafile2$MA_plotlab==
                    "St. Martins Marsh Aquatic Preserve_Natural"] <- "35-75mm"
sizeclass2 <- unique(datafile2$SizeClass)

for(m in seq_along(models2)){
  modelobj <- models2[[m]]
  oyres_i <- setDT(broom.mixed::tidy(modelobj))
  #tidy() does not like that parameter values have underscores for
  #some reason, so the resulting table is incomplete

  if(nrow(oyres_i[effect=="fixed", ]) - nrow(summary(modelobj)$fixed) == -1){
    missingrow <- data.table(effect="fixed",
                               component="cond",
                               #not sure what "cond" means in the tidy summary.
                               group=NA,
                               term=rownames(summary(modelobj)$fixed)[2],
                               estimate=summary(modelobj)$fixed$Estimate[2],
                               std.error=summary(modelobj)$fixed$Est.Error[2],
                               conf.low=summary(modelobj)$fixed`1-95% CI`[2],
                               conf.high=summary(modelobj)$fixed`u-95% CI`[2])
    oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
  }

  oyres_i <- oyres_i %>%
    mutate(
      indicator = indicator,
      managed_area = unique(datafile2$ManagedAreaName),
      habitat_class = unique(datafile2$HabitatClassification),
      size_class = sizeclass2,
      live_date_qual = if_else(
        str_detect(modelobj$file, "_hist"), "Estimate", "Exact"
      ),
      n_programs = if (class(try(datafile2$LiveDate_Qualifier)) != "try-error") {
        datafile2 %>%
          filter(LiveDate_Qualifier == if_else(str_detect(modelobj$file, "_hist"), "Estimate", "Exact"))
          pull(ProgramID) %>%
          unique() %>%
          length()
      } else {
        datafile2 %>%

```

```

    pull(ProgramID) %>%
    unique() %>%
    length()
},
programs = if (class(try(datafile2$LiveDate_Qualifier)) != "try-error") {
  list(datafile2 %>%
        filter(LiveDate_Qualifier == if_else(str_detect(modelobj$file, "_hist"), "Estimate", "ProgramID"))
        %>%
        unique())
} else {
  list(datafile2 %>% pull(ProgramID) %>% unique())
},
filename = modelobj$file
)
oysterresults_temp <- rbind(oysterresults_temp, oyres_i)
}

ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
                  str_detect(indicator, "ensity") ~ "Den",
                  str_detect(indicator, "^S|^s") ~ "SH")

if(nrow(data1)>0){
  sizeclass1 <- unique(data1$SizeClass)
} else {
  sizeclass1 <- ""
}
if(nrow(data2)>0){
  sizeclass2 <- unique(data2$SizeClass)
} else {
  sizeclass2 <- ""
}

# Set size labels
if(sizeclass1 != ""){
  size1 <- case_when(
    str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25to75",
    str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35to75",
    str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "o75",
    TRUE ~ "raw")
  sizelab1 <- case_when(
    str_detect(sizeclass1, "25") & str_detect(sizeclass1, "75") ~ "25-75mm",
    str_detect(sizeclass1, "35") & str_detect(sizeclass1, "75") ~ "35-75mm",
    str_detect(sizeclass1, "25")==FALSE & str_detect(sizeclass1, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}
if(sizeclass2 != ""){
  size2 <- case_when(
    str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25to75",
    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35to75",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "o75",
    TRUE ~ "raw")
  sizelab2 <- case_when(
    str_detect(sizeclass2, "25") & str_detect(sizeclass2, "75") ~ "25-75mm",
    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35-75mm",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}

```

```

    str_detect(sizeclass2, "35") & str_detect(sizeclass2, "75") ~ "35-75mm",
    str_detect(sizeclass2, "25")==FALSE & str_detect(sizeclass2, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
} else {
  size2 <- "o75"
  sizelab2 <- "\u2265 75mm"
}

#Marginal effects plot including random effects
## Hist plot settings
if(nrow(data2)>0){
  y_max <- round(max(data2[!is.na(ShellHeight_mm), ShellHeight_mm]), -0)+1
} else {
  y_max <- round(max(data1[!is.na(ShellHeight_mm), ShellHeight_mm]), -0)+1
}
y_breaks <- seq(25, 300, 50)
y_labs <- seq(25, 300, 50)
y_minor <- seq(0, 300, 25)
ylim_upper <- ceiling(y_max/25)*25

yrdiff1 <- unique(data1$YearDiff)
yrdiff2 <- unique(data2$YearDiff)

# function to set year breaks, type == "hist" or "live"
set_breaks <- function(type, data1, data2){
  ldq <- ifelse(type=="hist", "Estimate", "Exact")

  maxyr <- max(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
                 data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  minyr <- min(data1[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate],
                 data2[!is.na(LiveDate) & LiveDate_Qualifier==ldq, LiveDate])
  nyrs <- maxyr - minyr + 1

  current_year <- as.integer(format(Sys.Date(), "%Y"))

  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 40 years of data
    brk <- 10
  } else if(nyrs>=20){
    # Set breaks to every 5 years if between 40 and 20 years of data
    brk <- 5
  } else if(nyrs>=12){
    # Set breaks to every 3 years if between 20 and 12 years of data
    brk <- 3
  } else if(nyrs>=8){
    # Set breaks to every 2 years if between 12 and 8 years of data
    brk <- 2
  } else if(nyrs>=5){
    # Set breaks to every year if between 8 and 5 years of data
    brk <- 1
  } else {
    # Ensure 5 years are included on axis
  }
}

```

```

total_ticks <- 5
extra_years <- total_ticks - nyears
# Always add 1 year before the first year
years_before <- min(1, extra_years)
years_after <- extra_years - years_before
# Adjust min and max year, without going beyond current year
minyr <- minyr - years_before
maxyr <- min(maxyr + years_after, current_year)
# Re-check if we have enough years (in case maxyr hit current year)
minyr <- max(minyr, maxyr - (total_ticks - 1))
brk <- 1
}
return(list("seq" = seq(minyr,maxyr,brk),"maxyr" = maxyr, "minyr" = minyr))
}

## Check data for Exact and Estimate
n_hist1 <- nrow(data1[data1$LiveDate_Qualifier=="Estimate" &
                      !is.na(data1$ShellHeight_mm),])
n_live1 <- nrow(data1[data1$LiveDate_Qualifier=="Exact" &
                      !is.na(data1$ShellHeight_mm),])
n_hist2 <- nrow(data2[data2$LiveDate_Qualifier=="Estimate" &
                      !is.na(data2$ShellHeight_mm),])
n_live2 <- nrow(data2[data2$LiveDate_Qualifier=="Exact" &
                      !is.na(data2$ShellHeight_mm),])

# Plot variable to record which plots to show (dead, live, or both)
available_plots <- c()
# If "Estimate" data exists, set y-axis (years)
if(n_hist1>0 | n_hist2>0){
  yrlist_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["seq"]]
  maxyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["maxyr"]]
  minyr_hist <- set_breaks(type = "hist", data1 = data1, data2 = data2)[["minyr"]]
  available_plots <- c(available_plots, "dead")
}
# If "Exact" data exists, set y-axis (years)
if(n_live1>0 | n_live2>0){
  yrlist_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["seq"]]
  maxyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["maxyr"]]
  minyr_live <- set_breaks(type = "live", data1 = data1, data2 = data2)[["minyr"]]
  available_plots <- c(available_plots, "live")
}

set.seed(987)
if(!is.null(models1) & !QAQCPlots){
  liveplot_1 <- plot(conditional_effects(models1[[1]]), re_formula=NULL, plot=FALSE)
}

if(!is.null(models2) & !QAQCPlots){
  liveplot_2 <- plot(conditional_effects(models2[[1]]), re_formula=NULL, plot=FALSE)
}

# Set boolean values for whether liveplot1&2 are available
liveplot1_avail <- class(try(liveplot_1, silent=TRUE)) != "try-error"

```

```

liveplot2_avail <- class(try(liveplot_2, silent=TRUE)) != "try-error"

# Set ribbon transparency value
a_ribb <- 0.2
# Set size and shapes for plots
p_shape <- c("size2"=24, "size1"=21)
sizelab <- c("size2"=sizelab2, "size1"=sizelab1)

col1 <- NA
col2 <- NA

# "transparent" allows for dummy values to be plotted. Ensures proper legend display
if(liveplot1_avail){
  col1 <- c(size1="#00374f")
} else{
  col1 <- c(size1="transparent")
}

if(liveplot2_avail){
  col2 <- c(size2="#0094b0")
} else{
  col2 <- c(size2="transparent")
}

p_color <- c(col2, col1)

# Initial plots to set legends
plot_leg <- ggplot() +
  {if(liveplot1_avail){
    list(geom_ribbon(data=liveplot_1$RelYear$data,
                     aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                          ymin=lower__, ymax=upper__,
                          fill="size1"),
                     alpha=a_ribb,
                     show.legend = TRUE),
        geom_line(data=liveplot_1$RelYear$data,
                  aes(x=RelYear+yrdiff1, y=estimate__,
                      color="size1"),
                  lwd=0.75,
                  show.legend = TRUE),
    # Dummy values
    geom_ribbon(data=liveplot_1$RelYear$data,
                aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
                    ymin=lower__, ymax=upper__,
                    fill="size2"),
                alpha=a_ribb,
                show.legend = TRUE),
    geom_line(data=liveplot_1$RelYear$data,
              aes(x=RelYear+yrdiff1, y=estimate__,
                  color="size2"),
              lwd=0.75,
              show.legend = TRUE))
  } + 
}

```

```

{if(liveplot2_avail){
  list(geom_ribbon(data=liveplot_2$RelYear$data,
                  aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                       ymin=lower__, ymax=upper__,
                       fill="size2"),
                  alpha=a_rabb,
                  show.legend = TRUE),
       geom_line(data=liveplot_2$RelYear$data,
                  aes(x=RelYear+yrdiff2, y=estimate__,
                       color="size2"),
                  lwd=0.75,
                  show.legend = TRUE),
       # Dummy values
       geom_ribbon(data=liveplot_2$RelYear$data,
                  aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
                       ymin=lower__, ymax=upper__,
                       fill="size1"),
                  alpha=a_rabb,
                  show.legend = TRUE),
       geom_line(data=liveplot_2$RelYear$data,
                  aes(x=RelYear+yrdiff2, y=estimate__,
                       color="size1"),
                  lwd=0.75,
                  show.legend = TRUE))
  )} +
  # Dummy points
  geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
             position=plot_jitter, size=2, color="transparent", fill = "transparent",
             alpha=0.8, show.legend = TRUE) +
  geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
             position=plot_jitter, size=2, color="#333333", fill = "#cccccc",
             alpha=0.8, show.legend = TRUE) +
  # Dummy points
  geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
             position=plot_jitter, size=2, color="transparent", fill = "transparent",
             alpha=0.8, show.legend = TRUE) +
  geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate), ],
             aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
             position=plot_jitter, size=2, color="#333333", fill = "#cccccc",
             alpha=0.8, show.legend = TRUE) +
  plot_theme +
  theme(legend.position="right") +
  scale_shape_manual(name="Size class",
                     breaks = c("size2", "size1"),
                     values=p_shape,
                     labels=sizelab) +
  scale_color_manual(name="Size class",
                     breaks = c("size2", "size1"),
                     values=p_color,
                     labels=sizelab,

```

```

aesthetics = c("color", "fill"))

leg <- get_legend(plot_leg)
rm(plot_leg)

# Dead oyster shell plot
if("dead" %in% available_plots){
  plot1 <- ggplot() +
    geom_hline(yintercept=75, linewidth=1, color="grey") +
    {if(n_hist1>0){
      geom_point(data=data1[!is.na(RelYear) &
        !is.na(LiveDate) &
        LiveDate_Qualifier=="Estimate", ],
      aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
      position=plot_jitter, size=2, color="#333333", fill="#cccccc",
      alpha=0.8, inherit.aes=FALSE)
    }} +
    {if(n_hist2>0){
      geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
        LiveDate_Qualifier=="Estimate", ],
      aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
      position=plot_jitter, size=2, color="#333333", fill="#cccccc",
      alpha=0.8, inherit.aes=FALSE)
    }} +
    scale_x_continuous(limits=c(minyr_hist-0.25, maxyr_hist+0.25),
                        breaks=yrlist_hist) +
    scale_y_continuous(breaks=y_breaks,
                       labels=y_labs, minor_breaks=y_minor) +
    plot_theme +
    theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
          legend.position="none") +
    labs(subtitle="Dead Oyster Shells",
         x="Estimated year",
         y="Shell height (mm)") +
    scale_shape_manual(name="Shell heights",
                      values=c("size1"=21, "size2"=24),
                      labels=c(sizelab1, sizelab2)) +
    scale_color_manual(name="Shell heights",
                      values=c("size1"="#00374f", "size2"="#0094b0"),
                      labels=c(sizelab1, sizelab2)) +
    scale_fill_manual(name="Shell heights",
                      values=c("size1"="#00374f", "size2"="#0094b0"),
                      labels=c(sizelab1, sizelab2)) +
    coord_cartesian(ylim=c(25, ylim_upper))
}

# Live oyster shell plot
if("live" %in% available_plots){
  plot2 <- ggplot() +
    geom_hline(yintercept=75, size=1, color="grey") +
    {if(n_live1>0){
      geom_point(data=data1[!is.na(RelYear) & !is.na(LiveDate) &
        LiveDate_Qualifier=="Exact", ],

```

```

    aes(x=LiveDate, y=ShellHeight_mm, shape="size1"),
    position=plot_jitter, size=2, color="#333333", fill="#cccccc",
    alpha=0.8, inherit.aes=FALSE)
}} +
{if(n_live2>0){
  geom_point(data=data2[!is.na(RelYear) & !is.na(LiveDate) &
    LiveDate_Qualifier=="Exact", ],
    aes(x=LiveDate, y=ShellHeight_mm, shape="size2"),
    position=plot_jitter, size=2, color="#333333", fill="#cccccc",
    alpha=0.8, inherit.aes=FALSE)
}} +
{if(liveplot1_avail){
  list(geom_ribbon(data=liveplot_1$RelYear$data,
    aes(x=RelYear+yrdiff1, y=ShellHeight_mm,
        ymin=lower__, ymax=upper__, fill="size1"),
    alpha=a_ribb),
    geom_line(data=liveplot_1$RelYear$data,
      aes(x=RelYear+yrdiff1, y=estimate__, color="size1"),
      lwd=0.75))
}} +
{if(liveplot2_avail){
  list(geom_ribbon(data=liveplot_2$RelYear$data,
    aes(x=RelYear+yrdiff2, y=ShellHeight_mm,
        ymin=lower__, ymax=upper__, fill="size2"),
    alpha=a_ribb),
    geom_line(data=liveplot_2$RelYear$data,
      aes(x=RelYear+yrdiff2, y=estimate__, color="size2"),
      lwd=0.75))
}} +
scale_x_continuous(limits=c(minyr_live-0.25, maxyr_live+0.25),
  breaks=yrlist_live) +
scale_y_continuous(breaks=y_breaks,
  labels=y_labs, minor_breaks=y_minor) +
plot_theme +
theme(plot.subtitle=element_text(hjust=0, size=10, color="#314963"),
  legend.position="none") +
labs(subtitle="Live Oysters",
  x="Year",
  y="Shell height (mm)") +
scale_shape_manual(name="Shell heights",
  values=c("size1"=21, "size2"=24),
  labels=c(sizelab1, sizelab2)) +
scale_color_manual(name="Shell heights",
  values=c("size1"="#00374f", "size2"="#0094b0"),
  labels=c(sizelab1, sizelab2)) +
scale_fill_manual(name="Shell heights",
  values=c("size1"="#00374f", "size2"="#0094b0"),
  labels=c(sizelab1, sizelab2)) +
coord_cartesian(ylim=c(25, ylim_upper))
}

# Set plot title
plot_title <- ggplot() +

```

```

    labs(title=paste0("Oyster Size Class (", habitat_type, ")"), subtitle=ma) +
    plot_theme +
    theme(plot.subtitle=element_text(hjust=0.5, size=10, color="#314963"),
          panel.border=element_blank(),
          panel.grid.major=element_blank(),
          panel.grid.minor=element_blank(), axis.line=element_blank())

  if("live" %in% available_plots & "dead" %in% available_plots){
    # Remove y-axis labels, ticks, title before combining both plots
    plot2 <- plot2 +
      theme(legend.position="none",
            axis.text.y=element_blank(), #remove y-axis labels
            axis.ticks.y=element_blank(), #remove y-axis ticks
            axis.title.y=element_blank()) #removes y-axis title
    # Combine live and dead plots + legend
    plot_comb <- ggarrange(plot1, plot2, leg, nrow=1,
                           widths=c(0.46, 0.39, 0.15))
  } else if("live" %in% available_plots & !"dead" %in% available_plots){
    # Combine live plots with legend
    plot_comb <- ggarrange(plot2, leg, nrow=1,
                           widths=c(0.85, 0.15))
  } else if("dead" %in% available_plots & !"live" %in% available_plots){
    # Combine dead plots with legend
    plot_comb <- ggarrange(plot1, leg, nrow=1,
                           widths=c(0.85, 0.15))
  }
  plot_comb <- ggarrange(plot_title, plot_comb, ncol=1,
                         heights=c(0.125, 0.875))

  # Specify save location (QAQC Plots saved elsewhere)
  if(QAQCPlots){
    file_name <- paste0("output/QAQC/Oyster_SH_GLMM_", ma_abrev, "_", habitat_type, ".png")
  } else {
    file_name <- paste0("output/Shell_Height/Figures/Oyster_SH_GLMM_", ma_abrev, "_", habitat_type, ".png")
  }

  ggsave(file_name,
         plot_comb,
         width=8,
         height=4,
         units="in",
         dpi=200,
         bg="white")

  return(oysterresults_temp)

  cat("---- Shell Height plot created for", ma, "--", habitat_type, "\n")
}

split_tasks <- split(task_list, ceiling(seq_along(1:nrow(task_list)) / 4))
oyster_sh <- oysterraw %>% filter(!is.na(ShellHeight_mm))
# Subset and save temp .rds objects for each MA (breaks up oysterraw)

```

```

for(ma in unique(oyster_sh$ManagedAreaName)){
  saveRDS(oyster_sh[oyster_sh$ManagedAreaName == ma, ],
          file = paste0("output/tmp/oystersh_", make.names(ma), ".rds"))
}

results_all <- list()
for(b in seq_along(split_tasks)){
  batch <- split_tasks[[b]]

  results_list <- future_lapply(seq_len(nrow(batch)), function(i) {
    task <- batch[i, ]
    oysterraw_path <- paste0("output/tmp/oystersh_", make.names(task$ManagedAreaName), ".rds")
    oysterraw_sub <- readRDS(oysterraw_path)
    cat("-- Analyzing ", task$ManagedAreaName, "\n")
    shell_height_models_par(
      ma = task$ManagedAreaName,
      ma_abrev = task$ma_abrev,
      habitat_type = task$HabitatType,
      oysterraw = oysterraw_sub
    )
  }, future.seed = TRUE)
  results_all[[b]] <- data.table::rbindlist(results_list, fill = TRUE)
  gc()
}

oysterresults_sh <- data.table::rbindlist(results_all, fill = TRUE)
fwrite(oysterresults_sh, "output/oyresults/oysterresults_sh.csv")

```

Density Analysis

Subsets data for that which is related to density. The data is further subset for each managed area. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. `density_models_par` function also creates figures and saves data.

```

oysterraw$YearDiff <- oysterraw$LiveDate-oysterraw$RelYear
# #Make a collapsed version of the oysterraw table for density
oysterraw_den <- oysterraw[, c("ProgramID", "ProgramName", "LocationID",
                               "ProgramLocationID", "QuadIdentifier",
                               "ReefIdentifier", "LiveDate",
                               "LiveDate_Qualifier", "SampleDate", "Year",
                               "Month", "ManagedAreaName", "Region",
                               "SurveyMethod", "HabitatClassification",
                               "QuadSize_m2", "MADup", "Density_m2",
                               "Number_of_Oysters_Counted_Total_Count",
                               "Number_of_Oysters_Counted_Live_Count",
                               "Number_of_Oysters_Counted_Dead_Count",
                               "ObsIndex", "UniversalReefID",
                               "MA_plotlab", "Subtidal", "RelYear", "YearDiff")]
oysterraw_den[!is.na(Density_m2), DensIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Total_Count), NTotalIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Live_Count), NLiveIndex := ObsIndex]
oysterraw_den[!is.na(Number_of_Oysters_Counted_Dead_Count), NDeadIndex := ObsIndex]
oysterraw_den[, ObsIndex := NULL]

```

```

oysterraw_den <- unique(oysterraw_den)
oysterraw_den <- oysterraw_den %>%
  dplyr::group_by(ProgramID, ProgramName, LocationID, ProgramLocationID,
                 QuadIdentifier, ReefIdentifier, LiveDate,
                 LiveDate_Qualifier, SampleDate, Year, Month,
                 ManagedAreaName, Region, SurveyMethod,
                 HabitatClassification, QuadSize_m2, MADup, UniversalReefID,
                 MA_plotlab, Subtidal) %>%
  tidyr::fill(Density_m2, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              DensIndex, NTotIndex, NLiveIndex, NDeadIndex) %>%
  tidyr::fill(Density_m2, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              DensIndex, NTotIndex, NLiveIndex, NDeadIndex,
              .direction='up') %>%
dplyr::distinct()

oysterraw_den <- subset(oysterraw_den, !is.na(oysterraw_den$Density_m2) |
  !is.na(oysterraw_den$Number_of_Oysters_Counted_Total_Count) |
  !is.na(oysterraw_den$Number_of_Oysters_Counted_Live_Count) |
  !is.na(oysterraw_den$Number_of_Oysters_Counted_Dead_Count) |
  !is.na(oysterraw_den$DensIndex) |
  !is.na(oysterraw_den$NTotIndex) |
  !is.na(oysterraw_den$NLiveIndex) |
  !is.na(oysterraw_den$NDeadIndex))

setDT(oysterraw_den)

#Remove NAs in Density_m2 column
oysterraw_den <- subset(oysterraw_den, !is.na(oysterraw_den$Density_m2))

#Summarize density data by managed area
den_all_sum <- summarySE(oysterraw_den, measurevar='Density_m2',
                           groupvars=c('ManagedAreaName', 'Year'))

# Find out which MAs should receive models for Density
den_stats <- ma_stats[[ "Density"]][[ "MA_Ov_Stats"]] %>%
  select(ManagedAreaName, ParameterName, HabitatType) %>%
  as.data.table()

task_list <- den_stats[, .(HabitatType = unique(HabitatType)), by = ManagedAreaName]
# Add MA abbreviation
task_list[, ma_abrev := MA_All[.SD, on = "ManagedAreaName", Abbreviation]]
task_list <- as.data.frame(task_list)

# Density modelling function
density_models_par <- function(ma, ma_abrev, habitat_type, oysterraw_den){
  library(future)
  library(future.apply)
  library(tidyverse)
  library(data.table)
  library(ggplot2)

```

```

library(brms)
library(Rmisc)
library(rstan)
library(rstantools)
library(stringr)
library(sf)
library(tictoc)
library(rstudioapi)
library(ggpubr)
cat("----", paste0("Habitat type: ", habitat_type, "\n"))
# Combined MA name with habitat type
ma_plotlabel <- paste0(ma, "_", str_to_title(habitat_type))
# At least 5 years of data are required in order to run model analyses
# Function checks N years of data, returns T or F
suff_years <- function(data){length(unique(data$Year))>=5}
# Create subset for each MA
ma_subset <- subset(oysterraw_den, oysterraw_den$MA_plotlab==ma_plotlabel)
# Ensure Density_m2 is numeric & rounded
ma_subset[, Density_m2 := as.integer(round(Density_m2))]
# Save data used in model
saveRDS(ma_subset, paste0("output/model_results/data/", ma_abrev, "_density_", Sys.Date(), "_", habitat_type))

if(suff_years(ma_subset) & !QAQCPlots){
  cat("---- Sufficient years of data for Density. Running model. \n")
  # Determine model family
  # When to use negbinomial or zero-inflated-negbinomial
  # If zeroes make up >30% of dataset, use zero-inflated
  if(mean(ma_subset$Density_m2 == 0)>0.3){
    fam <- zero_inflated_negbinomial()
  } else {
    fam <- negbinomial()
  }
  # Determine formula (if to add subtidal or quadsize factor)
  # If more than 1 values for either Subtidal or QuadSize_m2, include as fixed effect
  num_subtidals <- length(unique(ma_subset$Subtidal))
  num_quads <- length(unique(ma_subset$QuadSize_m2))
  if(num_subtidals>1 & num_quads>1){
    f <- brms::brmsformula(Density_m2 ~ RelYear + Subtidal + QuadSize_m2 + (1 + RelYear | UniversalReefID))
  } else if(num_subtidals>1 & !num_quads>1) {
    f <- brms::brmsformula(Density_m2 ~ RelYear + Subtidal + (1 + RelYear | UniversalReefID))
  } else if(!num_subtidals>1 & num_quads>1){
    f <- brms::brmsformula(Density_m2 ~ RelYear + QuadSize_m2 + (1 + RelYear | UniversalReefID))
  } else {
    f <- brms::brmsformula(Density_m2 ~ RelYear + (1 + RelYear | UniversalReefID))
  }
  if(ma_abrev=="PISAP"){
    f <- Density_m2 ~ RelYear + (0 + RelYear | UniversalReefID)
  }
  if(ma_abrev=="ANERR" & habitat_type=="Natural"){
    f <- Density_m2 ~ RelYear + Subtidal + QuadSize_m2 + (0 + RelYear | UniversalReefID)
  }

  cat("----- Using Formula: ", paste(f[1]), "\n")
}

```

```

cat("----- Using Family: ", paste(fam[1]), "\n")

den_glmm <- brm(formula=f, data=ma_subset,
                  family=fam, cores=ncores,
                  control= list(adapt_delta=0.995, max_treedepth=20),
                  iter=iter, warmup=warmup, chains=nchains,
                  init=0, thin=3, seed=sample.int(.Machine$integer.max, 1),
                  backend="rstan",
                  file=paste0("output/model_results/GLMMs/", ma_abrev, "_den_glmm9_",
                  habitat_type, "))

} else {
  den_glmm <- NA
}

# Create model results tables and save diagnostic plots and marginal effects plots
datafile <- ma_subset
models <- list(den_glmm)
indicator <- "Density"
meplotzoom <- FALSE
oysterresults_temp <- data.frame()
if(class(den_glmm)=="brmsfit"){
  for(m in seq_along(models)){
    modelobj <- models[[m]]
    sizeclass <- ifelse(str_detect(modelobj$file, "25to75|seed"),
                         "25-75mm",
                         ifelse(str_detect(modelobj$file, "35to75|seed"),
                               "35-75mm",
                               ifelse(str_detect(modelobj$file,
                                                 "o75|market"),
                                     "o75|market",
                                     ">75mm", "NA")))
    oyres_i <- setDT(broom.mixed::tidy(modelobj))
    #tidy() does not like that parameter values have underscores for
    #some reason, so the resulting table is incomplete

    if(nrow(oyres_i[effect=="fixed", ]) - nrow(summary(modelobj)$fixed) == 1){
      missingrow <- data.table(effect="fixed",
                                 component="cond",
                                 #not sure what "cond" means in the tidy summary.
                                 group=NA,
                                 term=rownames(summary(modelobj)$fixed)[2],
                                 estimate=summary(modelobj)$fixed$Estimate[2],
                                 std.error=summary(modelobj)$fixed$Est.Error[2],
                                 conf.low=summary(modelobj)$fixed`l-95% CI`[2],
                                 conf.high=summary(modelobj)$fixed`u-95% CI`[2])
      oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
    }

    oyres_i[, `:=` (indicator=indicator,
                   managed_area=unique(datafile$ManagedAreaName),
                   habitat_class=unique(datafile$HabitatClassification),
                   size_class=sizeclass,
                   live_date_qual=ifelse(
                     str_detect(modelobj$file, "_hist"),
                     "Estimate", "Exact"),

```

```

n_programs=if(
  class(try(datafile$LiveDate_Qualifier)) != "try-error"){
  length(
    unique(
      datafile[LiveDate_Qualifier===
        ifelse(
          str_detect(
            modelobj$file,
            "_hist"),
          "Estimate",
          "Exact"),
        ProgramID)))
} else{length(unique(datafile[, ProgramID]))},
programs=if(class(try(
  datafile$LiveDate_Qualifier)) != "try-error"){
  list(unique(datafile[LiveDate_Qualifier===
    ifelse(
      str_detect(
        modelobj$file,
        "_hist"),
      "Estimate",
      "Exact"),
    ProgramID)))
} else{list(unique(datafile[, ProgramID]))},
  filename=modelobj$file)]
oysterresults_temp <- rbind(oysterresults_temp, oyres_i)
}
} else {
  sizeclass <- ""
}

data <- datafile

ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
  str_detect(indicator, "ensity") ~ "Den",
  str_detect(indicator, "^S|^s") ~ "SH")

if(sizeclass != ""){
  size <- case_when(str_detect(sizeclass, "25") &
    str_detect(sizeclass, "75") ~ "25to75",
    str_detect(sizeclass, "35") &
    str_detect(sizeclass, "75") ~ "35to75",
    str_detect(sizeclass, "25")==FALSE &
    str_detect(sizeclass, "75") ~ "o75", TRUE ~ "raw")
  sizelab <- case_when(str_detect(sizeclass, "25") &
    str_detect(sizeclass, "75") ~ "25-75mm",
    str_detect(sizeclass, "35") &
    str_detect(sizeclass, "75") ~ "35-75mm",
    str_detect(sizeclass, "25")==FALSE &
    str_detect(sizeclass, "75") ~ "\u2265 75mm",
    TRUE ~ "raw")
}

}

```

```

if(ind=="Den"){
  nyrs <- max(data$LiveDate)-min(data$LiveDate)+1
  maxyr <- max(data$LiveDate)
  minyr <- min(data$LiveDate)
  yrdiff <- unique(data$YearDiff)
  current_year <- as.integer(format(Sys.Date(), "%Y"))
  # Creates break intervals for plots based on number of years of data
  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 40 years of data
    brk <- 10
  } else if(nyrs>=20){
    # Set breaks to every 5 years if between 40 and 20 years of data
    brk <- 5
  } else if(nyrs>=12){
    # Set breaks to every 3 years if between 20 and 12 years of data
    brk <- 3
  } else if(nyrs>=8){
    # Set breaks to every 2 years if between 12 and 8 years of data
    brk <- 2
  } else if(nyrs>=5){
    # Set breaks to every year if between 8 and 5 years of data
    brk <- 1
  } else {
    # Ensure 5 years are included on axis
    total_ticks <- 5
    extra_years <- total_ticks - nyrs
    # Always add 1 year before the first year
    years_before <- min(1, extra_years)
    years_after <- extra_years - years_before
    # Adjust min and max year, without going beyond current year
    minyr <- minyr - years_before
    maxyr <- min(maxyr + years_after, current_year)
    # Re-check if we have enough years (in case maxyr hit current year)
    minyr <- max(minyr, maxyr - (total_ticks - 1))
    brk <- 1
  }
  yrlist <- seq(minyr,maxyr,brk)

  if(class(den_glmm)=="brmsfit"){
    denplots <- plot(conditional_effects(models[[1]]), re_formula=NULL, plot=FALSE)
  }

  plot1 <- ggplot() +
    {if("meanDen_int" %in% colnames(data)){
      geom_point(data=data, aes(x=LiveDate,
                                 y=meanDen_int), position=plot_jitter,
                  shape=21, size=2, color="#333333", fill="#cccccc",
                  alpha=0.8, inherit.aes=FALSE)
    } else{
      geom_point(data=data, aes(x=LiveDate,
                                 y=Density_m2), position=plot_jitter,
                  shape=21, size=2, color="#333333", fill="#cccccc",
    }}}
}

```

```

            alpha=0.8, inherit.aes=FALSE)
}} +
{if(class(den_glmm)=="brmsfit"){
  list(geom_ribbon(data=denplots$RelYear$data,
                  aes(x=RelYear+yrdiff, y=Density_m2,
                       ymin=lower__, ymax=upper__),
                  fill="#000099", alpha=0.1, inherit.aes=FALSE),
       geom_line(data=denplots$RelYear$data,
                  aes(x=RelYear+yrdiff,
                      y=estimate__),
                  color="#000099", lwd=0.75, inherit.aes=FALSE))
}} +
scale_x_continuous(limits=c(minyr-0.25, maxyr+0.25),
                    breaks=yrlist) +
scale_y_continuous(breaks = scales::pretty_breaks(n = 6)) +
plot_theme +
{if("meanDen_int" %in% colnames(data)){
  labs(title=paste0("Oyster Density (", habitat_type, ")"),
       subtitle=ma,
       x="Year",
       y=bquote('Estimated density (*~m^{~-2}~*)'))
} else{
  labs(title=paste0("Oyster Density (", habitat_type, ")"),
       subtitle=ma,
       x="Year",
       y=bquote('Density (*~m^{~-2}~*)'))
}}
}

ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

# Specify save location (QAQC Plots saved elsewhere)
if(QAQCPlots){
  file_name <- paste0("output/QAQC/Oyster_Dens_GLMM_", ma_short, "_", habitat_type,
                      ifelse(sizeclass != "", paste0("_", size), "_raw"), ".png")
} else {
  file_name <- paste0("output/Density/Figures/Oyster_Dens_GLMM_", ma_short, "_", habitat_type,
                      ifelse(sizeclass != "", paste0("_", size), "_raw"), ".png")
}

ggsave(file_name,
       plot1,
       width=8,
       height=4,
       units="in",
       dpi=200)
}

cat("---- Density plot created for", ma, "-", habitat_type, "\n")
return(oysterresults_temp)
}

split_tasks <- split(task_list, ceiling(seq_along(1:nrow(task_list)) / 4))
oyster_den <- oysterraw %>% filter(!is.na(Density_m2))

```

```

# Subset and save temp .rds objects for each MA (breaks up oyiterraw)
for(ma in unique(oyster_den$ManagedAreaName)){
  saveRDS(oyster_den[oyster_den$ManagedAreaName == ma, ],
          file = paste0("output/tmp/oysterden_", make.names(ma), ".rds"))
}

results_all <- list()
for(b in seq_along(split_tasks)){
  batch <- split_tasks[[b]]
  results_list <- future_lapply(seq_len(nrow(batch)), function(i) {
    task <- batch[i, ]
    oyiterraw_path <- paste0("output/tmp/oysterden_", make.names(task$ManagedAreaName), ".rds")
    oyiterraw_sub <- readRDS(oyiterraw_path)

    cat(paste0("Now starting ", task$ManagedAreaName, "\n"))

    density_models_par(
      ma = task$ManagedAreaName,
      ma_abrev = task$ma_abrev,
      habitat_type = task$HabitatType,
      oyiterraw_den = oyiterraw_sub
    )

  }, future.seed = TRUE)
  results_all[[b]] <- data.table::rbindlist(results_list, fill = TRUE)
  gc()
}

oysterresults_den <- data.table::rbindlist(results_all, fill = TRUE)
fwrite(oysterresults_den, "output/oyresults/oysterresults_den.csv")

```

Percent Live Analysis

Subsets data for that which is related to percent live. The data is further subset for each managed area. Appropriate GLMMs are called from file for each shell size class, or are created if they don't exist. pctlive_models_par function also creates figures and save data.

```

#Make a collapsed version of the oyiterraw table for percent live
oyiterraw_pct <- oyiterraw[, c("ProgramID", "ProgramName", "ProgramLocationID",
                               "QuadIdentifier", "ReefIdentifier", "LiveDate",
                               "LiveDate_Qualifier", "SampleDate", "Year",
                               "Month", "ManagedAreaName", "Region",
                               "SurveyMethod", "PercentLiveMethod",
                               "HabitatClassification", "QuadSize_m2", "MADup",
                               "PercentLive_pct",
                               "Number_of_Oysters_Counted_Total_Count",
                               "Number_of_Oysters_Counted_Live_Count",
                               "Number_of_Oysters_Counted_Dead_Count",
                               "ObsIndex", "UniversalReefID",
                               "MA_plotlab", "Subtidal", "RelYear", "YearDiff")]
oyiterraw_pct[!is.na(PercentLive_pct), PctIndex := ObsIndex]
oyiterraw_pct[!is.na(Number_of_Oysters_Counted_Total_Count),
             NTotIndex := ObsIndex]

```

```

oysterraw_pct[!is.na(Number_of_Oysters_Counted_Live_Count),
              NLiveIndex := ObsIndex]
oysterraw_pct[!is.na(Number_of_Oysters_Counted_Dead_Count),
              NDeadIndex := ObsIndex]
oysterraw_pct[, ObsIndex := NULL]

oysterraw_pct <- unique(oysterraw_pct)
oysterraw_pct <- oysterraw_pct %>%
  dplyr::group_by(ProgramID, ProgramName, ProgramLocationID, QuadIdentifier,
                  ReefIdentifier, LiveDate, LiveDate_Qualifier, SampleDate,
                  Year, Month, ManagedAreaName, Region, SurveyMethod,
                  PercentLiveMethod, HabitatClassification, QuadSize_m2,
                  MADUp, UniversalReefID, MA_plotlab, Subtidal,
                  RelYear) %>%
  tidyr::fill(PercentLive_pct, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              PctIndex, NTotIndex, NLiveIndex, NDeadIndex) %>%
  tidyr::fill(PercentLive_pct, Number_of_Oysters_Counted_Total_Count,
              Number_of_Oysters_Counted_Live_Count,
              Number_of_Oysters_Counted_Dead_Count,
              PctIndex, NTotIndex, NLiveIndex, NDeadIndex,
              .direction='up') %>%
  dplyr::distinct()

oysterraw_pct <- subset(oysterraw_pct, !is.na(oysterraw_pct$PercentLive_pct) |
                           !is.na(oysterraw_pct$Number_of_Oysters_Counted_Total_Count) |
                           !is.na(oysterraw_pct$Number_of_Oysters_Counted_Live_Count) |
                           !is.na(oysterraw_pct$Number_of_Oysters_Counted_Dead_Count) |
                           !is.na(oysterraw_pct$PctIndex) |
                           !is.na(oysterraw_pct$NTotIndex) |
                           !is.na(oysterraw_pct$NLiveIndex) |
                           !is.na(oysterraw_pct$NDeadIndex))
setDT(oysterraw_pct)

#Calculate PercentLive_pct values for some ProgramIDs where it is missing.
#Couldn't include at the start of the script because need to use the counts columns
#rather than the QuadSize_m2 column which is filled for the whole combined table.
oysterraw_pct[ProgramID==972 | ProgramID==4014 | ProgramID==4044,
             PercentLive_pct := 
               (Number_of_Oysters_Counted_Live_Count /
                (Number_of_Oysters_Counted_Live_Count +
                 Number_of_Oysters_Counted_Dead_Count) * 100)]

#Filter NAs for PercentLive_pct (these are related to 1) programs that do
#counts to measure density, but do not estimate percent live and
#2) Programs that are listed as measuring percent live by a Point-intercept
#method, which cannot be calculated from counts.
oysterraw_pct <- oysterraw_pct[!is.na(PercentLive_pct), ]

#Add column of decimal versions of percent live values
oysterraw_pct[, PercentLive_dec := PercentLive_pct/100]

```

```

#Summarize percent live values
pct_all_sum <- summarySE(oysterraw_pct, measurevar='PercentLive_pct',
                           groupvars=c('ManagedAreaName', 'Year', 'PercentLiveMethod'))

# Find out which MAs should receive models for Percent Live
pct_stats <- ma_stats[["Percent Live"]][["MA_Ov_Stats"]] %>%
  select(ManagedAreaName, ParameterName, HabitatType) %>%
  as.data.table()

task_list <- pct_stats[, .(HabitatType = unique(HabitatType)), by = ManagedAreaName]
# Add MA abbreviation
task_list[, ma_abrev := MA_All[.SD, on = "ManagedAreaName", Abbreviation]]
task_list <- as.data.frame(task_list)

pctlive_models_par <- function(ma, ma_abrev, habitat_type, oysterraw_pct){
  library(data.table)
  # Combined MA name with habitat type
  ma_plotlabel <- paste0(ma, "_", str_to_title(habitat_type))
  # At least 5 years of data are required in order to run model analyses
  # Function checks N years of data, returns T or F
  suff_years <- function(data){length(unique(data$Year))>=5}
  # Create subset for each MA
  ma_subset <- subset(oysterraw_pct, oysterraw_pct$MA_plotlab==ma_plotlabel)
  # Exception to only run model on Percent data for LBAP (exclude 8 values from 5035 which are point-in
  if(ma_abrev=="LBAP"){ma_subset <- ma_subset[!PercentLiveMethod=="Point-intercept"]}
  # Save data used in model
  saveRDS(ma_subset, paste0("output/model_results/data/", ma_abrev, "_PrcLive_", Sys.Date(), "_", habitat_type))
  # If enough years of data, perform modelling. If not, plot data points only
  if(suff_years(ma_subset) & !QAQCPplots){
    cat("---- Sufficient years of data. \n")
    #PercentLiveMethod=="Percent" for Lemon Bay program(s) with sufficient data,
    #so cannot be modeled as binomial
    if(ma_abrev=="LBAP"){
      pct_glm <- brm(
        formula=PercentLive_dec ~ RelYear + (0 + RelYear | ReefIdentifier),
        data=subset(ma_subset, ma_subset$PercentLive_dec > 0 & ma_subset$PercentLive_dec < 1),family=Binomial,
        cores=ncores, control= list(adapt_delta=0.995, max_treedepth=20),
        iter=iter, warmup=warmup, chains=nchains, init=0, thin=3, seed=8465,
        backend="rstan", save_pars = save_pars(all = TRUE),
        file=paste0("output/model_results/GLMMs/", ma_abrev, "_pct_glm_", habitat_type, ".rds")
      )
    } else {
      # Check to see if previous model already exists
      model_file <- paste0("output/model_results/GLMMs/", ma_abrev, "_pct_glm_", habitat_type, ".rds")
      if(!file.exists(model_file)){
        ma_subset <- as.data.frame(ma_subset)
        ma_subset$LiveSuccess <- round(ma_subset$PercentLive_pct)
        ma_subset$Trials <- 100

        # ma_subset[, PercentLive_pct_rounded := round(PercentLive_pct)]
        # ma_subset[, LiveObs := lapply(PercentLive_pct_rounded, function(pct) {
        #   c(rep(1L, pct), rep(0L, 100 - pct))
      }
    }
  }
}

```

```

# })]
# binom <- ma_subset[, .(LiveObs = unlist(LiveObs)),
#                      by = .(ProgramID, ProgramLocationID, QuadIdentifier, Year,
#                             ManagedAreaName, PercentLiveMethod, UniversalReefID,
#                             Region, MA_plotlab, Relyear, PercentLive_pct)]

# Save binomial data (used in model)
saveRDS(ma_subset, paste0("output/model_results/data/", ma_abrev, "_PrcLive_binom_", Sys.Date()))

# Run model
cat("----- Running binomial model \n")
pct_glmm <- brm(
  formula=LiveSuccess | trials(Trials) ~ RelYear + (1 | UniversalReefID),
  data=ma_subset, family=binomial, cores=ncores,
  control= list(adapt_delta=0.995, max_treedepth=20),
  iter=iter, warmup=warmup, chains=nchains, init=0, thin=3,
  seed=4331, backend="rstan", save_pars = save_pars(all = TRUE),
  file=paste0("output/model_results/GLMMs/", ma_abrev, "_pct_glmm_", habitat_type, ".rds"))
)
} else {
  pct_glmm <- readRDS(model_file)
}
}
} else {
  pct_glmm <- NA
}

# Create model results tables and save diagnostic plots and marginal effects plots
datafile <- setDT(ma_subset)
models <- list(pct_glmm)
indicator <- "Percent live"
meplotzoom <- FALSE
oysterresults_temp <- data.frame()

if(class(pct_glmm)=="brmsfit"){
  for(m in seq_along(models)){
    modelobj <- models[[m]]
    sizeclass <- ifelse(str_detect(modelobj$file, "25to75|seed"),
                         "25-75mm",
                         ifelse(str_detect(modelobj$file, "35to75|seed"),
                               "35-75mm",
                               ifelse(str_detect(modelobj$file,
                                                 "o75|market"),
                                     ">75mm", "NA")))
    oyres_i <- setDT(broom.mixed::tidy(modelobj))
    #tidy() does not like that parameter values have underscores for
    #some reason, so the resulting table is incomplete

    if(nrow(oyres_i[effect=="fixed", ]) - nrow(summary(modelobj)$fixed) == -1){
      missingrow <- data.table(effect="fixed",
                                 component="cond",
                                 #not sure what "cond" means in the tidy summary.
                                 group=NA,
                                 term=rownames(summary(modelobj)$fixed)[2],
                                 estimate=summary(modelobj)$fixed$Estimate[2],

```

```

        std.error=summary(modelobj)$fixed$Est.Error[2],
        conf.low=summary(modelobj)$fixed`^1-95% CI`[2],
        conf.high=summary(modelobj)$fixed`^u-95% CI`[2])
oyres_i <- rbind(oyres_i, missingrow) %>% arrange(effect, group)
}

oyres_i[, `:=` (indicator=indicator,
  managed_area=unique(datafile$ManagedAreaName),
  habitat_class=unique(datafile$HabitatClassification),
  size_class=sizeclass,
  live_date_qual=ifelse(
    str_detect(modelobj$file, "_hist"),
    "Estimate", "Exact"),
  n_programs=if(
    class(try(datafile$LiveDate_Qualifier)) != "try-error"){
    length(
      unique(
        datafile[LiveDate_Qualifier===
          ifelse(
            str_detect(
              modelobj$file,
              "_hist"),
            "Estimate",
            "Exact"),
          ProgramID]))),
  } else{length(unique(datafile[, ProgramID]))},
  programs=if(class(try(
    datafile$LiveDate_Qualifier)) != "try-error")){
    list(unique(datafile[LiveDate_Qualifier===
      ifelse(
        str_detect(
          modelobj$file,
          "_hist"),
        "Estimate",
        "Exact"),
      ProgramID)))
  } else{list(unique(datafile[, ProgramID]))},
  filename=modelobj$file)]
oysterresults_temp <- rbind(oysterresults_temp, oyres_i)
}
} else {
  sizeclass <- ""
}

data <- datafile

ind <- case_when(str_detect(indicator, "ercent") ~ "Pct",
  str_detect(indicator, "ensity") ~ "Den",
  str_detect(indicator, "^S|^s") ~ "SH")

if(sizeclass != ""){
  size <- case_when(str_detect(sizeclass, "25") &

```

```

        str_detect(sizeclass, "75") ~ "25to75",
        str_detect(sizeclass, "35") &
        str_detect(sizeclass, "75") ~ "35to75",
        str_detect(sizeclass, "25")==FALSE &
        str_detect(sizeclass, "75") ~ "o75", TRUE ~ "raw")
sizeLab <- case_when(str_detect(sizeclass, "25") &
                      str_detect(sizeclass, "75") ~ "25-75mm",
                      str_detect(sizeclass, "35") &
                      str_detect(sizeclass, "75") ~ "35-75mm",
                      str_detect(sizeclass, "25")==FALSE &
                      str_detect(sizeclass, "75") ~ "\u2265 75mm",
                      TRUE ~ "raw")
}

if(ind=="Pct"){
  nyrs <- max(data$LiveDate)-min(data$LiveDate)+1
  maxyr <- max(data$LiveDate)
  minyr <- min(data$LiveDate)
  yrdiff <- unique(data$YearDiff)
  current_year <- as.integer(format(Sys.Date(), "%Y"))
  # Creates break intervals for plots based on number of years of data
  if(nyrs>=40){
    # Set breaks to every 10 years if more than 40 years of data
    brk <- 10
  } else if(nyrs>=20){
    # Set breaks to every 5 years if between 40 and 20 years of data
    brk <- 5
  } else if(nyrs>=12){
    # Set breaks to every 3 years if between 20 and 12 years of data
    brk <- 3
  } else if(nyrs>=8){
    # Set breaks to every 2 years if between 12 and 8 years of data
    brk <- 2
  } else if(nyrs>=5){
    # Set breaks to every year if between 8 and 5 years of data
    brk <- 1
  } else {
    # Ensure 5 years are included on axis
    total_ticks <- 5
    extra_years <- total_ticks - nyrs
    # Always add 1 year before the first year
    years_before <- min(1, extra_years)
    years_after <- extra_years - years_before
    # Adjust min and max year, without going beyond current year
    minyr <- minyr - years_before
    maxyr <- min(maxyr + years_after, current_year)
    # Re-check if we have enough years (in case maxyr hit current year)
    minyr <- max(minyr, maxyr - (total_ticks - 1))
    brk <- 1
  }
  yrlist <- seq(minyr,maxyr,brk)

set.seed(987)

```

```

if(class(pct_glmm)=="brmsfit"){
  pctplots <- plot(conditional_effects(models[[1]]), re_formula=NULL, plot=FALSE)
}

plot1 <- ggplot() +
  geom_point(data=data, aes(x=LiveDate,
                             y=100*PercentLive_dec), position=plot_jitter,
             shape=21, size=2, color="#333333", fill="#cccccc",
             alpha=0.8, inherit.aes=FALSE) +
  if(class(pct_glmm)=="brmsfit"){
    list(
      geom_ribbon(data = pctplots$RelYear$data,
                  aes(x = RelYear + yrdiff,
                      y = 100 * estimate__,
                      ymin = 100 * lower__,
                      ymax = 100 * upper__),
                  fill = "#000099", alpha = 0.1, inherit.aes = FALSE),
      geom_line(data = pctplots$RelYear$data,
                 aes(x = RelYear + yrdiff,
                     y = 100 * estimate__),
                 color = "#000099", lwd = 0.75, inherit.aes = FALSE)
    )
  }
} +
  scale_x_continuous(limits=c(minyr-0.25, maxyr+0.25),
                     breaks=yrlist) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  plot_theme +
  theme(legend.text=element_text(size=10),
        legend.title=element_text(size=10)) +
{
  if(unique(ma_subset$PercentLiveMethod)=="Percent"){
    labs(title=paste0("Percent Live Oysters (", habitat_type, ")"),
         subtitle=ma,
         x="Year",
         y="Percent live (%)")
  } else {
    labs(title=paste0("Oyster Percent Live Cover (", habitat_type, ")"),
         subtitle=ma,
         x="Year",
         y="Live cover (%)")
  }
}

ma_short <- MA_All[ManagedAreaName==ma, Abbreviation]

# Specify save location (QAQC Plots saved elsewhere)
if(QAQCPlots){
  file_name <- paste0("output/QAQC/Oyster_PrcLive_GLMM_",
                      ma_short, "_", habitat_type, "_raw.png")
} else {
  file_name <- paste0("output/Percent_Live/Figures/Oyster_PrcLive_GLMM_",
                      ma_short, "_", habitat_type, "_raw.png")
}

```

```

ggsave(file_name,
       plot1,
       width=8,
       height=4,
       units="in",
       dpi=200)
}

cat("---- Percent Live plot created for", ma, "-", habitat_type, "\n")
return(oysterresults_temp)
}

split_tasks <- split(task_list, ceiling(seq_along(1:nrow(task_list)) / 4))
# Subset and save temp .rds objects for each MA (breaks up oysterraw)
for(ma in unique(oysterraw_pct$ManagedAreaName)){
  saveRDS(oysterraw_pct[oysterraw_pct$ManagedAreaName == ma, ],
          file = paste0("output/tmp/oysterpct_", make.names(ma), ".rds"))
}

results_all <- list()
for(b in seq_along(split_tasks)){
  batch <- split_tasks[[b]]
  results_list <- future_lapply(seq_len(nrow(batch)), function(i) {
    task <- batch[i, ]
    oysterraw_path <- paste0("output/tmp/oysterpct_", make.names(task$ManagedAreaName), ".rds")
    oysterraw_sub <- readRDS(oysterraw_path)

    cat(paste0("Now starting ", task$ManagedAreaName, "\n"))

    pctlive_models_par(
      ma = task$ManagedAreaName,
      ma_abrev = task$ma_abrev,
      habitat_type = task$HabitatType,
      oysterraw_pct = oysterraw_sub
    )

  }, future.seed = TRUE)
  results_all[[b]] <- data.table::rbindlist(results_list, fill = TRUE)
  gc()
}

oysterresults_pct <- data.table::rbindlist(results_all, fill = TRUE)
fwrite(oysterresults_pct, "output/oysteresults/oysterresults_pct.csv")

```

Save & Export Results

The compiled model results variable is saved to a csv and rds file. The model results are evaluated to see if any models need to be reconsidered based on the rhat convergence being over 1.05. rhat values are written to file. All figures are then compressed into .zip files for each respective indicator.

```

# Combine all results into a single file for processing (Oyster_ResultsCompile.R)
all_oysterresults <- bind_rows(oysterresults_sh,oysterresults_den, oysterresults_pct)

fwrite(all_oysterresults, paste0("output/GLMM_AllDates_ModelResults.csv"), sep=",")

```

```

saveRDS(all_oysterresults, paste0("output/GLMM_AllDates_ModelResults.rds"))

#Get Rhat values for all models to check which ones may need to be reparameterized
model_list <- unique(all_oysterresults$filename)
rhats_all <- data.table(filename=character(),
                         term=character(),
                         rhat=numeric())
rhats_sum <- data.table(filename=character(),
                         rhat=numeric())
fam_overview <- data.table(filename=character(),
                            family=character())

for(mod in model_list){
  mod_i <- readRDS(mod)
  allrhat_i <- rhat(mod_i)
  sumrhat_i <- c(summary(mod_i)$fixed$Rhat, summary(mod_i)$spec_pars$Rhat)
  allrhat_model_i <- data.table(filename=mod,
                                  term=names(allrhat_i),
                                  rhat=allrhat_i)
  sumrhat_model_i <- data.table(filename=mod,
                                  rhat=sumrhat_i)
  rhats_all <- rbind(rhats_all, allrhat_model_i)
  rhats_sum <- rbind(rhats_sum, sumrhat_model_i)
  sum <- summary(mod_i)
  familyType <- sum$formula$family$family
  fam_overview <- rbind(fam_overview, data.table(filename=mod,family=familyType))
}

# extract MA_abrev from model names
fam_overview <- fam_overview %>% rowwise() %>% mutate(
  ma_abrev = str_split_1(tail(str_split_1(filename, "/"), 1), "_")[1],
  parameter = str_split_1(tail(str_split_1(filename, "/"), 1), "_")[2],
  habtype = str_split_1(tail(str_split_1(tail(str_split_1(filename, "/"), 1), "_"), 1), ".rds")[1],
  ma = MA_All[Abbreviation==ma_abrev, ManagedAreaName],
  param = ifelse(parameter=="den", "Density", ifelse(parameter=="pct", "Percent Live", "Shell Height"))
) %>% as.data.table()

rhats_all[, rhat_r := round(rhat, 2)]
rhats_sum[, rhat_r := round(rhat, 2)]

saveRDS(rhats_all, paste0("output/rhats_all_", Sys.Date(), ".rds"))
saveRDS(rhats_sum, paste0("output/rhats_sum_", Sys.Date(), ".rds"))
saveRDS(fam_overview, paste0("output/model_family_overview", Sys.Date(), ".rds"))

models_to_check_allrhat <- unique(rhats_all[rhat_r > 1.05, filename])
models_to_check_sumrhat <- unique(rhats_sum[rhat_r > 1.05, filename])

# Zip all figures
for(p in c("Density", "Percent_Live", "Shell_Height")){
  out_dir <- paste0("output/", p)
  fig_list <- list.files(paste0(out_dir, "/Figures"), pattern=".png", full=FALSE)
  filename <- paste0("Oyster", gsub("_", "", p), "Figures")
  setwd(paste0(out_dir, "/Figures"))
}

```

```
    zip(filename, files=fig_list)
    setwd(wd)
}
```