Development of Deterioration Curves for Ohio Bridges

***Prepared by*:**

Yoojung Yoon, Ph.D.
Faysal Ahamed
Jai Lee, PE
Travis Butz, PE
Olya Watts, PE

***Prepared for*:**

Ohio's Research Initiative for Locals
The Ohio Department of Transportation,
Office of Statewide Planning & Research

Project ID Number: 115890

January 2024

Final Report

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
FHWA/OH-2024/08			
4. Title and Subtitle		5. Report Date	
Development of Deterioration Curves for Ohio Bridges		**January 2024**	
		6. Performing Organization Code	
7. Author(s)		8. Performing Organization Report No.	
Yoojung Yoon; Faysal Ahamed; Jai Lee; Travis Butz; and Olya Watts			
9. Performing Organization Name and Address		10. Work Unit No. (TRAIS)	
West Virginia University 886 Chestnut Ridge Road, PO Box 6845, Morgantown, WV 26506-6845		11. Contract or Grant No. **37853**	
12. Sponsoring Agency Name and Address		13. Type of Report and Period Covered	
Ohio Department of Transportation 1980 West Broad Street Columbus, Ohio 43223		**Final Report** 14. Sponsoring Agency Code	
15. Supplementary Notes			
Prepared in cooperation with the Ohio Department of Transportation (ODOT)			
16. Abstract			

The objective of this research was to develop the deterioration curves of the primary bridge superstructure designs to understand their characteristics over time. The research methodologies involved a meticulous data collection and processing step, analyzing Ohio's historical bridge inventory dating back to the mid-1980s, followed by deterioration model development and comparative analysis. A regression nonlinear optimization (RNO) model was applied to develop deterioration curves for each superstructure type, employing Python scripts for plotting the best-fit polynomial regression curves and MS Excel solver for Markovian transition probabilities. Furthermore, a comparative analysis was conducted, examining deterioration characteristics among different superstructure designs for each maintenance responsibility (e.g., state, county, and city/municipality) and average annual degradation rates over a 5-year age range. The comparison of deterioration curves for six bridge structure types owned by the state DOT indicates that stringer beam and slab designs demonstrate superior durability over time. However, box beam designs exhibit rapid deterioration as they age. In county and city/municipality settings, slab designs generally show more gradual deterioration, while frame designs display early deterioration patterns. Data also suggests variations in degradation rates among different design types, emphasizing the influence of construction quality and design longevity on bridge performance.

17. Keywords		18. Distribution Statement	
Deterioration Curves, Superstructures, Bridge Management, Bridge Deterioration		**No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161**	
19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified	**Unclassified**		

Form DOT F 1700.7 (8-72) **Reproduction of completed pages authorized**

Credits and Acknowledgments

Prepared in cooperation with the Ohio Department of Transportation

and the U.S. Department of Transportation, Federal Highway Administration

The research team expresses its gratitude to the Ohio Department of Transportation (ODOT), Ohio's Research Initiative for Locals (ORIL), and the Federal Highway Administration (FHWA) for their sponsorship of this study. We extend our sincere appreciation to the members of the Technical Advisory Committee (TAC) for their invaluable guidance, comments, and constructive suggestions: Mr. Warren Schlatter, Mr. Brad Noll, Mr. Mark Droll, Mr. Steve Bergstresser, and Mr. Brion Rhodes. Acknowledgment is also due to the various local public agencies that generously provided the data utilized in this research. We would also like to thank Mike Brokaw of the Federal Highway Administration (formerly of ODOT) for his assistance in gathering the data. A special note of appreciation goes to Ms. Vicky Fout for her dedicated time and invaluable assistance throughout this project.

Table of Contents

List of Figures

List of Tables

1. PROBLEM STATEMENT

The American Society of Civil Engineers (ASCE) *2021 Report Card for America's Infrastructure* highlighted Ohio's inventory of 44,736 bridges exceeding a 10-foot span, maintained by either the Ohio Department of Transportation (ODOT) or local public agencies (LPAs). Of these, 58% are in good condition, while 36% are deemed satisfactory or fair, leaving 6% in a poor state—categorized as structurally deficient (ASCE, 2021). Notably, Ohio's rate of structurally deficient bridges (6%) is lower than the national average of 7.5%. Ohio's position as the state with the second-highest number of bridges in the nation poses a substantial challenge. In the 2020 ODOT bridge inventory, a total of 2,843 bridges were flagged as structurally deficient (ASCE, 2021). The American Road & Transportation Builders Association's (ARTBA) annual bridge report based on the 2021 National Bridge Inventory (NBI) data indicates that Ohio has 1,334 structurally deficient NBI bridges, ranking 10th highest among U.S. States (ARTBA, 2022).

ODOT oversees a robust bridge inventory and inspection system aimed at consistently safeguarding Ohio's bridge quality. The recent adoption of AssetWise, replacing the previous Structure Management System (SMS) in 2020, signifies a pivotal shift (ODOT, n.a). AssetWise, comprising essential inspection and maintenance modules, utilizes reactive and recurring maintenance data that is crucial for deciphering genuine maintenance-induced deterioration patterns in bridges (Yoon and Hastak, 2016). ODOT's AssetWise system holds data for about 45,000 bridges of various bridge superstructure types, such as slab, stringer/multi-beam or girder, box beam or girders, and frame. With the advancements in materials and construction techniques, bridge design types have evolved to accommodate various requirements, including increased traffic loads, longer spans, aesthetic considerations, and higher structural durability (Upadhya et al., 2021). As of 2022, Table 1 shows an itemization of Ohio bridge superstructures and their respective proportions.

Table 1. Ohio Bridge Superstructure Design Types and Proportions

Superstructure Type	State-Owned	Non-State Owned	No. of Bridges	Percentage
Slab	2,710	3,534	6,244	14%
Stringer/Multi-beam or Girder	5,778	6,360	12,138	27%
Girder and Floor beam System	85	615	700	2%
Tee Beam	44	945	989	2%
Box Beam or Girders - Multiple	1,412	7,463	8,875	20%
Box Beam or Girders - Single or Spread	2	18	20	0%
Frame (except frame culverts)	425	3,061	3,486	8%
Orthotropic	-	2	2	0%
Truss - Deck	5	17	22	0%
Truss - Thru	25	1,100	1,125	2%
Arch - Deck	69	362	431	1%
Arch - Thru	7	17	24	0%
Suspension	4	3	7	0%
Stayed Girder	8	8	16	0%
Movable - Lift	-	5	5	0%
Movable - Bascule	4	1	5	0%
Movable - Swing	-	1	1	0%
Culvert (includes frame culverts)	3,749	7,151	10,900	24%
Mixed types	-	1	1	0%
Segmental Box Girder	-	-	-	0%
Channel Beam	-	-	-	0%

Under the Ohio Revised Code (ORC), bridges across state and local transportation systems undergo mandatory annual inspections, surpassing the Federal Highway Administration's (FHWA) biennial inspection requirement for state DOTs (ASCE, 2021). Ohio's unique status as the sole state with this elevated inspection frequency has generated an expansive dataset of historical bridge inventory and inspection records, proving invaluable for effective bridge management. Evaluating the deterioration process of bridges is a fundamental part of effective bridge management to generate strategies at both the project (i.e., individual bridges) and network (i.e., collection of individual bridges) levels, considering agency-defined goals and constraints. Moreover, comprehending the variations in time-related degradation processes among different types of bridge superstructure designs and across local and statewide scales can assist state and local authorities in choosing the suitable superstructure type for specific circumstances. However, there is a gap in such research conducted on the state of Ohio, despite its greater inspection frequency, generating an extensive data set of historical bridge inventory and inspection data available for bridge management.

2. RESEARCH BACKGROUND

2.1. Goals and Objectives

The goal of this research was to enhance bridge owners' understanding of projected service life and performance of specific bridge superstructure types. The objective of this research was to develop deterioration curves based on the NBI historical condition data for common superstructure types in Ohio.

2.2. Scope of the Research

Deterioration curves were developed for bridges under the maintenance responsibility of state DOT, county, and local/municipal entities in Ohio. Deterioration curves were developed for each superstructure type that exceeded 3% of the Ohio bridge population. These superstructure types include slab, stringer/multi-beam or girder, box beam or girders – multiple, and frame, as indicated in Table 1. Some design types were then further divided into smaller groups based on main span material codes and wearing surface types, resulting in a total of six design types as follows:

- Slab bridges
- Stringer/multi-beam or girder bridges with prestressed (PS) concrete
- Stringer/multi-beam or girder bridges with steel beam
- Box beam or girders – multiple with asphalt (AS) wearing surface (WS)
- Box beam or girders – multiple with concrete (Conc) deck
- Frame except for frame culverts

The purpose of these deterioration curves is to show the natural deterioration patterns of bridges over their lifespan without considering any improvements resulting from maintenance interventions. Also, it should be noted that the data analysis to develop deterioration models used the data points of historical condition ratings with no consideration of internal and external factors that may affect the bridge deterioration process. This research utilized the Markov transition modeling technique, extensively used by state Departments of Transportation (DOTs) and AASHTOWare Bridge Management (BrM), to achieve its research purpose, instead of the popular artificial intelligence (AI)-based deterioration models, while artificial intelligence (AI)-based deterioration models are currently in trend. Section 7.1 presents detailed discussions of various factors that influence the bridge deterioration process and analysis models applicable to bridge deterioration.

2.3. Specific Tasks to be Accomplished

The specific tasks accomplished to achieve the research objective and scope include:

- Project management: a project start-up meeting and monthly status calls
- Data collection and processing for reliable data analysis
- Deterioration model development, applying regression non-linear optimization (RNO), which is a Markovian-based non-linear optimization model
- Comparative analysis of deterioration curves

2.4. Summary of Key Literature Search Findings

A summary of the key findings of state DOT research on bridge deterioration curves, factors for deterioration curve analysis, and deterioration analysis models are presented in this section. A complete literature review on these topics is included in Appendix 7.1.

2.4.1. State DOTs Research for Bridge Deterioration Curves

The investigation of state DOTs research on bridge deterioration curves includes reports from multiple state DOTs, including Colorado, Florida, Illinois, Indiana, Michigan, Minnesota, Montana, Nebraska, North Carolina, Ohio, Wisconsin, and Wyoming, as well as one regional state DOT consortium, Midwest state DOTs. The primary objective of this research was to develop deterioration models for bridge components or elements by analyzing historical condition data. A conventional approach to developing deterioration curves involved categorizing bridge structures based on internal and external factors known to influence the deterioration process. While the majority of research applied traditional Markovian-based probabilistic models for deterioration curves, Michigan DOT utilized artificial neural networks to predict the future condition ratings of concrete bridge decks.

2.4.2. Factors for Deterioration Curve Analysis

Bridge deterioration results from various factors, so it is essential to clarify these factors to formalize the deterioration process into a mathematical or statistical framework and reduce the dimensionality of deterioration models by focusing on the factors that significantly influence deterioration. The literature review of this research identified a comprehensive list of influential factors, which were categorized as follows:

- Age: age from construction or reconstruction
- Design: number of spans, bridge roadway width, deck area, degree of skew, design load, maximum span length, rebar protection, structure length, structure type, and use of deck overlays
- Material: approach surface type, deck material, structural material, superstructure material, and type of wearing surface
- Reconstruction: structure improvement length and presence of reconstruction
- Functional class: functional classification
- Operation: annual daily traffic, average daily truck traffic, percent truck traffic, service under a bridge, and current bridge condition
- Environment: climatic conditions, freeze-thaw cycles, and number of cold days
- Region: district and geographic region

2.4.3. Deterioration Analysis Models

Bridge deterioration models can be categorized into four main groups: deterministic, mechanistic, stochastic, and AI-based. Deterministic models, like regression and curve-fitting, rely on predetermined patterns influenced by specific factors. On the other hand, stochastic models, such as state-based Markov chains and time-based Weibull distributions, factor in uncertainty and randomness. Mechanistic models delve into the physical processes driving deterioration, while AI models, like ANNs and case-based reasoning (CBR), leverage computer learning techniques to identify patterns from historical data. Although each type of deterioration model has its advantages and limitations, regression and Markov chain approaches are favored due to their simplicity and practicality.

3. RESEARCH APPROACH

The steps taken to conduct the research consisted of project management, data collection and processing, deterioration model development, and comparative analysis of deterioration curves, as illustrated in Figure 1. The final report serves as the deliverable that documents the completion of research activities, results, and findings. A comprehensive explanation of each individual step is provided in the subsequent subsections.

Figure 1. Steps Conducted for the Research

3.1. Project Management

The objective of the project management step was to facilitate open communication between the research team and the Technical Advisory Committee (TAC) to ensure successful project execution. This step included a project start-up meeting and monthly status calls for project progress. The start-up meeting took place on February 1, 2023. The agenda covered three main areas: 1) introducing the TAC members and the research team, 2) reviewing project details such as contractual obligations, scope, approach, deliverables, schedule, office policies, and procedures, and 3) addressing any technical concerns or questions. The monthly status calls of the project, lasting between 30 to 60 minutes, were scheduled each month from March to November. Each monthly status call provided updates on progress since the previous meeting, identified any encountered problems and planned solutions, and discussed any technical issues requiring TAC's expertise and suggestions.

3.2. Data Collection and Processing

3.2.1. Data Collection

The main data collection was conducted using the in-house ODOT database, which offered a historical inventory of bridges at the state and local levels (counties and cities) dating back to the mid-1980s. The raw data in the coded items (i.e., Items 1 through 116, including reserved items), following the FHWA's recording and coding guide for the nation's bridges (FHWA, 1995), were stored as a Microsoft Excel worksheet format. The data types required for data analysis to develop the deterioration curves were extracted from the database. These data types, along with the associated FHWA item numbers, are as follows:

- Structure number (Item 8), as bridge identifiers
- Facility carried by structure (Item 7), to limit to road bridges only
- Main structure type (Item 43), to identify the specific superstructure types and materials used
- Maintenance responsibility (Item 21), to label bridges as either state, county, or city/municipality
- Historical condition rating (Item 59), for bridge superstructures
- Type of wearing surfaces (Item 108A), for asphalt and concrete
- Year built (Item 27) and inspection date (Item 90), to determine bridge ages
- Year reconstructed (Item 106), to reset bridge ages

The number of bridges identified in the raw data for each superstructure type is presented in Table 2.

Table 2. Number of Bridges for Each Superstructure Type

Superstructure Type	Maintenance Responsibility			Total
	State	County	City/ Municipality	
Stringer PS Beam	568	315	67	950
Stringer Steel Beam	6,131	6,836	412	13,379
Box beam Asphalt Wearing Surface	1,102	6,308	269	7,679
Box beam Concrete Deck	286	493	150	929
Slab	3,052	2,520	235	5,807
Frame	163	689	173	1,025
Total	11,302	17,161	1,306	29,769

3.2.2. Data Processing

The activities for data processing included data removal for unreliable bridges and data points, bridge age reset to remove the effect of treatment activities, including maintenance, repair, and rehabilitation, and data filtering. Python scripts with analytical functions were utilized for data removal. The flowchart and Python scripts for data processing are presented in Appendices 7.2 and 7.3.

Data Removal

The data removal first involved aggregating inspection condition ratings collected at different time points (either annually or biennially) for the same structure, creating a time-series dataset. The aggregated historical condition ratings for each bridge structure were examined to identify missing, inconsistent, and noisy data points. The bridges with unrecoverable condition ratings were removed to ensure the reliability of the data analysis results. Unrecoverable data points were defined as:

- Missing condition ratings for more than two consecutive inspections, making it impossible to infer them from the condition ratings recorded both before and after the missing data points.
- Inconsistent condition ratings showing year-to-year fluctuations in conditions.
- Noisy condition ratings with more than one condition rating decreased (e.g., from 9 to 7) but maintained the same condition rating (e.g., 7) in subsequent inspection records.

Bridges and data points were also removed for the conditions listed below:

- Reconstruction or rehabilitation-related data missing, making it difficult to precisely reset the age of a bridge even with high condition ratings.
- Older bridges that, even after 60 years of age, continue to have exceptionally high condition ratings (e.g., 8, 7, or 6).
- Instances of incorrectly entered data where the year of inspection comes before the bridge's construction.
- Condition ratings below 3, which are considered undesirable and unacceptable for any bridge controlled by the state DOTs.

Table 3 lists the number of data points before and after data processing for each superstructure type.

Table 3. Data Points before and after Data Processing for the Selected Superstructure Types

Maintenance Responsibility	Data Processing	Stringer PS Beam	Stringer Steel Beam	Box Beam AS WS	Box Beam Conc Deck	Slab	Frame
State	Before	9,858	197,854	34,469	7,005	112,747	9,402
	After	8,476	167,042	29,439	5,845	95,477	7,496
	Ratio	86.0%	84.4%	85.4%	83.4%	84.7%	79.7%
County	Before	7,195	210,751	172,371	9,859	143,567	48,548
	After	6,187	177,452	147,915	8,549	120,707	39,392
	Ratio	86.0%	84.2%	85.8%	86.7%	84.1%	81.1%
City/ Municipality	Before	1,149	10,724	6,308	2,917	8,214	5,954
	After	976	8,974	5,329	2,479	6,704	3,131
	Ratio	84.9%	83.7%	84.5%	85.0%	81.6%	52.6%

Bridge Age Reset

When the bridge superstructure was replaced, the bridge age was reset to "0." When the bridge was repaired or rehabilitated, the bridge age was reset to the age at which the post-rehabilitation condition rating originally occurred. Figure 2 illustrates this process for a 20-year-old bridge that is rehabilitated to a condition rating of "8." The improved condition rating corresponds to the condition rating of the bridge 3 years after construction. Therefore, the original age of the bridge is reset to the age of 3 in this case. Resetting ages results in the segmentation of a single time-series dataset of condition ratings for each bridge structure into multiple time-series datasets, as seen in Figure 3.

Age of CR 8 showing first

Original Age	0	1	2	3	4	5	6	7	8	……	14	15	16	17	18	19	20	21	22	23	24	25
Condition Rating	9	9	9	8	8	8	8	8	7	……	7	7	7	7	6	6	8	8	8	7	7	7
Reset Age	0	1	2	3	4	5	6	7	12	……	14	15	16	17	18	19	3	4	5	6	7	8

Rehabilitation

Figure 2. Example of Age Reset due to Treatment Effect

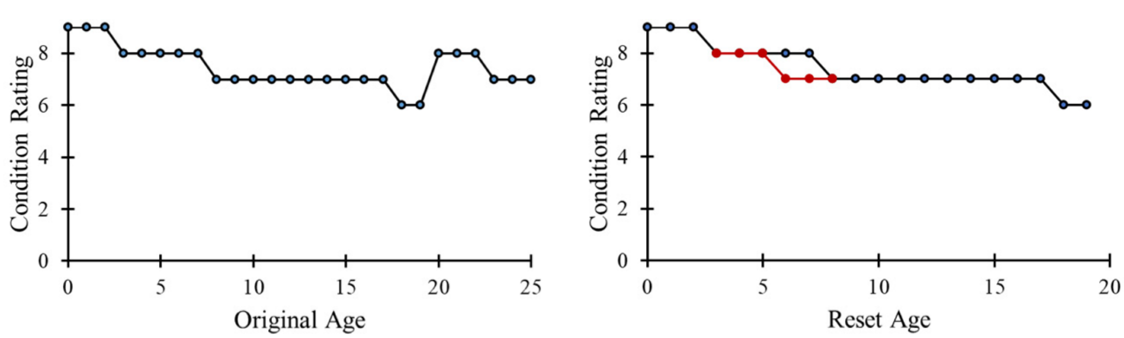

Figure 3. Condition Ratings before (left) and after Age Reset (right)

Data Filtering

After data removal and bridge age reset, the data points pertaining to bridge condition ratings were examined for data filtering at the project level (individual bridges) to ensure more dependable data analysis. The project-level data filtering removed approximately 15% of data points at each condition rating, assuming them as outliers. These data points as outliers often represented the cases where a bridge sustained the same condition rating for an unusually extended period or displayed lower condition ratings at an unusually early age (10 years or less). The 15% filtering was adaptable based on the condition ratings. Specifically, for a higher rating of 9 or 8, a one-tailed 15% filter to the oldest age data points was applied. For other condition ratings, a two-tailed 15% filter (7.5 each) was used, accounting for the youngest and oldest age data points.

One-Tailed 15% Filtering for a Higher Rating of 9 or 8: Bridges rated 9 or 8 underwent a one-tailed 15% filtering approach. This filtering removed the condition data for years beyond 85% of the total years where the bridges maintained a rating of 9 or 8.

Two-Tailed 15% Filter for Other Condition Ratings: Bridges with ratings of 7 or below were subjected to a two-tailed 15% filtering approach. This filtering removed condition data for years prior to 7.5% and beyond 92.5% of the total years where the bridges maintained the same rating.

3.3. Deterioration Model Development

Developing deterioration models for the selected superstructure types employed a regression nonlinear optimization (RNO) model, formulated as follows (Jiang et al., 1998; Ranjith et al., 2013):

$$min \sum_{t=0}^{T} |Y(t) - E(t, P)| \qquad \text{Eq. 1}$$

where, T = a total analysis period; $Y(t)$ = expected condition rating at age t from a regression model; and $E(t, P)$ = expected condition rating at age t from a transition probability matrix P.

The regression model $Y(t)$ is derived from a nonlinear line that best fits the actual condition ratings of bridges used for data analysis. For example, Figure 4 shows the best-fit regression model, which is the third-order polynomial equation for the data points entered and returns the expected condition rating at age t ($0 < t \leq 70$).

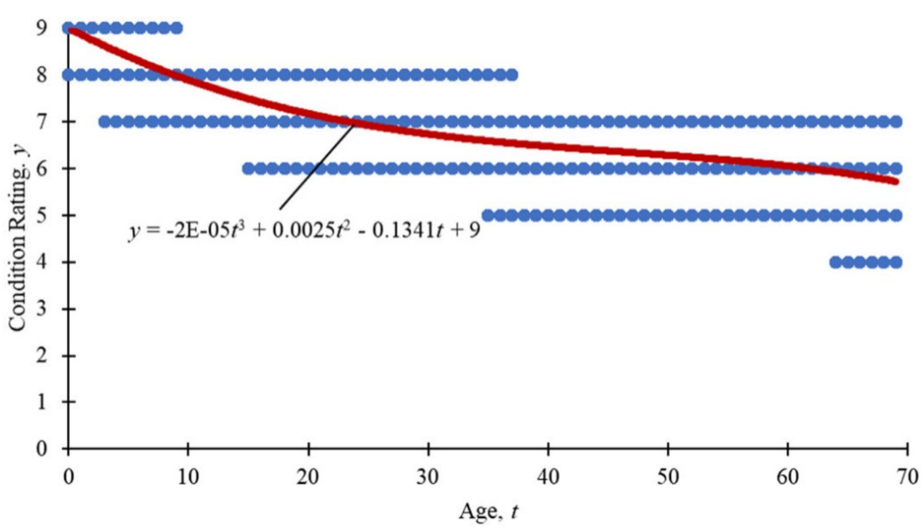

Figure 4. Regression Analysis for Data Points of Condition Ratings

A Markovian transition probability matrix is the most widely used stochastic process for preparing deterioration models of bridge structures. Each transition probability from condition rating i to j (i, j = 9, 8, 7, …, 1; $i \geq j$) in the matrix is represented as p_{ij}. When i and j are equal, p_{ij} indicates the probability of staying in the same condition rating. Figure 5 illustrates the transition probabilities for the three condition ratings, 9, 8, and 7.

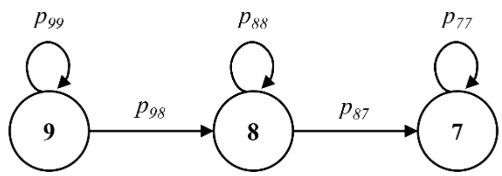

Figure 5. Transition Probabilities in a Markov Chain

In the context of bridge deterioration, it describes the likelihood of a bridge component moving from one condition rating to another on a scale of 0 to 9, with 9 representing near-perfect condition according to the FHWA rating system (Dunker and Rabbat 1990; Markow and Hyman 2009). This matrix represents the probability of transferring from one condition rating to another in a year. A transition probability matrix P is :

$$P = \begin{bmatrix} p_{99} & p_{98} & 0 & \cdots & 0 & 0 \\ 0 & p_{88} & p_{87} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & p_{22} & p_{21} \\ 0 & 0 & 0 & \cdots & 0 & p_{11} \end{bmatrix} \qquad \text{Eq. 2}$$

The sum of the transition probabilities at each row of the matrix should be one. Using two entries in each matrix row assumes that a condition rating would not drop by more than one condition rating between two consecutive inspections. The determination of the transition probabilities in the matrix is completed by minimizing the sum of the absolute differences between $Y(t)$ and $E(t, P)$ in Eq. 1.

The best-fit regression model for each superstructure design was identified using Python scripts (see Appendix 7.3), taking the stages as follows:

- Extract 'Reset Age' and 'Superstructure Summary' values from the filtered dataset.
- Fit the data to get the best polynomial regression model by employing the 'polyfit' function.
- Generate evenly spaced 'Reset Age' values (say, 0,1,2,3…) to construct a continuous deterioration curve.
- Calculate corresponding 'Superstructure Summary' values using the 'polyval' function.
- Calculate the R-squared value to assess the goodness of fit.
- Derive the explicit equation of the fitted polynomial for further analysis.
- Visualize the resulting curve alongside the original data points.

After determining each polynomial regression model for the selected superstructure designs, the Solver in Microsoft Excel was utilized to optimize the transition probabilities for each design.

3.4. Comparative Analysis of Deterioration Curves

A comparative analysis was performed to examine the characteristics of the deterioration curves developed for each selected superstructure type. This analysis focused on the following aspects:

- Similarities and differences of deterioration curves, segmented into different age groups (e.g., ages 0-5, 6-10, 11-15, …), for each maintenance responsibility (e.g., state, county, and city/municipality)

- Average annual degradation rates in the condition rating over an age range (e.g., 5-year span) among the selected superstructure types

The comparative analysis also considered the data strength, which refers to the relative number of data points used for data analysis at each age. This consideration helped compare the characteristics of deterioration curves. A deterioration curve exhibiting greater data strength, characterized by a larger number of data points, offers a more dependable representation of the prevalent deterioration pattern than a curve with a smaller number of data points. This ensures that the observed similarities and differences between the curves are not merely the result of random fluctuations but reflect actual deterioration patterns.

4. RESEARCH FINDINGS AND CONCLUSIONS

Deterioration curves for each selected superstructure and material type are provided for bridges maintained by the state, counties, and cities/municipalities in Figures 6, 7, and 8, respectively. The *y*-axis is the condition rating (CR), and the x-axis is the superstructure age. The deterioration curves are depicted using distinct line types and thicknesses according to the data strength. The data strength percentages in the table indicate the percentage of bridges in a category that falls within the age range indicated. For example, the dashed section of the line includes bridges with ages in the upper 20% of the dataset.

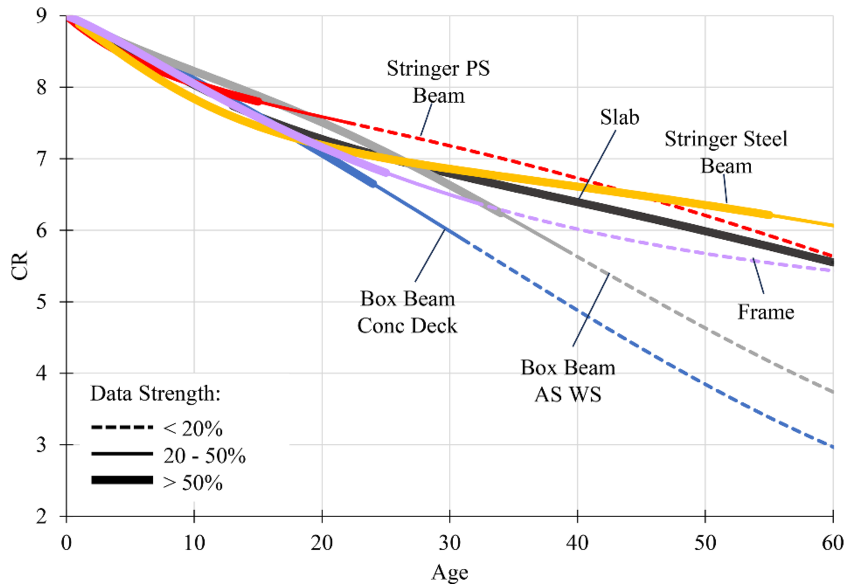

Figure 6. Deterioration Curves of State Bridge Structures

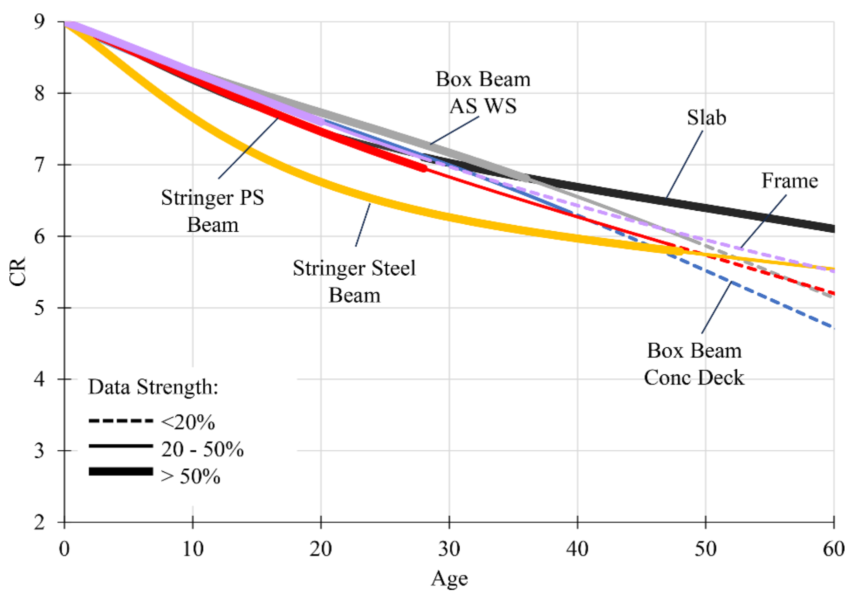

Figure 7. Deterioration Curves of County Bridge Structures

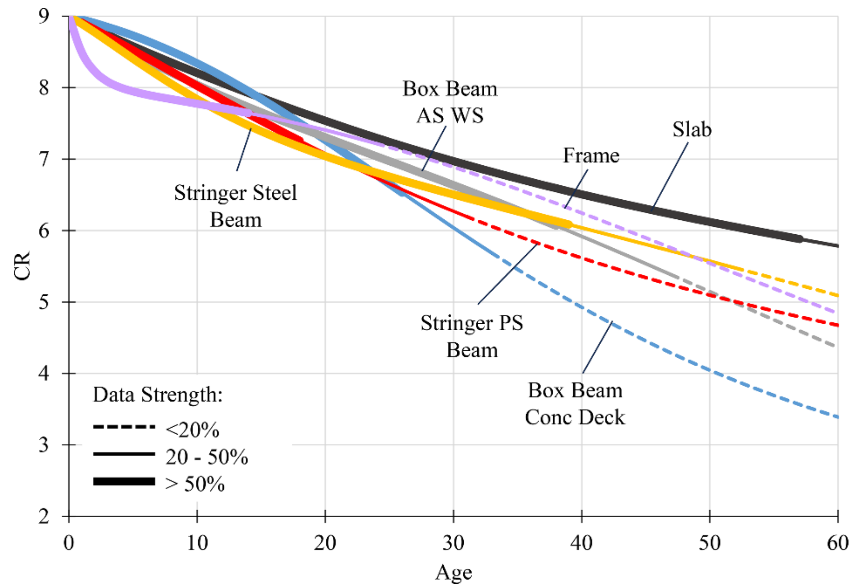

Figure 8. Deterioration Curves of City/Municipality Bridge Structures

4.1. Deterioration Curves of State-Owned Bridge Superstructures

A comparison of deterioration curves for six bridge types under the ownership of the state DOT (see Figure 6) indicates that the stringer beam and slab designs exhibit superior durability over an extended time, as evidenced by their higher CR values. This is followed by the frame and box beam designs. The graph shows that CRs of box beam bridges decrease more rapidly than other structure types after 20 to 25 years of service. The data strengths indicate that slab and stringer steel beam superstructures are prevalent in the state system and have remained active for many years. Over the past fifteen years, slab design for bridge superstructures has become less prevalent (refer to Figure 67 in the Appendix). Stringer PS beam bridges have the shortest application history by the state DOT. The average annual degradation on the *y*-axis in Figure 9 indicates numerical variations in the average condition ratings over a five-year range. Box beam bridges maintain a relatively constant degradation rate relative to slab, stringer, and frame bridges, which exhibit a slowing degradation rate over time.

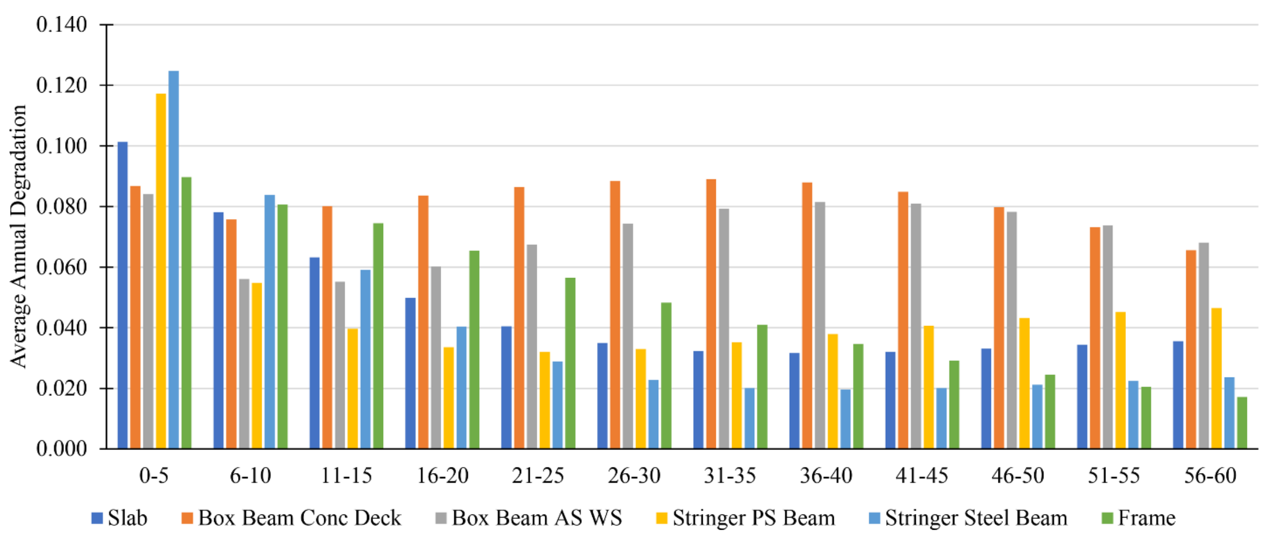

Figure 9. Average Annual Degradation Rates over 5-year Spans, State DOT

4.2. Deterioration Curves of County-Owned Bridge Superstructures

A comparative analysis of county-owned bridge superstructures in Figure 7 finds that, except for stringer steel beams, the early deterioration patterns for all designs are comparable until the age of around 30 years, at which point deterioration variations among design types become apparent. The CRs of slab superstructures decrease more slowly and gradually with age. Initially, box beam superstructures exhibit similar CRs to other designs; however, as the superstructures get older, their CRs undergo a substantial decline. The deterioration pattern of county stringer steel beam superstructures resembles those of state and city/municipality superstructures with the same design. However, this design has a more pronounced decrease in the early-age deterioration rates, suggesting the need for further investigation into potential influences, such as initial construction quality. The data strength information in Figure 10 confirms that the degradation of box beam superstructures remains more constant throughout time, similar to what was observed in Figure 9.

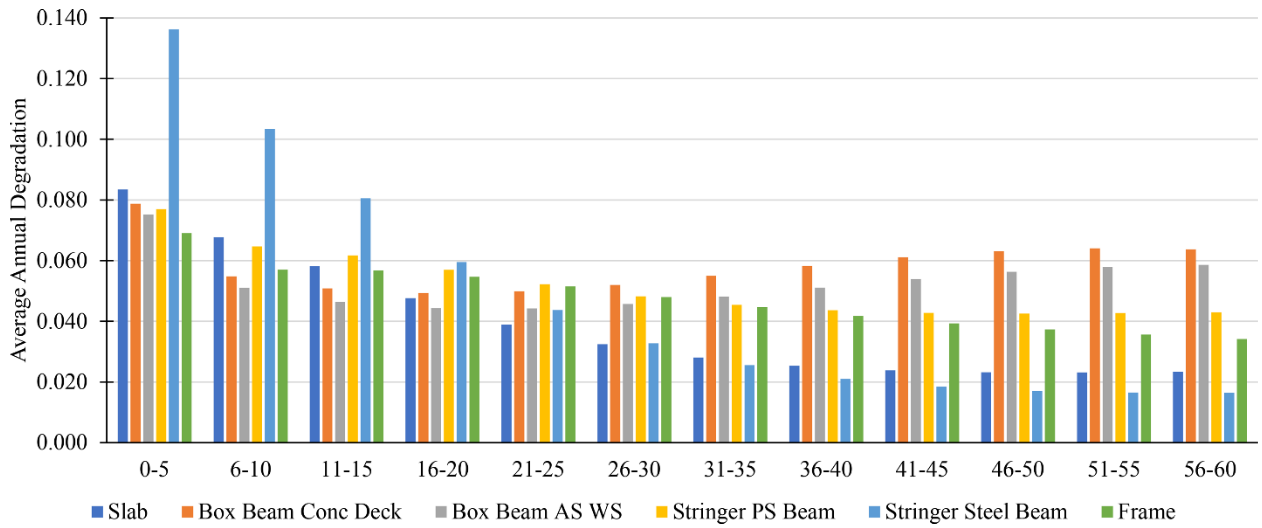

Figure 10. Average Annual Degradation Rates over 5-year Spans, County

4.3. Deterioration Curves of City/Municipality-Owned Bridge Superstructures

An examination of the deterioration curves of city/municipality-owned bridge superstructures in Figure 8 indicates that slab bridges demonstrated higher durability than other superstructure types over the 60-year analysis period. The comparative analysis finds an exceptionally early deterioration pattern for the frame design. The correlation between construction quality and initial bridge performance is widely acknowledged (Uddin et al., 2013). The data strengths indicate that the slab, stringer steel beam, and box beam with AS WS designs have been widely utilized by city and municipal transportation authorities for an extended period, while the application of the slab design type has decreased in comparison to other design types over the past three years. The results of the average 5-year degradation rates in Figure 11 are comparable to those for DOT and county superstructures in Figure 9 and Figure 10, respectively. There is a noticeable decline in the degradation rate of frame superstructures in the 0-5 age range compared to other design options.

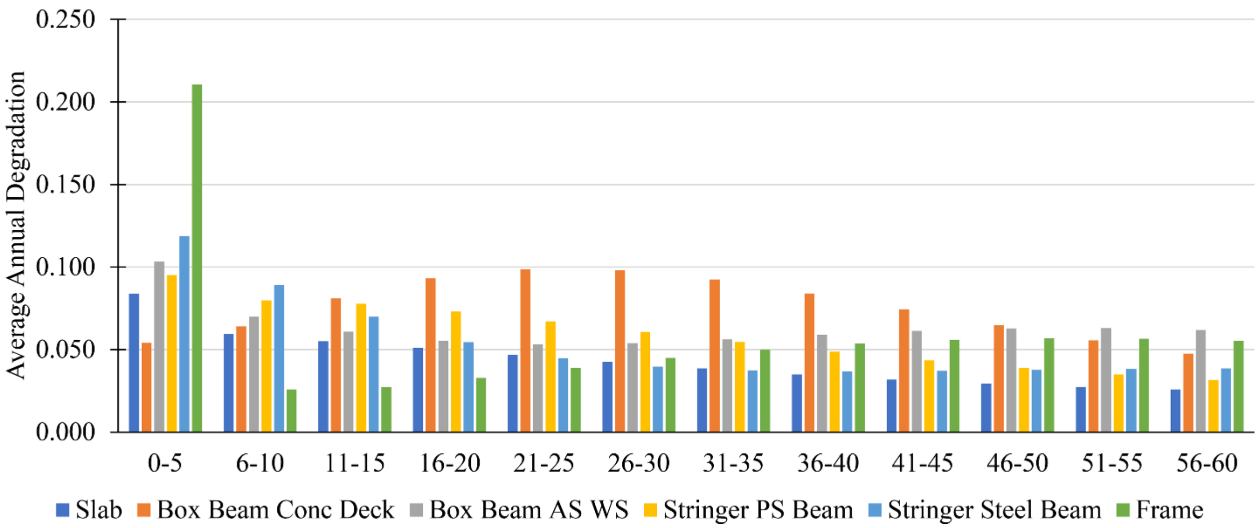

Figure 11. Average Annual Degradation Rates over 5-year Spans, City/Municipality

4.4. Summary Note

This section focuses on a high-level discussion of the research results and conclusions. Complete details and findings, including deterioration curves based on polynomial regression and RNO models, deterioration curve comparison among the selected primary superstructure designs, and average condition ratings over 5-year age ranges, are presented in Appendix 7.4.

The minimum condition rating applied to the deterioration curves was 3, considering the availability of data. In contrast, the transition matrices utilized a range of condition ratings from 9 to 1 to generate deterioration curves closer to the polynomial models through the optimal process in Eq. 1.

The TAC of this study expressed some concern regarding the deterioration curves for box beams with an asphalt wearing surface (non-composite) and box beams with a concrete deck (composite). The deterioration curve for box beams with a concrete deck was expected to show slower degradation than that for box beams with an asphalt wearing surface, but the research results showed the opposite. The research team did a preliminary assessment to identify possible reasons for this result and found that parameters such as ADT and deck area may be contributing factors (see Appendix 7.4.5). Additionally, general maintenance or repair activities that do not change the inspection ratings may be potential factors in the deterioration curves. This suggests that there could be a contrary outcome if the data is reexamined using sets of bridges with similar ranges of ADT and other variables.

5. RECOMMENDATIONS FOR IMPLEMENTATION

5.1. Recommendations for Implementation

Several recommendations for the implementation based on the research findings are:

Feedback through Stakeholder Engagement: Stakeholder engagement involves disseminating research findings to professional and government agencies at the state, county, and city/municipality levels through presentations and publications to receive feedback on the research findings and make this research actionable and effective for the state of Ohio.

Investigation on the Necessity for Design Standard Revisions or Improvements: Gaining insight into the deterioration characteristics of the selected superstructures over time enables the state to assess the existing bridge design standards and ascertain the necessity for revisions or improvements.

Integration into Bridge Management Planning: By integrating the developed deterioration curves into bridge management planning processes, it is possible to more effectively establish long-term strategic decision-making for maintenance, repair, rehabilitation, and reconstruction. Consequently, the allocation of funds and resources for bridge management can be conducted with greater transparency.

5.2. Steps Needed to Implement

Implementing the research findings involves a multifaceted approach as follows:

Understanding the Research Findings: Ensure a clear understanding of research findings and possible implications through stakeholder engagement.

Developing Implementation Plans in Details: Outline specific activities required, responsibilities, timelines, staffing and resource planning, and budgets.

Capacity Building: Develop a training program for responsible parties (e.g., individuals or teams) to obtain the necessary skills and allocate resources.

Monitoring and Evaluation: Monitor the implementation process and evaluate the efficiency and applicability of the research findings and recommendations, engaging with the research team involved in the study.

Feedback Loop: Establish a feedback process to compile inputs from those directly involved in and affected by the implementation. Based on feedback, the recommendations and implementation plans can be refined, and future research can be proposed to refine the research findings and tackle new challenges.

Documentation: Maintain a comprehensive record of all the implementation steps for future reference and transparent communication.

5.3. Expected Benefits from Implementation

The research findings will directly benefit ODOT and local agencies. The research will deepen understanding of the lifetime deterioration patterns of common bridge superstructure types and time-variant characteristics over their service life. The advanced knowledge will benefit ODOT and local agencies in better judging appropriate superstructure-type selections made by in-house and consultant design personnel and establishing bridge management strategies from short- and long-term perspectives. Clarifying the acceleration or decrease in annual degradation rates of the selected superstructure designs can enhance risk management and safety planning for the state's bridge network by flexibly determining the priority of maintenance, repair, or rehabilitation needs over time. Also, the research findings will aid in addressing the federal requirement to forecast deterioration for all national highway system (NHS) bridge assets.

The results from this research can greatly benefit a variety of users. Designers (in-house or design agency consultants) can utilize the results of this research to prepare better life-cycle costs for bridge type selections being considered for new bridges or bridge replacements. The research results, particularly the comparative analysis, will provide a fundamental reference for subsequent investigations into the impact of internal and external factors, such as design, functional class, material, operation, and environment, on the patterns of superstructure deterioration.

5.4. Potential Risks and Obstacles to Implementation

The potential risks and obstacles to implementing the research findings on recommendations could be:

Data Integration and Management: Incorporating deterioration models, which are novel insights, into existing bridge management systems necessitates the implementation of technical modifications and the effective management of consistency concerns.

Changing Stakeholder Perception: It is vital to overcome antagonism from stakeholders accustomed to conventional practices.

Resource and Funding Availability: Acquiring supplementary funding and resources is critical to implementing newly discovered research findings regarding recommendations while adhering to financial limitations and competition.

Model Accuracy and Adaptability: Ensuring the accuracy and adaptability of deterioration models requires regular updates based on additional inspection data and emerging trends in applying advanced data analysis techniques, such as artificial intelligence.

Interagency Collaboration: Efficient collaboration and knowledge sharing are crucial to disseminating best practices and research updates across various agencies.

5.5. Strategies to Overcome Potential Risks and Obstacles

Multiple strategic approaches may be contemplated to effectively overcome the presented potential risks and obstacles to implement the research findings:

Data Integration and Management: Adopt phased integration with a user-centered design for easy adoption and compatibility integration and management.

Changing Stakeholder Perception: Utilize educational workshops, pilot projects, and performance metrics to showcase the benefits of implementing the research findings while ensuring the transparency of these benefits.

Resource and Funding Availability: Conduct cost-benefit analyses and implement recommendations in phased stages aligned with budget cycles.

Model Accuracy and Adaptability: Ensure continuous monitoring, integrate data analytics for model updates, and employ scenario planning for future uncertainties.

Interagency Collaboration: Create collaboration platforms, organize joint training sessions, and develop standard protocols for broader adoption and effective knowledge exchange.

5.6. Potential Users and Other Organizations That May Be Affected

The research findings that provide valuable insights into the deterioration patterns of the primary bridge superstructure designs over time might affect the potential users and other organizations as follows:

- State DOTs and transportation agencies responsible for bridge management might find the findings useful and be inspired to conduct comparable investigations.
- Organizations specializing in bridge design may utilize these findings to substantiate their design

selection recommendations.

- Consulting firms involved in regular bridge inspections and management activities can use the insights to improve task prioritization, resource allocation, and the development of proactive management plans.
- Researchers in bridge engineering and infrastructure asset management could utilize the research findings to develop more detailed, comprehensive future research.
- The general public and communities interested in gaining knowledge about the safety/risk of bridges in their regions and voicing their opinions regarding the selection of a proposed bridge design may use the research findings.

5.7. Suggested Time Frame for Implementation and Estimated Costs

Implementing the research findings can be envisioned as a phased process, unfolding over different timeframes based on each recommendation's complexity and resource requirements. Here's a tentative roadmap:

Short-Term (1-2 Years): Elicitation of feedback through stakeholder engagement; Investigation of the need for revisions or improvements to design standards; Commencement of a capacity building initiative. The estimated cost as an internal process would be less than $20,000 in total.

Mid-Term (3-5 Years): Development of detailed implementation plans for a pilot study; Formulation of monitoring and evaluation procedures. The estimated cost, involving one research project, would range from $50,000 to $100,000 annually.

Long-Term (6-10 Years): Extension of the pilot study to a scale-up integration into bridge management based on monitoring and evaluation results; Continuous updates on the deterioration curves based on additional inspection data. The estimated cost, involving the update on the existing bridge management system and possibly one research project, would range from $100,000 to $200,000 annually.

Note: The final cost estimation should be prepared in conjunction with the TAC, depending on the finalized recommendations and time frame for implementation.

5.8. Recommendations on How to Evaluate the Ongoing Performance of the Implemented Result

The recommendations to evaluate the ongoing performance of the implemented result focusing on a pilot study and scale-up integration are:

- Increased transparency among communities, design consulting firms, state and local administrations, and the general public in the design selection process for bridge superstructures.
- Enhanced resource allocation efficiency and cost-saving for bridge management practice while observing the overall condition ratings of bridge superstructure network sustained or improved.
- Improved efficiency of the bridge management system that integrates the research findings relative to the original system.

6. BIBLIOGRAPHY

American Road & Transportation Builders Association. (2022). *Ohio: 2022 bridge profile*. Retrieved February 10, 2022, from https://artbabridgereport.org/reports/state/OH.pdf.

American Society of Civil Engineers. (2021). *2021 Report card for Ohio's infrastructure*. Retrieved February 12, 2022, from https://infrastructurereportcard.org/wp-content/uploads/2021/07/FullReport-OH_2021_smaller.pdf.

Caltrans Division of Research, Innovation and System Information. (2020). *Bridge deterioration models and rates*. Retrieved February 21, 2022, from https://dot.ca.gov/-/media/dot-media/programs/research-innovation-system-information/documents/preliminary-investigations/pi-0274-a11y.pdf.

Cavalline, T. L., Whelan, M. J., Tempest, B. Q., Goyal, R., & Ramsey, J. D. (2015). *Determination of bridge deterioration models and bridge user costs for the NCDOT bridge management system*. FHWA/NC/2014- 07, North Carolina DOT, Raleigh, NC.

Chang, M., & Maguire, M. (2016). *Developing deterioration models for Wyoming bridges*. FHWA-WY- 16/09F, Wyoming. Dept. of Transportation, Cheyenne, WY.

DeLisle, R. R., Sullo, P., and Grivas, D. A., (2004) "Element-Level Bridge Deterioration Modeling Using Condition Durations", *83rd TRB Annual Meeting*, January, Washington, D.C.

Dunker, K. F., and Rabbat, B. G. (1990). Performance of highway bridges. *Concrete Int.*, 12(8), 40–43.

Federal Highway Administration (FHWA). Recording and Coding Guide for the structure Inventory and Appraisal of the Nation's Bridges. Report No.FHWA-PD-96-001.Office of Engineering Bridge Division, Washington, D.C., 1995.

Fu, G. (2021). *Evaluation of Illinois Bridge Deterioration Models*. FHWA-ICT-21-024, Illinois Center for Transportation, Springfield, IL.

Hatami, A., & Morcous, G. (2011). *Developing deterioration models for Nebraska bridges*. SPR-P1 (11) M302, Nebraska Transportation Center, Lincoln, NE.

Huang, Y. H. (2010). "Artificial neural network model of bridge deterioration." *Journal of Performance of Constructed Facilities*, 24(6), 597-602.

Hunt, V. J., Helmicki, A. J., & Swanson, J. A. (2011). *Development of degradation rates for various bridge types in the state of Ohio.* FHWA/OH-2011/9, Ohio. Dept. of Transportation, Columbus, OH.

Ilbeigi, M. E. M. M., & Ebrahimi Meimand, M. (2020). "Statistical forecasting of bridge deterioration conditions." *Journal of Performance of Constructed Facilities*, 34(1), 04019104.

Markow, M. J., & Hyman, W. A. (2009). *Bridge management systems for transportation agency decision making* (Vol. 397). Transportation Research Board.

MDT. (n.a.) *Development of deterioration curves for bridge elements in Montana*. Montana DOT, St. Billings, MT.

Mishalani, R. G. and Madanat, S. M. (2002). "Computation of Infrastructure Transition Probabilities Using Stochastic Duration Models." *Journal of Infrastructure Systems*, 8(4), pp. 139–148.

Morcous, G. (2006). "Performance prediction of bridge deck systems using Markov chains." *Journal of performance of Constructed Facilities*, 20(2), 146-155.

Morcous, G., Rivard, H., and Hanna, A. M. (2002). "Case-Based Reasoning System for Modeling Infrastructure Deterioration." *Journal of Computing in Civil Engineering*, 16(2), 104-114.

Moomen, M., Qiao, Y., Agbelie, B. R., Labi, S., & Sinha, K. C. (2016). *Bridge deterioration models to support*

Indiana's bridge management system. FHWA/IN/JTRP-2016/03, INDOT, Indianapolis, IN.

Nelson, S. L., & Olson and Nesvold Engineers, P. S. C. (2014). *Deterioration rates of Minnesota concrete bridge decks*. MN/RC 2014-40, Minnesota DOT, Research Services & Library, St. Paul, MN.

Nickless, K. (2017). *Investigation of mechanistic deterioration modeling for bridge design and management*. MS thesis, Colorado State University, Fort Collins, CO.

ODOT. (n.a.). *AssetWise*. Retrieved February 11, 2022, from https://www.transportation.ohio.gov/working/data-tools/resources/assetwise-inspection-system

O'Leary, M. and Walsh, J. (2018). *Bridge element deterioration of concrete substructures.* WA-RD 893.1, Washington State DOT, Olympia, WA.

Ranjith, S., Setunge, S., Gravina, R., & Venkatesan, S. (2013). Deterioration prediction of timber bridge elements using the Markov chain. Journal of Performance of Constructed Facilities, 27(3), 319-325.

Sobanjo, J. O., & Thompson, P. D. (2011). *Enhancement of the FDOT's project level and network level bridge management analysis tools*. FDOT, Tallahassee, FL.

Transportation Pooled Fund. (2022). *Bridge element deterioration for Midwest states*. Retrieved February 21, 2022, from https://www.pooledfund.org/Details/Study/655.

Uddin, W., Hudson, W. R., and Haas, R. (2013). Public Infrastructure Asset Management, 2nd edition, McGraw-Hill Education, New York.

Upadhya, P. R., Das, M. S., & Das, B. B. (2021). Multi-criteria Decision-making Approach for Selecting a Bridge Superstructure Construction Method. In Recent Trends in Civil Engineering: Select Proceedings of TMSF 2019 (pp. 497-508). Springer Singapore.

Winn, E. K., & Burgueño, R. (2013). *Development and validation of deterioration models for concrete bridge decks-phase 1: artificial intelligence models and bridge management system.* RC-1587a, Michigan. Dept. of Transportation, Lansing, MI.

Yoon, Y. (2012). *Planning of optimal rehabilitation strategies for infrastructure using Time Float and Multiyear Prioritization Approach*. Ph.D. Dissertation, Purdue University, West Lafayette, IN.

Yoon, Y., & Hastak, M. (2016). "Condition improvement measurement using the condition evaluation criteria of concrete bridge decks." *Journal of Transportation Engineering*, 142(11), 04016054.

7. APPENDIX

7.1. A Complete Literature Review

7.1.1. State DOTs Research for Bridge Deterioration Curves

As ODOT seeks to develop in-house bridge deterioration curves through this research, the ongoing and completed work for in-house deterioration models was the primary focus. The preliminary investigation conducted by California DOT (Caltrans) in 2020 identified several state DOTs (e.g., CO, FL, IL, IN, MI, KS, OR, VA, WI, and WY) known for using in-house deterioration models (Caltrans, 2020). A brief literature review of these states found the final reports for some of these state DOTs, adding other state DOTs for in-house deterioration models as well. Each research is recognized by its title, sponsoring agency, and year completed.

<u>TPF-5 (432) Bridge Element Deterioration for Midwest States; Midwest State DOTs; Transportation Pooled Fund (2022)</u>

One of the main objectives of this research led by Wisconsin DOT (WisDOT) is to develop component- and element-level deterioration models for Midwest DOT bridges. This research utilizes historic bridge data retrieved from the twelve partner states in the Midwest region, reflecting the regional environment (winter/summer), operations practices (application of deicing chemicals), maintenance practices, and design/construction details. The Markov-based approach was suggested to develop base deterioration models for bridge components and elements. As developing deterioration curves in this research is based on the data from all Midwest states, ODOT can utilize the expected end-products as a reference.

<u>Development of Degradation Rates for Various Bridge Types in the State of Ohio; Hunt et al. (2011)</u>

This research was conducted to understand the degradation rates of the ODOT's OPIs (e.g., GA, FC, WC, and PCS) by the Markov chain-based statistical analysis for the historical bridge inspection data obtained from the state Bridge Management System (BMS). The dynamic degradation rates for the four OPIs were presented using the moving window approach that considered the latest two- or five-year BMS data set for a target year for degradation rates (i.e., the BMS data generated from 1996 to 2000 for 2001, from 1997 to 2001 for 2002, and so on). The degradation rates in this research represent the transition rates of the four OPI measures from almost deficiency (e.g., GA = 5) to current deficiency (e.g., GA < 5) between two consecutive inspection years, which indicates that the OPI degradation rates are not lifetime deterioration curves.

<u>Bridge Deterioration Models to Support Indiana's Bridge Management System; Moomen et al. (2016)</u>

This research developed families of deterioration curves for bridge components (e.g., deck, superstructure, and substructure) for Indiana's state highway bridge. The NBI condition ratings were analyzed for various independent variables using the regression and binary probit modeling techniques. The regression analysis categorized NHS and non-NHS bridges by influential factors, such as administrative region, functional class, and superstructure material and design type. As a result, several regression-based deterioration models were developed: six for the deck, forty-two for the superstructure, and nine for the substructure. On the other hand, the research presented three probabilistic deterioration models for deck, superstructure, and substructure. The research recommended incorporating some grouping criteria, such as the bridge design type, as explanatory variables to reduce the number of deterioration curves in the regression model.

<u>Enhancement of the FDOT's Project Level and Network Level Bridge Management Analysis Tools; Sobanjo and Thompson (2011)</u>

Through this research, Florida DOT (FDOT) developed improved deterioration models for the Pontis Bridge Management System – currently, AASHTOWare BrM – by analyzing the FDOT's element-level bridge inspection data. The prediction of deterioration curves was based on the Markovian transition probability matrices. The research estimated a separate Markovian transition probability matrix for each of the 151 bridge elements in four different environments: benign, low, moderate, and severe. The final models were collapsed into

72 element types by grouping the individual element/environmental models.

Investigation of Mechanistic Deterioration Modeling for Bridge Design and Management; Nickless (2017)

This research investigated the applicability of the mechanistic deterioration modeling approach to predict bridge deterioration, considering the management of existing bridges and the design of new bridges in Colorado. For the applicability investigation, a mechanistic deterioration curve for reinforced concrete bridge decks was developed, which was then evaluated for the current practices in Colorado DOT (CDOT), such as epoxy-coated rebar, waterproofing membranes, and asphalt wearing surfaces. Based on the results, this research suggested that mechanistic models could be used to supplement the Markov chain deterioration models developed by the previous CDOT-sponsored research, Deterioration and Cost Information for Bridge Management, for the bridge elements in Colorado.

Development and Validation of Deterioration Models for Concrete Bridge Decks; Winn and Burgueño (2013)

When this research was conducted, Michigan DOT (MDOT) used AASHTO Points BMS integrated with in-house bridge component-level deterioration models, such as the Bridge Condition Forecasting System (BCFS). This research applied two types of artificial neural networks (ANNs), multilayer perceptions and ensembles of neural networks, to predict the decreasing condition ratings of concrete bridge decks. The prediction powers of these two ANN-based models outperformed the conventional Markov models. Also, this research investigated the influential factors causing condition changes for the developed ANN models, which was expected to allow MDOT to understand concrete bridge deck deterioration at the project and network levels.

Development of Deterioration Curves for Bridge Elements in Montana; MDT (n.a.)

Montana DOT (MDT) solicited this research in progress to develop deterioration curves for bridge elements in Montana. The selected research team plans to use Markov models to create deterioration curves for bridge elements in groups and conduct incremental analysis for different explanatory variables, such as traffic characteristics, bridge type, and environmental conditions for any changes in group deterioration curves.

Deterioration Rates of Minnesota Concrete Bridge Decks; Nelson et al. (2014)

This research analyzed the NBI condition code data for 2,601 bridges with concrete decks. The majority of the concrete deck bridges were supported by girder-type prestressed concrete or continuous steel superstructures. The concrete bridge decks were categorized based on superstructure types, reinforcement types (e.g., black bars, epoxy-coated top bars, and all epoxy-coated bars), the presence of concrete overlay, average daily traffic, and the presence of 3-inch cover to the top mat of reinforcement. The analysis approach to determining the deterioration rates of concrete bridge decks in each category was estimating the length of time that bridge decks stayed or dropped at NBI condition codes.

Developing Deterioration Models for Wyoming Bridges; Chang and Maguire (2016)

This research developed deterioration models for Wyoming bridges by analyzing the NBI data for both stochastic and deterministic models. To determine explanatory variables that significantly affected the deterioration behavior of bridge components, a well-known penalized regression, the Least Absolute Shrinkage and Selection Operator (LASSO), was used to eliminate human bias in explanatory variable selection. Bridges were grouped by the explanatory variables selected. Then, the deterministic deterioration models for the bridge groups were developed by using a curve-fitting method for the mean of bridge ages for each condition rating. The stochastic models applied the Markov chain to estimate the transition probability matrix by a percentage prediction method. The same bridge groups as the deterministic models were considered in the stochastic models.

Evaluation of Illinois Bridge Deterioration Models; Fu (2021)

This research was designed to review the reliability of the state's current deterministic deterioration models for the bridge components, including deck, superstructure, substructure, culvert, and deck beam. The deterministic models use transition time (in the number of years) between every two NBI condition ratings. This research found that the current deterioration models are inadequate in forecasting condition ratings, recommending future research to apply probabilistic transition times (e.g., Weibull distribution) for the current deterioration models as a long-term solution.

Developing Deterioration Models for Nebraska Bridges; Hatami and Morcous (2011)

This research aimed to develop deterioration models for Nebraska bridges using the NBI condition ratings of bridge components. The bridges were classified based on the values of factors such as bridge design, construction, geographical location and environment, and traffic volume for homogenous and consistent data analysis. Developing component-level deterioration models applied a deterministic approach of curve-fitting methods. This research also presented state-based stochastic deterioration models to the AASHTO Points that the Nebraska Department of Roads (NDOR) adopted to support their BMS when this research was implemented.

Bridge Element Deterioration of Concrete Substructures; O'Leary and Wals (2018)

This research presented probabilistic deterioration models for concrete bridge substructure elements in the state of Washington. The deterioration models for concrete substructures included elements such as concrete pile/column and concrete submerged pile/column located in Eastern and Western Washington climates. The probabilistic models were based on the average age of a concrete substructure element at the transition from one condition state to the next.

Determination of Bridge Deterioration Models and Bridge User Costs for the NCDOT Bridge Management System; Cavalline et al. (2015)

One of the research objectives was to provide North Carolina DOT (NCDOT) with revised, updated deterioration models for use in the BMS software. Time-based deterministic deterioration models updating NCDOT's 2002 deterioration models were developed for bridge components and culverts grouped by material type, design type, geographic location, or average daily traffic, depending on the component type. The time-based deterministic models computed the expected duration spent in each NBI condition rating. Also, this research presented probabilistic deterioration models using transition probability matrices to facilitate NCDOT's transition from the deterministic deterioration models to the preferred probabilistic models.

7.1.2. Factors for Deterioration Curve Analysis

The selection of explanatory variables (thereafter, factors following the RFP) that have actual effects on bridge deterioration is essential to developing more reliable bridge deterioration curves. This section presents the factors proven in previous studies through various statistical analyses to affect bridge deterioration. The summary of these factors is listed in Table 4, which also includes the factors from the survey results conducted by Caltrans (Caltrans, 2020). The survey investigated the factors accommodated by the deterioration models of state DOTs. The number of factors representing the state count out of 21 responses to the survey. The research team classified these factors by their similar features for possible combinations in the data analysis of this research. Also, in addition to the bridge structure types, the categorical factors in Table 4 (e.g., rebar protection, use of deck overlays, and climatic conditions) will be considered to group bridges into families, which is necessary to reduce the number of deterioration models (Moomen et al., 2016). The research team will use only the factors statistically significant to Ohio bridge condition ratings, which will be found during the data analysis, to develop deterioration curves. The details of each study reviewed are presented in this section.

Table 4. Factors Recommended for Data Analysis

Factor	Classification	NC	WY	WI	IN	MI	OH	Caltrans Survey
Age	Age	Y	Y			Y	Y	17
Age from reconstruction	Age						Y	
Number of spans	Design	Y		Y		Y		2
Bridge roadway width	Design		Y					
Deck area	Design			Y			Y	
Degree of skew	Design			Y				2
Design load	Design		Y	Y		Y		4
Maximum span length	Design	Y	Y				Y	2
Rebar protection	Design					Y		9
Structure length	Design		Y	Y				
Structure type	Design					Y		8
Use of deck overlays	Design							16
Climatic conditions	Environment							11
Environment	Environment			Y				
Freeze-thaw cycles	Environment				Y			
Number of cold days	Environment				Y			
Functional classification	Functional class		Y		Y			11
Structure improvement length	Reconstruction						Y	
Presence of reconstruction	Reconstruction	Y			Y			
Approach surface type	Material							1
Deck material type	Material		Y					
Structural materials	Material						Y	
Superstructure material type	Material				Y			18
Type of wearing surface	Material		Y		Y			12
AADT	Operation				Y	Y		10
ADT	Operation		Y	Y		Y	Y	11
Percent truck traffic	Operation					Y		
Service under a bridge	Operation				Y			
Current bridge condition	Operation						Y	
District	Region			Y				
Geographic region	Region	Y			Y	Y		8

Cavalline et al. (2015) analyzed the hazard ratios of factors for all material-specific bridge components in North Carolina. Compared to the baseline case (HR = 1), the hazard ratio (HR) expresses the risk of failure due to a specific factor. For example, cracked components with an HR of 2 are likely to fail at twice the uncracked components (HR = 1). The factors found to increase component deterioration rates were ***age, bridge designs with multiple spans, the presence of reconstruction, geographic region***, and ***maximum span length***, affecting deterioration rates across all bridge components. The effect of age on the deterioration rates is well-documented in numerous deterioration modeling studies. Multi-span bridge decks necessarily include expansion joints with a higher propensity for failure, affecting the overall deck condition ratings. The presence of reconstruction showed higher deterioration rates than original or rebuilt bridges. The effect of geographic regions on deterioration rates is attributed to the use of deicing salts and freeze-thaw cycles in cold-weather regions. Lastly, maximum span length specifically increased deterioration rates of all deck materials. This study also discussed other factors such as secondary routes, average daily traffic (ADT), and average daily truck traffic (ADTT), interestingly showing either the research results being poles apart or no effect on deterioration rates.

Chang and Maguire (2016) applied LASSO regression to determine the relative importance of candidate factors and used the top five for bridge components in Wyoming as follows:

- ***Year built, type of wearing surface, structure length, functional classification of inventory route,*** and ***ADT*** for decks
- ***Deck material type, year built, bridge roadway width, functional classification of inventory route,*** and ***length of maximum span*** for superstructures
- ***Year built, type of wearing surface, design load, bridge roadway width,*** and ***functional classification of inventory route*** for substructures

The analysis results showed two interesting factors for substructures (e.g., type of wearing surface and bridge roadway width) that the research team seemed to have no statistical relationships. These factors will be reevaluated through statistical analysis in Task-1 of this research.

Huang (2010) conducted the ANOVA analysis to determine the significance of eleven factors on condition states in the study and to develop a deterioration model for concrete bridge decks in Wisconsin. The analysis revealed eight significant factors: ***district, design load, ADT, environment, degree of skew, deck length, deck area,*** and ***the number of spans***. The values of the environmental factor are categorical, such as benign, low, moderate, and severe, depending on the traffic volume and environmental conditions at a bridge.

Moomen et al. (2016) developed bridge deterioration models to support Indiana's bridge management system. The probabilistic modeling results indicated that factors such as functional class, region, freeze-thaw cycles, and rehabilitation status are the most significant factors in transitioning a bridge component to a lower condition state. Specifically, for each bridge component, they found ***ADTT, type of wearing surface***, and ***the number of cold days*** as influential factors for the deck deterioration, ***superstructure material types*** for the superstructure, and ***the service under a bridge (waterway)*** for the substructure.

Winn and Burgueño (2013) utilized correlation analysis and chi-squared hypothesis testing to identify the factors affecting the condition ratings (from 3 to 9) of bridge decks in Michigan. The analysis found factors such as age, structure type, rebar protection, and region statistically significant for bridge decks in the hypothesis testing. Based on these factors, they developed a baseline deterioration model to test additional factors, increasing the predictive power of the model. Finally, they identified more factors such as ***ADT, percent truck traffic, ADTT, number of spans, design load***, and ***approach surface type***.

Ilbeigi and Meimand (2019) used the historical NBI data of the Ohio highway bridges in their study to predict future bridge deterioration conditions. As a part of the study scope, they analyzed factors significant to the ordinal regression-based deterioration models, which included ***age, ADT, deck area, the current condition of the bridge, length of structure improvement, age from reconstruction, structural materials***, and ***the maximum length of the span***. This study suggested one interesting factor, the length of structure improvement, which was not found in any other similar studies. The length of structure improvement represents the extent of reconstruction or major rehabilitation operations. The bridges with these operations have greater deterioration rates than new bridges, indicating that the quality of the operations is likely to be imperfect compared with new construction.

7.1.3. Deterioration Analysis Models

Deterioration models for bridges can be categorized into deterministic, mechanistic, stochastic, and artificial intelligence (AI) models (Morcous et al., 2002; Yoon, 2012; Moomen et al., 2016). Deterministic models assume that the deterioration of a bridge follows a pre-determined pattern in the relationship between the factors affecting bridge conditions. The common methods for the deterministic models include straight-line extrapolation, regression, and curve-fitting. Stochastic models predict the deterioration process of bridges by capturing the uncertainty and randomness of this process in one or more factors. These models can be classified either as state-based or time-based models. The state-based models, known as Markov chains, define deterioration processes as the probabilities of a bridge condition transitioning from one state to another in a discrete-time (Morcous, 2006). Markov chain-based models are the most common deterioration models being used by the bridge management systems of many state DOTs. On the other hand, the time-based models, known as Weibull distribution, characterize the deterioration process in the duration (years) of a bridge remaining at a particular state (or condition rating) (Mishalani and Madanat, 2002; DeLisle et al., 2004). Mechanistic models are based on physical processes causing condition degradation over time. Lastly, AI models take advantage of computer techniques to learn deterioration patterns from the training on historical condition data to develop deterioration models. The different methods for AI models include, for example, artificial neural networks (ANNs), case-based reason (CBR), and expert systems.

Each deterioration model possesses its own merits and inherent limitations. The most common analysis models among them are regression-based and Markov chain models. Table 5 summarizes the characteristics and limitations of these deterioration models with some methods as examples. However, it should be mentioned that there have been numerous studies suggesting various alternatives to overcome these limitations.

Table 5. Characteristics and Limitations of Different Deterioration Models

Model	Method	Characteristic	Limitation
Deterministic	Straight-line extrapolation Regression Curve-fitting	• No need for a large data set • A priori classification of bridges	• Inadequate accounting for uncertainty • Not applicable for censored data (i.e., censoring on condition rating durations)
Stochastic	State-based (Markov chain) Time-based (Weibull Distribution)	• Stochastic nature of deterioration • Computational efficiency for large bridge networks • No need for a large data set • Future conditions relying on the current state • Computational simplicity	• Very dependent on the quality and availability of data • Inaccurate in predicting the deterioration of individual bridges • History independence
Mechanistic		• Mathematical descriptions of the cause-effect on deterioration	• Challenge in capturing the complicated nature of multiple deterioration mechanisms
AI	CBR ANN Expert systems	• Possible to capture the high complexity of the NBI data (e.g., non-linearity, subjectivity, and missing and noisy data)	• Need for a large data set • Complex in interpreting model parameters

7.2. Flowchart for Data Processing

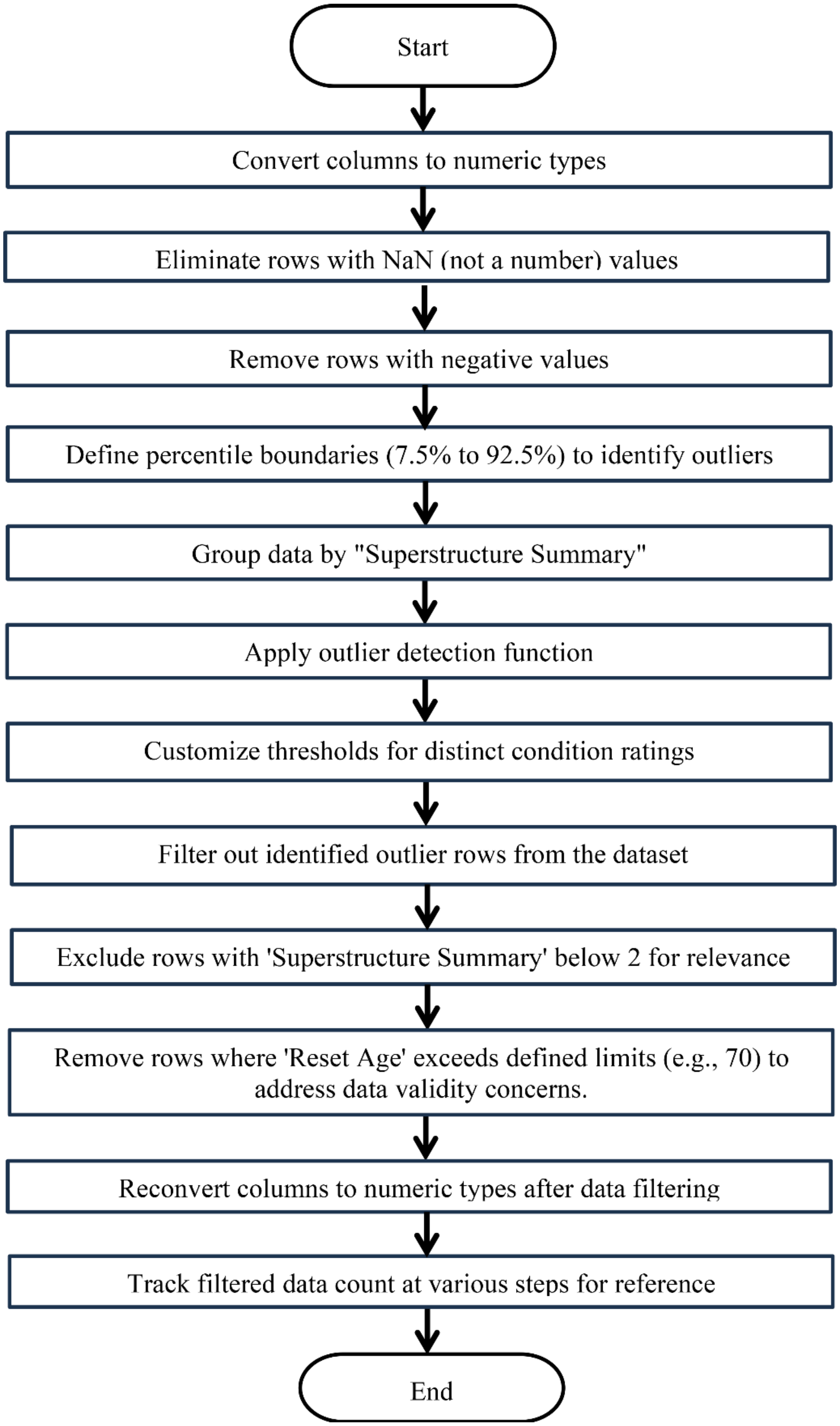

Figure 12. Analytical Function Flowchart for Data Processing in Python Scripts

7.3. Python Scripts used for Data Processing and Polynomial Analysis

Python Script 1: Data Processing and Polynomial Regression Analysis for State_Stringer_PS Beam_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Stringer_PS beam_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
```

```
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Deck Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 5:
        # For 'Deck Summary' 5, set lower bound to 15%
        lower_bound = group.quantile(.15)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 75]

# Count the remaining number of rows after removing outliers and Deck summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)
x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept

    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)
```

```python
# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State> Stringer >PS beam> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 2: Data Processing and Polynomial Regression Analysis for State_Stringer_Steel Beam_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Stringer_Steel_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
```

```
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 12%
        lower_bound = group.quantile(.12)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure Summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)
```

```
# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State> Stringer> Steel beam> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 3: Data Processing and Polynomial Regression Analysis for State_Box Beam_Asphalt Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Box beam_Asphalt WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
   lower_bound = group.quantile(lower_percentile / 100)
```

```
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Deck Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 4:
        # For 'Deck Summary' 4, set lower bound to 10%
        lower_bound = group.quantile(0.10)
    if group.name == 3:
        # For 'Deck Summary' 3, set lower bound to 10%
        lower_bound = group.quantile(0.10)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 70]

# Count the remaining number of rows after removing outliers and Deck summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
```

```
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)
# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State>Box beam>Asphalt WS>Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 4: Data Processing and Polynomial Regression Analysis for State_Box Beam_Concrete Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Box beam_Concrete WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
```

```
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Deck Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 6:
        # For 'Deck Summary' 6, set upper bound to 88%
        upper_bound = group.quantile(.88)
    if group.name == 5:
        # For 'Deck Summary' 5, set upper bound to 87%
        upper_bound = group.quantile(.87)
    if group.name ==4 :
        # For 'Deck Summary' 4, set upper bound to 80%
        upper_bound = group.quantile(.80)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 75]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
```

```
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]
# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State>Box beam>Concrete WS>Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 5: Data Processing and Polynomial Regression Analysis for State_Slab_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Slab_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
```

```
    if group.name == 9:
        # For 'Deck Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 4:
        # For 'Deck Summary' 4, set lower bound to 10%
        lower_bound = group.quantile(.10)
    if group.name == 3:
        # For 'Deck Summary' 3, set lower bound to 15%
        lower_bound = group.quantile(.15)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)
```

```
# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State>Slab>Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 6: Data Processing and Polynomial Regression Analysis for State_Frame_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/State_Frame_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
```

```
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 20%
        lower_bound = group.quantile(.26)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3 ]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]
```

```
# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for State>Frame>Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 7: Data Processing and Polynomial Regression Analysis for County_Stringer_PS Beam_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Stringer_PS beam_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)
# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
```

```
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 10%
        lower_bound = group.quantile(.10)
    return (group < lower_bound) | (group > upper_bound)
# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3 ]
filtered_data = filtered_data[filtered_data[x_column] <= 80]
# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
```

```
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County_Stringer_PS beam_Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 8: Data Processing and Polynomial Regression Analysis for County_Stringer_Steel Beam_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Stringer_Steel beam_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
```

```
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 100]
# Count the remaining number of rows after removing outliers and Superstructure summary < 4
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))
```

```
# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County_Stringer_Steel beam_Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 9: Data Processing and Polynomial Regression Analysis for County_Box beam_Asphalt Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

```
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Box beam_Asphalt WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)
# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 10%
        lower_bound = group.quantile(.12)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)
```

```
# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])
# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)
x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d
```

```
# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County_Box beam_Asphalt WS_Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 10: Data Processing and Polynomial Regression Analysis for County_Box Beam_Concrete Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Box beam_Concrete WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
```

```
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)
# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)
# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(.25)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 80]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)
```

```
# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)
```

```
# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County_Box beam_Concrete WS_Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 11: Data Processing and Polynomial Regression Analysis for County_Slab_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Slab_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()
```

```
# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 5:
        # For 'Superstructure Summary' 5, set lower bound to 10%
        lower_bound = group.quantile(.10)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 120]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)
# Reset the index of the filtered data
```

```
filtered_data.reset_index(drop=True, inplace=True)
x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
```

```
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County> Slab> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 12: Data Processing and Polynomial Regression Analysis for County_Frame_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/County_Frame_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]
# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)
```

```
# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 4:
        # For 'Superstructure Summary' 4, set lower bound to 12%
        lower_bound = group.quantile(.12)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 10%
        lower_bound = group.quantile(.10)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)
x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()
```

```
# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for County> Frame> Superstructure')
plt.legend()
```

```
# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 13: Data Processing and Polynomial Regression Analysis for City or Municipal_ Stringer_ PS Beam_ Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Stringer_PS beam_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
```

```
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
        upper_bound = group.quantile(.90)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
```

```
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Stringer> PS beam> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 14: Data Processing and Polynomial Regression Analysis for City or Municipal_ Stringer_ Steel Beam_ Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Stringer_Steel beam_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
   lower_bound = group.quantile(lower_percentile / 100)
```

```
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 100]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]
```

```
# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Stringer > Steel beam> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 15: Data Processing and Polynomial Regression Analysis for City or Municipal_Box Beam_Asphalt Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
```

```
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Box beam_Asphalt WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 20%
        lower_bound = group.quantile(.27)
```

```
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 70]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
```

```
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)
# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Box beam> Asphalt WS> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 16: Data Processing and Polynomial Regression Analysis for City or Municipal_Box beam_Concrete Wearing Surface_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
```

```python
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Box beam_Concrete WS_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)

# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()
# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 6:
        # For 'Superstructure Summary' 6, set upper bound to 85%
        upper_bound = group.quantile(.85)
    if group.name == 4:
        # For 'Superstructure Summary' 4, set upper bound to 85%
        upper_bound = group.quantile(.85)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)
```

```
# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >= 3]
filtered_data = filtered_data[filtered_data[x_column] <= 65]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), 63, 1000)
```

```
# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Box beam> Concrete WS> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 17: Data Processing and Polynomial Regression Analysis for City or Municipal_Slab_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Slab_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)
```

```
# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 7.5
upper_percentile = 92.5

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0%
        lower_bound = group.quantile(0)
    if group.name == 5:
        # For 'Superstructure Summary' 5, set lower bound to 10%
        lower_bound = group.quantile(.10)
    if group.name == 4:
        # For 'Superstructure Summary' 4, set lower bound to 10%
        lower_bound = group.quantile(.10)
    if group.name == 3:
         # For 'Superstructure Summary' 3, set lower bound to 10%
        lower_bound = group.quantile(.10)
    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)
```

```
# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Superstructure summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 110]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
   # Our model is y = ax^3 + bx^2 + cx + d
   # We're forcing d to be 9, so our optimization variables are a, b, and c
   a, b, c = coeffs
   d = 9  # Fixed intercept
   y_pred = a * x**3 + b * x**2 + c * x + d
   return np.sum((y - y_pred)**2)

# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)
```

```
# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Slab> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

Python Script 18: Data Processing and Polynomial Regression Analysis for City or Municipal_Frame_Superstructure

```
import numpy as np
from sklearn.metrics import r2_score
import pandas as pd
from scipy.stats import zscore
from google.colab import drive
import matplotlib.pyplot as plt
from scipy.optimize import minimize
drive.mount('/content/drive')

# Read the Excel file
df = pd.read_excel('/content/drive/MyDrive/City or Municipal_Frame_Superstructure.xlsx')

# Count the initial number of rows
initial_rows = len(df)
```

```
# Replace 'x' and 'y' with the actual column names in your Excel file
x_column = 'Reset Age'  # Replace with the correct column name for 'x'
y_column = 'Superstructure Summary'  # Replace with the correct column name for 'y'

# Convert 'x' and 'y' columns to numeric types
df[x_column] = pd.to_numeric(df[x_column], errors='coerce')
df[y_column] = pd.to_numeric(df[y_column], errors='coerce')

# Remove rows with any NaN values
df = df.dropna()

# Count the remaining number of rows after removing blank and NaN
remaining_rows_1 = len(df)

# Remove rows where either 'x' or 'y' is negative
df = df[(df[x_column].ge(0)) & (df[y_column].ge(0))]

# Count the remaining number of rows where either 'x' or 'y' is negative
remaining_rows_2 = len(df)

# Define percentiles for outlier detection
lower_percentile = 10
upper_percentile = 90

# Define a function to detect outliers based on percentiles
def has_outliers(group):
    lower_bound = group.quantile(lower_percentile / 100)
    upper_bound = group.quantile(upper_percentile / 100)
    if group.name == 9:
        # For 'Superstructure Summary' 9, set lower bound to 0% and upper bound to 85%
        lower_bound = group.quantile(0)
        upper_bound = group.quantile(.775)
    if group.name == 8:
        # For 'Superstructure Summary' 8, set upper bound to 85%
        upper_bound = group.quantile(.80)
    if group.name == 7:
        # For 'Superstructure Summary' 7, set lower bound to 15% and upper bound to 85%
        lower_bound = group.quantile(.15)
        upper_bound = group.quantile(.85)
    if group.name == 6:
        # For 'Superstructure Summary' 6, set lower bound to 20%
        lower_bound = group.quantile(.20)
    if group.name == 5:
        # For 'Superstructure Summary' 5, set lower bound to 25% and upper bound to 90%
        lower_bound = group.quantile(.25)
```

```
        upper_bound = group.quantile(.875)
    if group.name == 4:
        # For 'Superstructure Summary' 4, set lower bound to 15%
        lower_bound = group.quantile(.15)
    if group.name == 3:
        # For 'Superstructure Summary' 3, set lower bound to 15%
        lower_bound = group.quantile(.15)

    return (group < lower_bound) | (group > upper_bound)

# Check if the "Age" column has outliers in the filtered data
age_outliers = df.groupby(y_column)[x_column].transform(has_outliers)

# Filter out rows with outliers in the filtered data
filtered_data = df[~age_outliers]

# Remove rows where 'Deck summary' column is less than 3
filtered_data = filtered_data[filtered_data[y_column] >=3  ]
filtered_data = filtered_data[filtered_data[x_column] <= 65]

# Count the remaining number of rows after removing outliers and Superstructure summary < 3
remaining_rows_3 = len(filtered_data)

# Convert 'x' and 'y' columns to numeric types after all filtering steps
filtered_data[x_column] = pd.to_numeric(filtered_data[x_column])
filtered_data[y_column] = pd.to_numeric(filtered_data[y_column])

# Store the number of data points after filtering
filtered_count = len(filtered_data)

# Reset the index of the filtered data
filtered_data.reset_index(drop=True, inplace=True)

x_values = filtered_data[x_column].to_numpy()
y_values = filtered_data[y_column].to_numpy()

# Objective function to minimize (sum of squared errors)
def objective_function(coeffs, x, y):
    # Our model is y = ax^3 + bx^2 + cx + d
    # We're forcing d to be 9, so our optimization variables are a, b, and c
    a, b, c = coeffs
    d = 9  # Fixed intercept
    y_pred = a * x**3 + b * x**2 + c * x + d
    return np.sum((y - y_pred)**2)
```

```
# Initial guess for coefficients a, b, and c
initial_guess = [0, 0, 0]

# Perform the optimization
result = minimize(objective_function, initial_guess, args=(x_values, y_values))

# Extract the optimized coefficients
a, b, c = result.x
d = 9  # Fixed intercept

# Generate x values for the fitted curve
x_fit = np.linspace(x_values.min(), x_values.max(), 1000)

# Evaluate the polynomial at x_fit values
y_fit = a * x_fit**3 + b * x_fit**2 + c * x_fit + d

# Calculate R-squared value
y_predicted = a * x_values**3 + b * x_values**2 + c * x_values + d
r_squared = r2_score(y_values, y_predicted)

# Print the fitted equation
equation = f"y = {a}x^3 + {b}x^2 + {c}x + 9"
print("Fitted equation:", equation)

# Print the R-squared value
print("R-squared value:", r_squared)

# Plot the filtered data points and the fitted curve
plt.figure(figsize=(8, 5))
plt.scatter(x_values, y_values, label="Filtered Data")
plt.plot(x_fit, y_fit, color='red', label="Fitted Curve")
plt.xlabel("Reset Age")
plt.ylabel("Superstructure Summary")
plt.title('3rd Degree Polynomial Curve Fit for City or Municipal> Frame> Superstructure')
plt.legend()

# Print the number of rows before and after cleaning
print("Initial number of rows:", initial_rows)
print("Remaining number of rows after removing blank cell and NaN:", remaining_rows_1)
print("Remaining number of rows after removing negative values:", remaining_rows_2)
print("Remaining number of rows after removing outliers:", remaining_rows_3)
```

7.4. Research Findings in Detail

7.4.1. Deterioration Curves: Polynomial Regression Models

Table 6. Polynomial Regression Models of Superstructure Designs

Maintenance Responsibility	Superstructure Design	Polynomial Regression Model
State	Stringer PS Beam	$CR = 9 - 0.1141Age + 0.0025Age^2 - 2.4326e^{-5}Age^3$
State	Stringer Steel Beam	$CR = 9 - 0.1431Age + 0.0030Age^2 - 2.2547e^{-5}Age^3$
State	Box Beam AS WS	$CR = 9 - 0.1295Age + 0.0039Age^2 - 6.1942e^{-5}Age^3$
State	Box Beam Conc Deck	$CR = 9 - 0.0880Age + 0.0005Age^2 + 3.8438e^{-6}Age^3$
State	Slab	$CR = 9 - 0.1267Age + 0.0023Age^2 - 1.8599e^{-5}Age^3$
State	Frame	$CR = 9 - 0.1090Age + 0.0009Age^2 - 2.1029e^{-6}Age^3$
County	Stringer PS Beam	$CR = 9 - 0.0839Age + 0.0004Age^2 - 1.1443e^{-6}Age^3$
County	Stringer Steel Beam	$CR = 9 - 0.1658Age + 0.0031Age^2 - 1.9737e^{-5}Age^3$
County	Box Beam AS WS	$CR = 9 - 0.0733Age + 0.0005Age^2 - 5.8163e^{-6}Age^3$
County	Box Beam Conc Deck	$CR = 9 - 0.0781Age + 0.0006Age^2 - 6.2310e^{-6}Age^3$
County	Slab	$CR = 9 - 0.1038Age + 0.0015Age^2 - 9.2050e^{-6}Age^3$
County	Frame	$CR = 9 - 0.0750Age + 0.0003Age^2 + 3.0896e^{-8}Age^3$
City/ Municipal	Stringer PS Beam	$CR = 9 - 0.1077Age + 0.0006Age^2 - 8.9589e^{-7}Age^3$
City/ Municipal	Stringer Steel Beam	$CR = 9 - 0.1360Age + 0.0021Age^2 - 1.4308e^{-5}Age^3$
City/ Municipal	Box Beam AS WS	$CR = 9 - 0.1112Age + 0.0016Age^2 - 1.7282e^{-5}Age^3$
City/ Municipal	Box Beam Conc Deck	$CR = 9 - 0.0444Age - 0.0028Age^2 + 3.5872e^{-5}Age^3$
City/ Municipal	Slab	$CR = 9 - 0.0861Age + 0.0007Age^2 - 2.6701e^{-6}Age^3$
City/ Municipal	Frame	$CR = 9 - 0.2306Age + 0.0098Age^2 - 1.3447e^{-4}Age^3$

```
Mounted at /content/drive
Fitted equation: y = -2.4326291098538053e-05x^3 + 0.0024514486337479002x^2 + -0.11413198813439997x + 9
R-squared value: 0.494352594643793
Initial number of rows: 9858
Remaining number of rows after removing blank cell and NaN: 9856
Remaining number of rows after removing negative values: 9677
Remaining number of rows after removing outliers: 8476
```

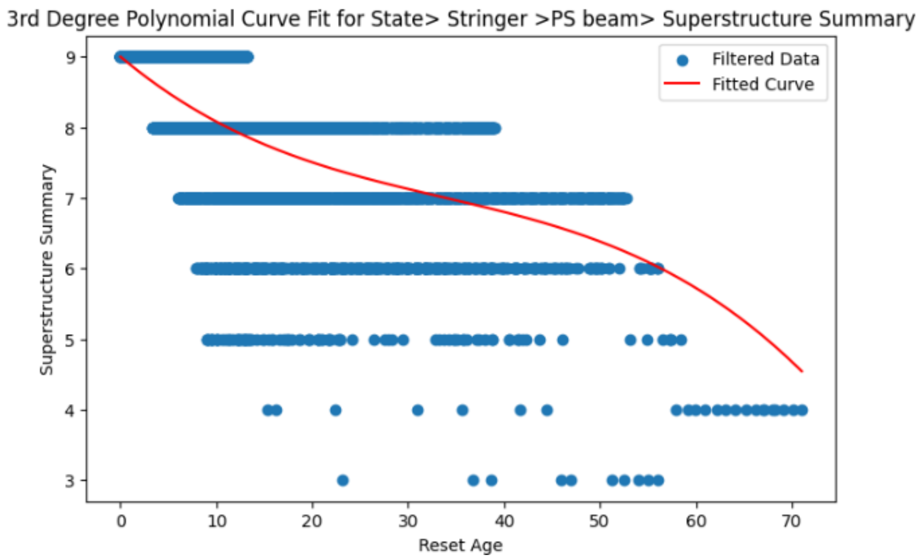

Figure 13. Polynomial Regression Model for State> Stringer> PS Beam> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -2.2547235241928936e-05x^3 + 0.003001314936586559x^2 + -0.14313732087236558x + 9
R-squared value: 0.3092235716205358
Initial number of rows: 197854
Remaining number of rows after removing blank cell and NaN: 197794
Remaining number of rows after removing negative values: 195380
Remaining number of rows after removing outliers: 167042
```

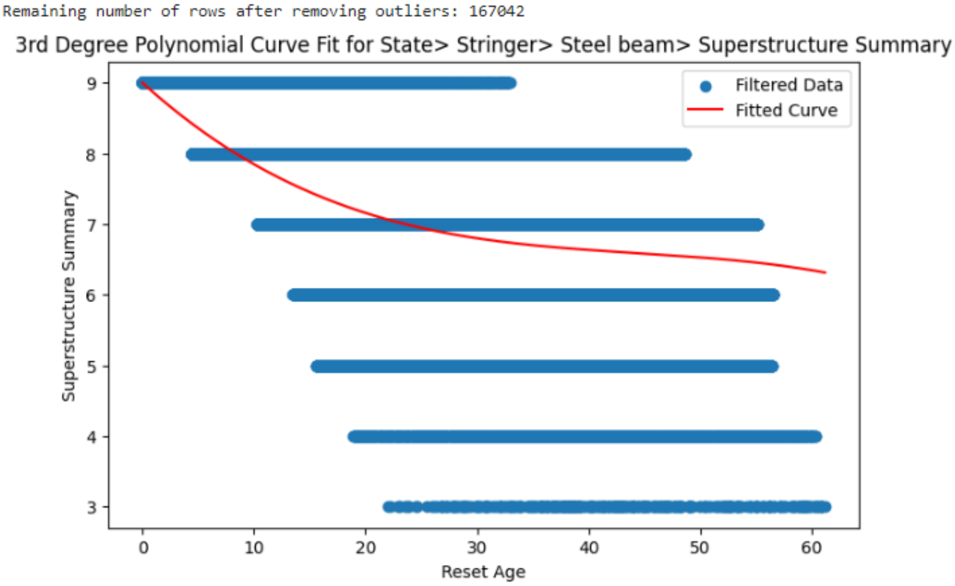

Figure 14. Polynomial Regression Model for State> Stringer> Steel Beam> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = -6.194207807226607e-05x^3 + 0.0039045167136125356x^2 + -0.12950023108341752x + 9
R-squared value: 0.42624612205624457
Initial number of rows: 34469
Remaining number of rows after removing blank cell and NaN: 34468
Remaining number of rows after removing negative values: 34239
Remaining number of rows after removing outliers: 29439
```

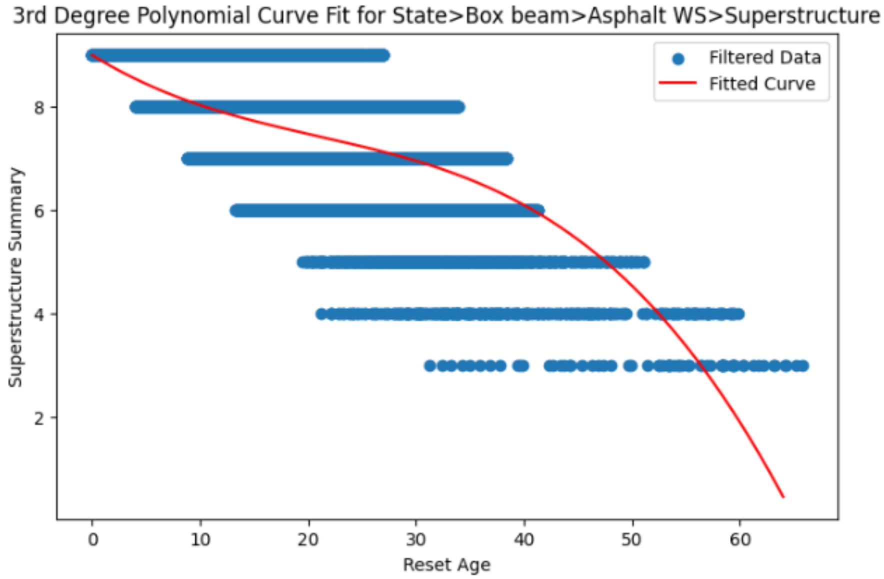

Figure 15. Polynomial Regression Model for State> Box beam> Asphalt Wearing Surface> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = 3.843814890245418e-06x^3 + -0.0005018613198119715x^2 + -0.08795427091907487x + 9
R-squared value: 0.5285317860607724
Initial number of rows: 7005
Remaining number of rows after removing blank cell and NaN: 6999
Remaining number of rows after removing negative values: 6829
Remaining number of rows after removing outliers: 5845
```

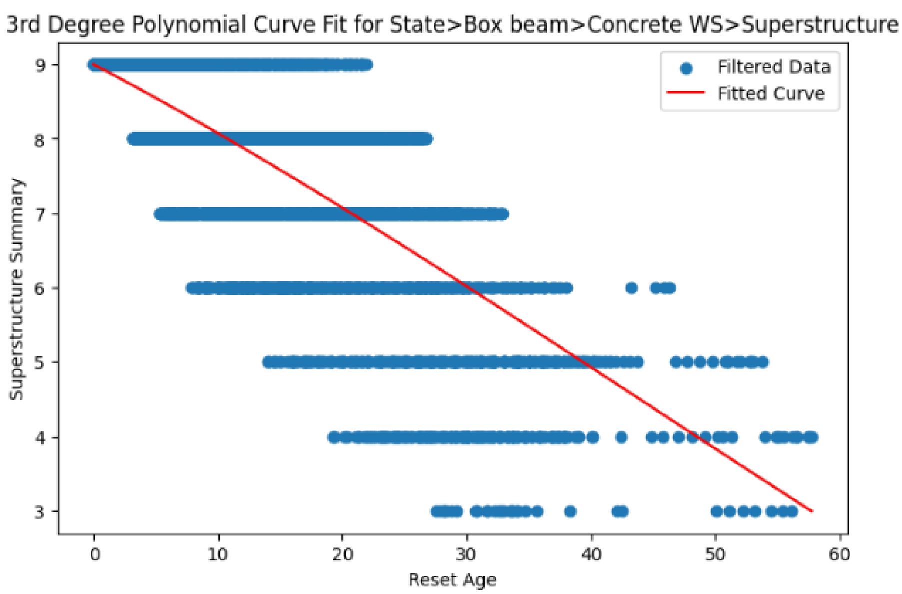

Figure 16. Polynomial Regression Model for State> Box beam> Concrete Deck> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -1.859927302407903e-05x^3 + 0.0023014349964693053x^2 + -0.12670976350867239x + 9
R-squared value: 0.4576376003981335
Initial number of rows: 112747
Remaining number of rows after removing blank cell and NaN: 112391
Remaining number of rows after removing negative values: 112180
Remaining number of rows after removing outliers: 95477
```

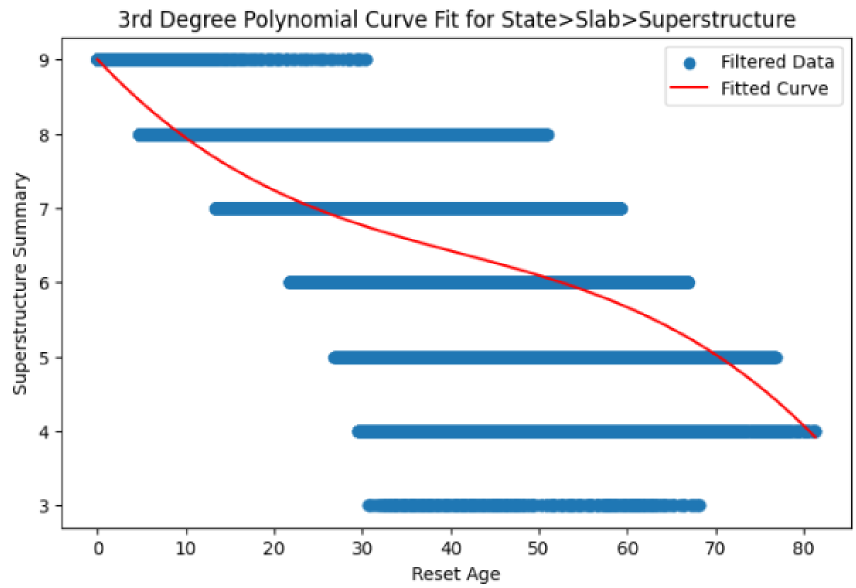

Figure 17. Polynomial Regression Model for State> Slab> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -2.1028847605775066e-06x^3 + 0.0009413757925997449x^2 + -0.1089557250675223x + 9
R-squared value: 0.6884630417477076
Initial number of rows: 9402
Remaining number of rows after removing blank cell and NaN: 8764
Remaining number of rows after removing negative values: 8658
Remaining number of rows after removing outliers: 7496
```

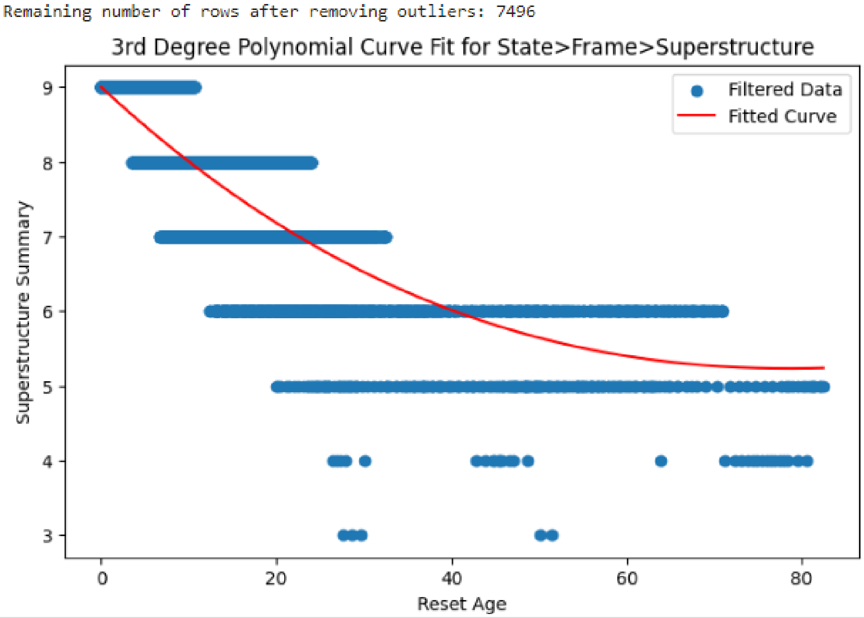

Figure 18. Polynomial Regression Model for State> Frame> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = -1.1443004479512619e-06x^3 + 0.0004256186503234071x^2 + -0.08388679815839047x + 9
R-squared value: 0.5503143654915622
Initial number of rows: 7195
Remaining number of rows after removing blank cell and NaN: 7176
Remaining number of rows after removing negative values: 7171
Remaining number of rows after removing outliers: 6187
```

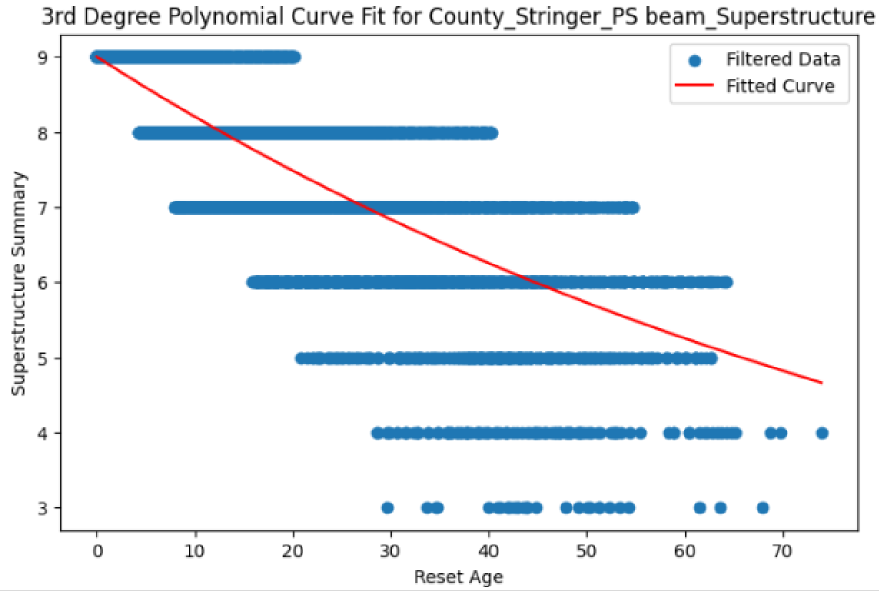

Figure 19. Polynomial Regression Model for County> Stringer> PS Beam> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = -1.9736979452542843e-05x^3 + 0.003061177262838897x^2 + -0.16583832363405984x + 9
R-squared value: 0.36463117031965964
Initial number of rows: 210751
Remaining number of rows after removing blank cell and NaN: 210397
Remaining number of rows after removing negative values: 208576
Remaining number of rows after removing outliers: 177452
```

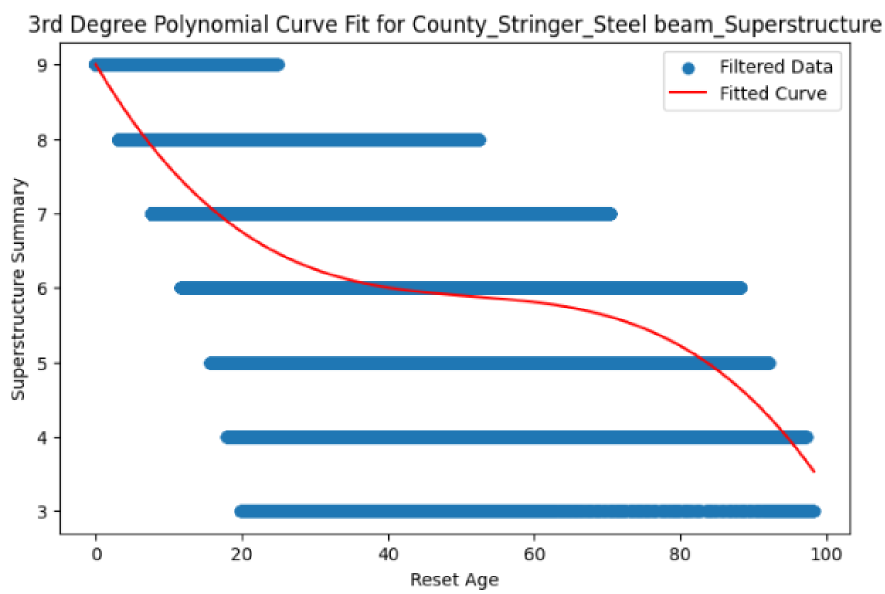

Figure 20. Polynomial Regression Model for County > Stringer> Steel Beam> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = -5.816307228104375e-06x^3 + 0.0005470940607255679x^2 + -0.07329823872809142x + 9
R-squared value: 0.46991437690531856
Initial number of rows: 172371
Remaining number of rows after removing blank cell and NaN: 172262
Remaining number of rows after removing negative values: 170939
Remaining number of rows after removing outliers: 147895
```

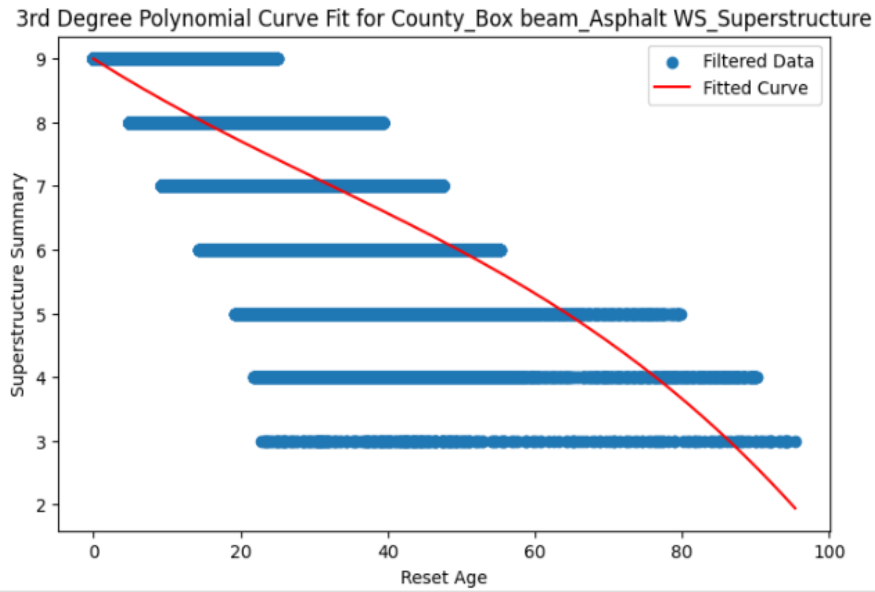

Figure 21. Polynomial Regression Model for County > Box Beam> Asphalt Wearing Surface> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -6.230987247088084e-06x^3 + 0.0005512077087174451x^2 + -0.0781418236207846x + 9
R-squared value: 0.5587130939781323
Initial number of rows: 9859
Remaining number of rows after removing blank cell and NaN: 9859
Remaining number of rows after removing negative values: 9854
Remaining number of rows after removing outliers: 8549
```

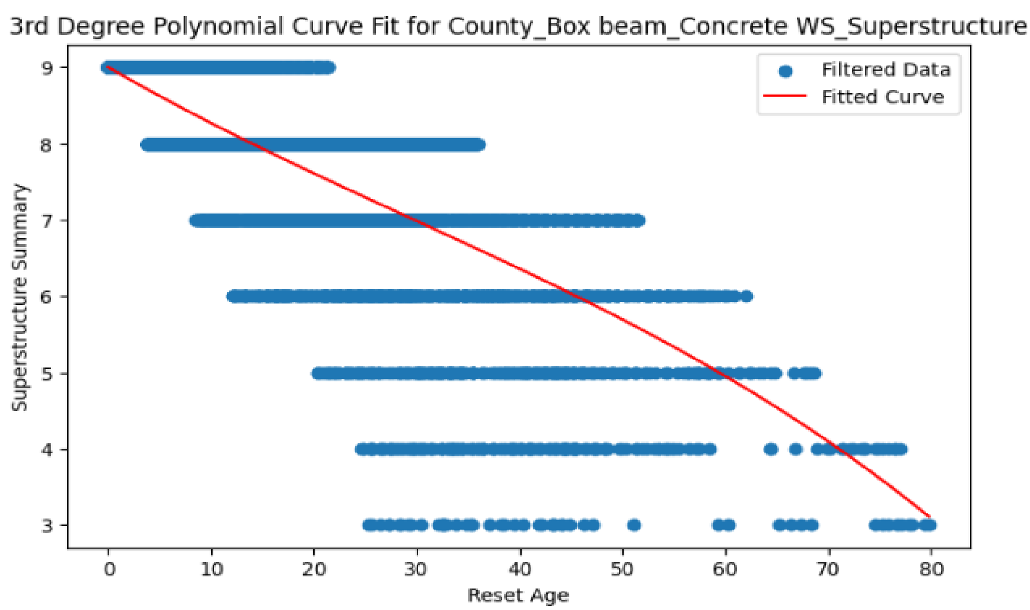

Figure 22. Polynomial Regression Model for County > Box Beam> Concrete Deck> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -9.205006222508724e-06x^3 + 0.0014970599006722855x^2 + -0.10375949323969484x + 9
R-squared value: 0.31243991111411973
Initial number of rows: 143567
Remaining number of rows after removing blank cell and NaN: 142681
Remaining number of rows after removing negative values: 142595
Remaining number of rows after removing outliers: 120707
```

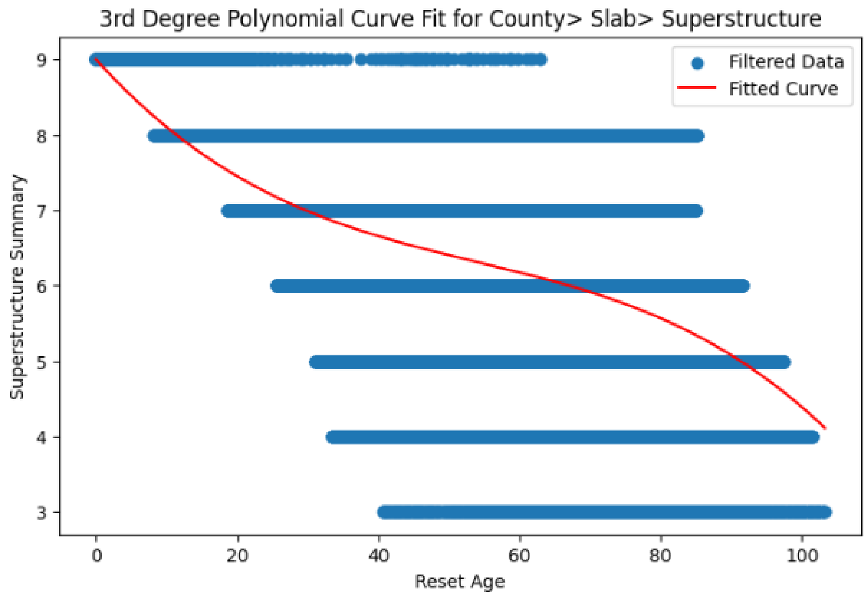

Figure 23. Polynomial Regression Model for County > Slab> Superstructure

```
Mounted at /content/drive
Fitted equation: y = 3.0896214501313084e-08x^3 + 0.00027017696025167636x^2 + -0.07503106614645175x + 9
R-squared value: 0.5683863782839624
Initial number of rows: 48548
Remaining number of rows after removing blank cell and NaN: 45016
Remaining number of rows after removing negative values: 44882
Remaining number of rows after removing outliers: 39392
```

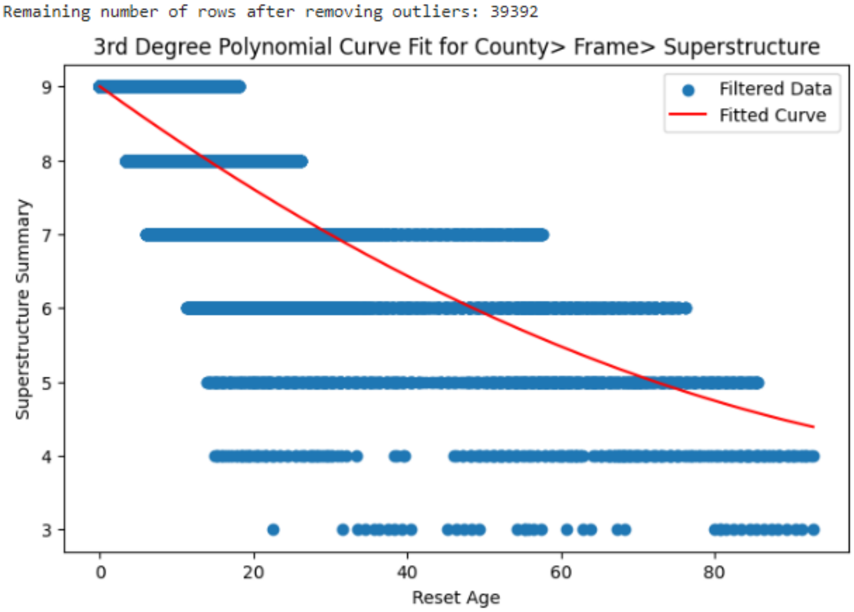

Figure 24. Polynomial Regression Model for County > Frame> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -8.958885492390429e-07x^3 + 0.0006401703230012156x^2 + -0.10771931224540002x + 9
R-squared value: 0.6358808166819482
Initial number of rows: 1149
Remaining number of rows after removing blank cell and NaN: 1149
Remaining number of rows after removing negative values: 1149
Remaining number of rows after removing outliers: 976
```

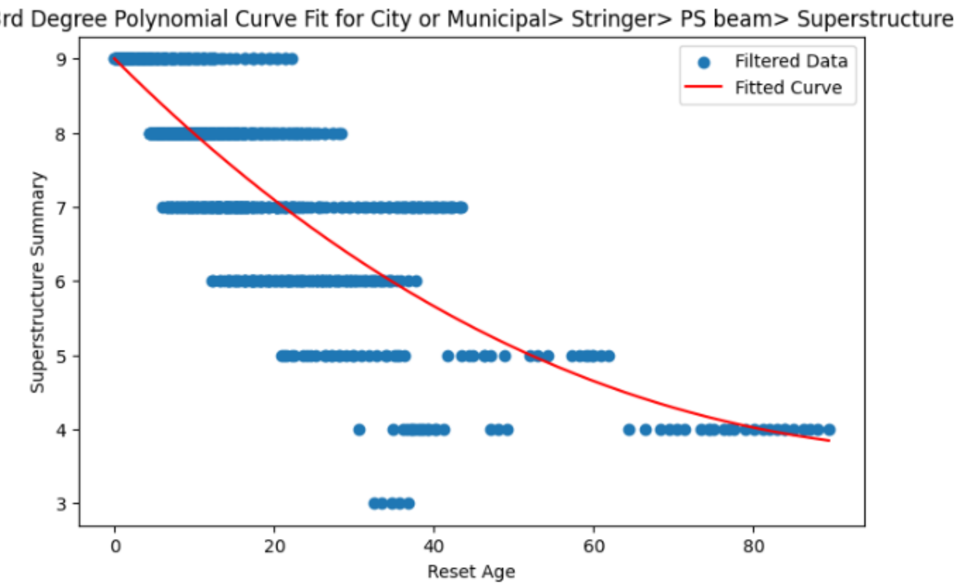

Figure 25. Polynomial Regression Model for City/Municipal> Stringer> PS Beam> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -1.4307559130221088e-05x^3 + 0.0021091368304080953x^2 + -0.13598600037074604x + 9
R-squared value: 0.49178155883336705
Initial number of rows: 10724
Remaining number of rows after removing blank cell and NaN: 10708
Remaining number of rows after removing negative values: 10594
Remaining number of rows after removing outliers: 8974
```

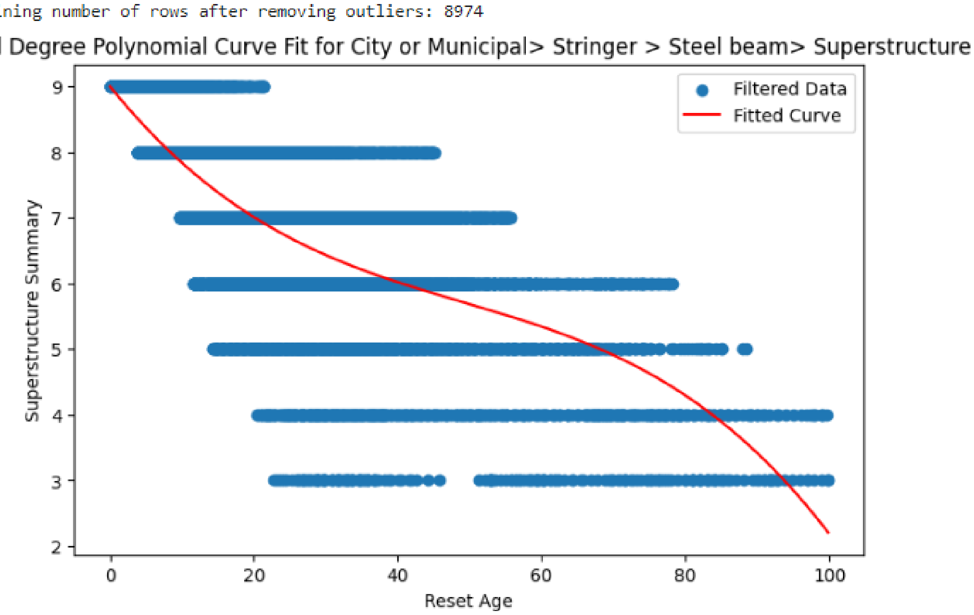

Figure 26. Polynomial Regression Model for City/Municipal > Stringer> Steel Beam> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -1.7281790174680163e-05x^3 + 0.0015883420555921676x^2 + -0.11121075515512653x + 9
R-squared value: 0.4605623521863398
Initial number of rows: 6308
Remaining number of rows after removing blank cell and NaN: 6308
Remaining number of rows after removing negative values: 6257
Remaining number of rows after removing outliers: 5329
```

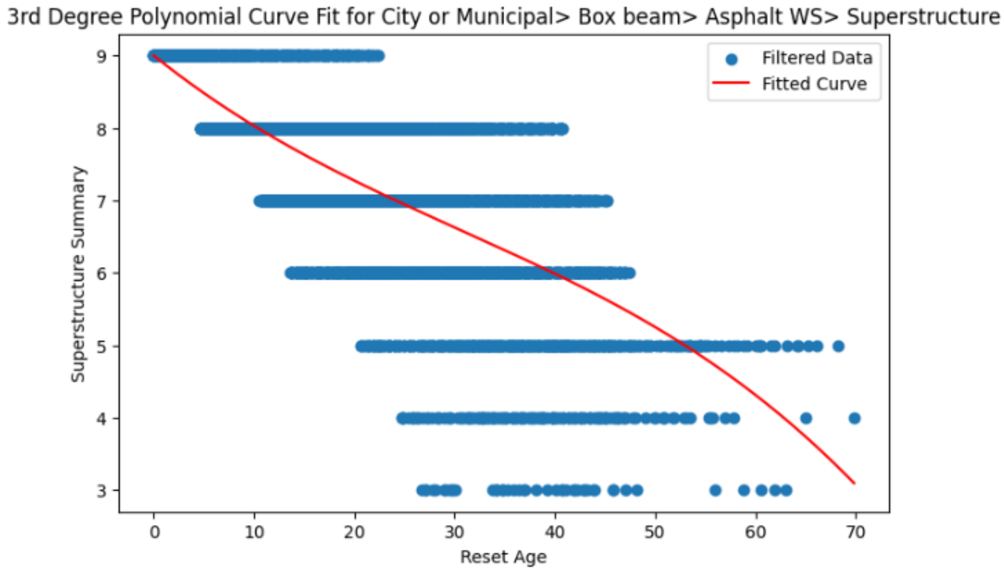

Figure 27. Polynomial Regression Model for City/Municipal > Box Beam> Asphalt Wearing Surface> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = 3.5871774657323e-05x^3 + -0.002799285435716417x^2 + -0.04440971854621387x + 9
R-squared value: 0.6382469841557434
Initial number of rows: 2917
Remaining number of rows after removing blank cell and NaN: 2917
Remaining number of rows after removing negative values: 2911
Remaining number of rows after removing outliers: 2479
```

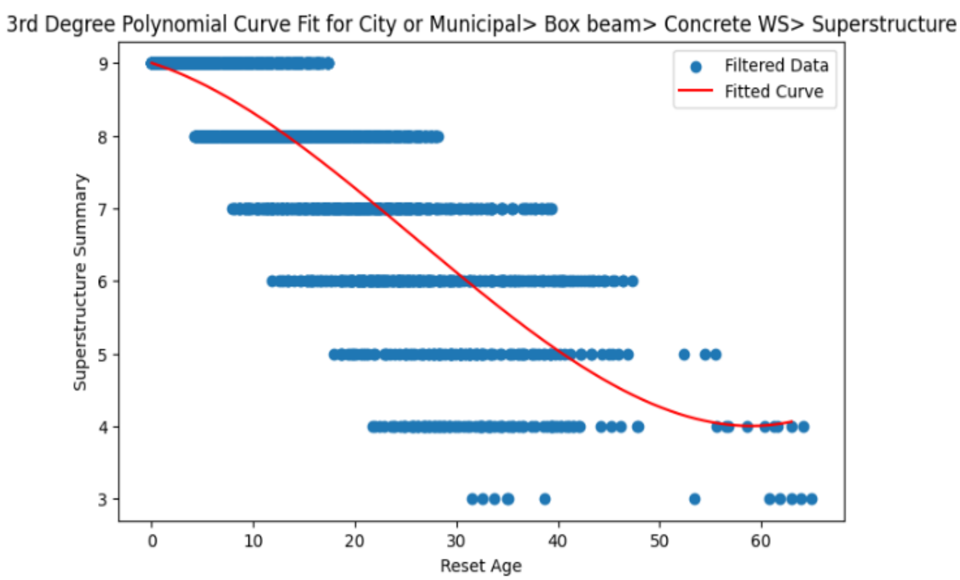

Figure 28. Polynomial Regression Model for City/Municipal > Box Beam> Concrete Deck> Superstructure

```
Mounted at /content/drive
Fitted equation: y = -2.6700614437103448e-06x^3 + 0.0006989129006007432x^2 + -0.08610791464816227x + 9
R-squared value: 0.5175920011862284
Initial number of rows: 8214
Remaining number of rows after removing blank cell and NaN: 7974
Remaining number of rows after removing negative values: 7947
Remaining number of rows after removing outliers: 6704
```

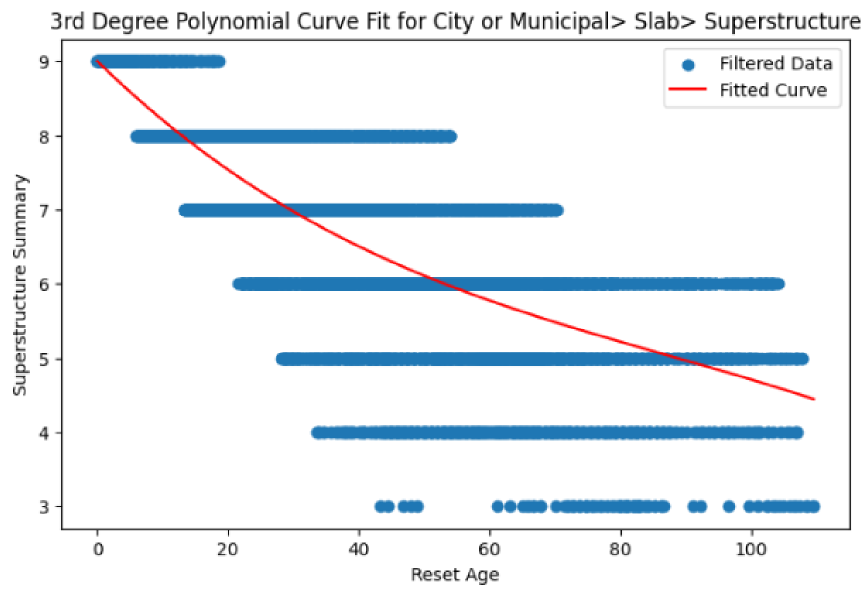

Figure 29. Polynomial Regression Model for City/Municipal > Slab> Superstructure

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Fitted equation: y = -0.00013446644764046145x^3 + 0.009750200023392445x^2 + -0.23064362328079455x + 9
R-squared value: 0.1114715337102764
Initial number of rows: 5954
Remaining number of rows after removing blank cell and NaN: 4452
Remaining number of rows after removing negative values: 4436
Remaining number of rows after removing outliers: 3131
```

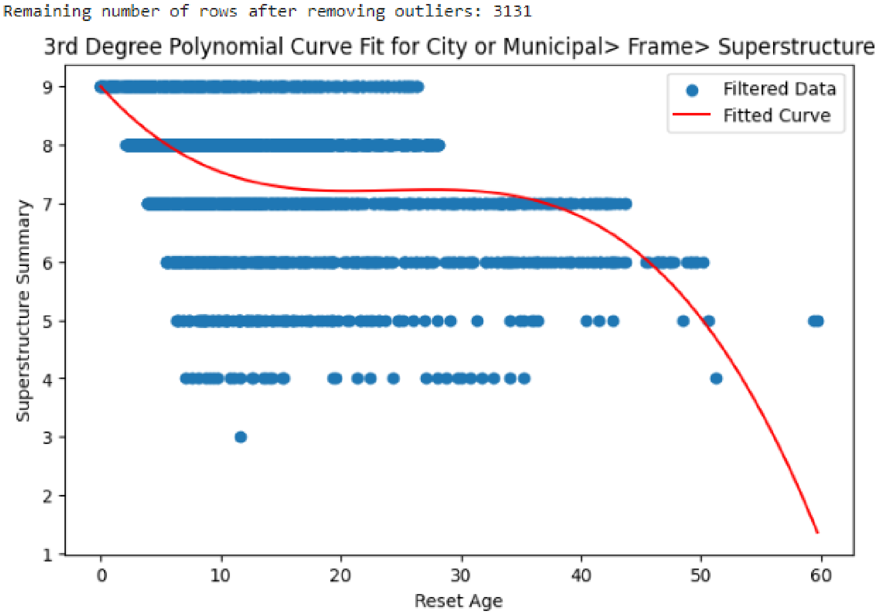

Figure 30. Polynomial Regression Model for City/Municipal > Frame> Superstructure.

7.4.2. Deterioration Curves: Regression Nonlinear Optimization (RNO) Models, Markovian Transition Probabilities, and Data Strengths

State>Stringer PS Beam

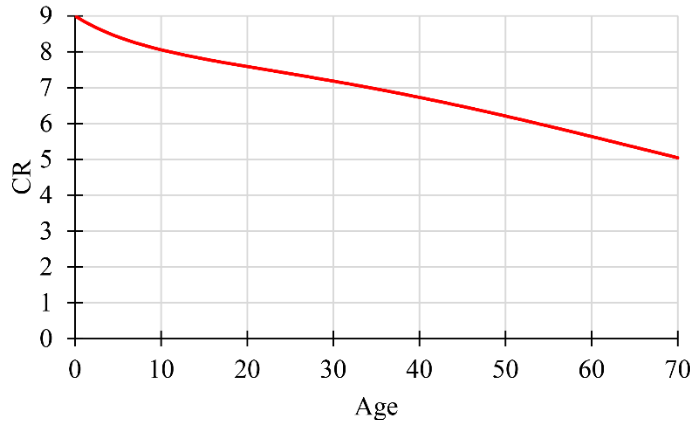

Figure 31. RNO Deterioration Curve, State Stringer PS Beam

Table 7. Transition Matrix, State Stringer PS Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.8598	0.1353	0.0049	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9692	0.0307	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9675	0.0324	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9458	0.0541	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9061	0.0923	0.0016	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8686	0.1124	0.0191	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0533	0.9467	0.0000
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0202	0.9798
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

Figure 32. Data Strengths, State Stringer PS Beam

State>Stringer Steel Beam

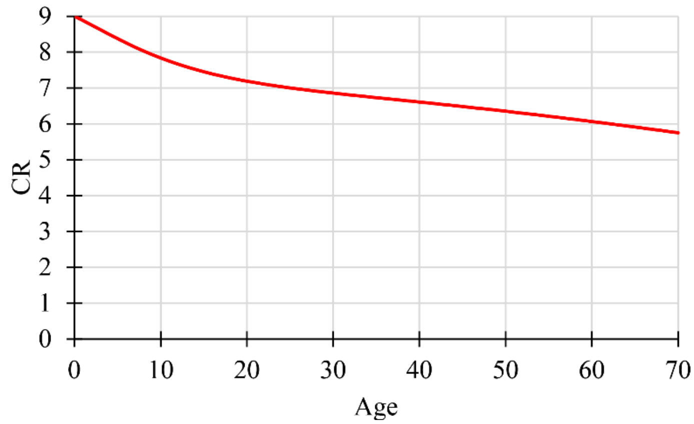

Figure 33. RNO Deterioration Curve, State Stringer Steel Beam

Table 8. Transition Matrix, State Stringer Steel Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.8807	0.1193	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8284	0.1715	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9895	0.0105	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9678	0.0321	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9263	0.0724	0.0012	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9073	0.0744	0.0184	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0201	0.9799	0.0000
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9900
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

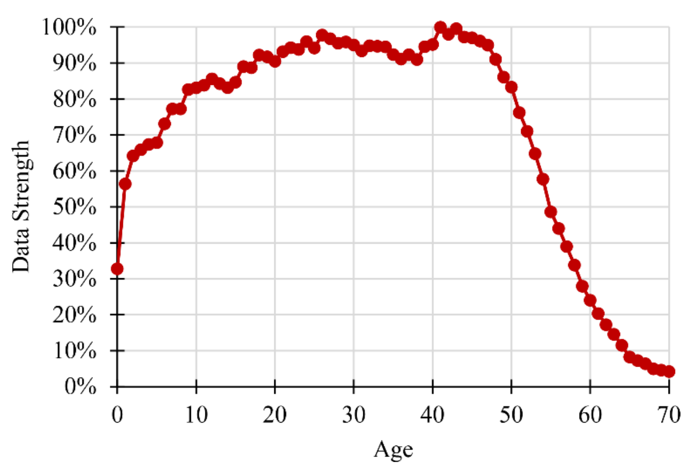

Figure 34. Data Strengths, State Stringer Steel Beam

<u>State>Box Beam AS WS</u>

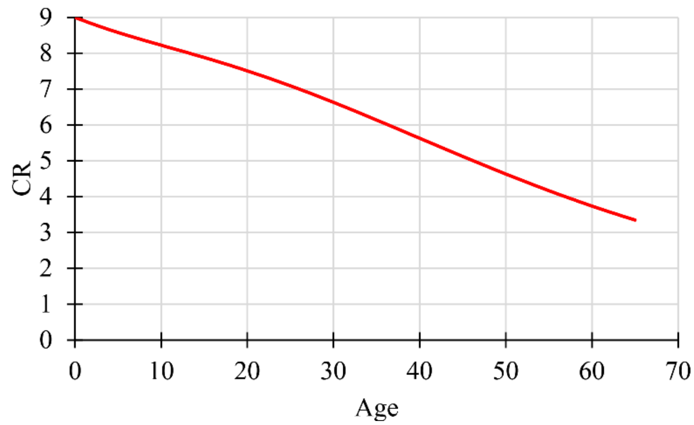

Figure 35. RNO Deterioration Curve, State Box Beam AS WS

Table 9. Transition Matrix, State Box Beam AS WS

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9079	0.0921	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9566	0.0434	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9370	0.0630	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9179	0.0820	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.1033	0.0999	0.7967	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.0100	0.9800	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9443	0.0457
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.9999
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

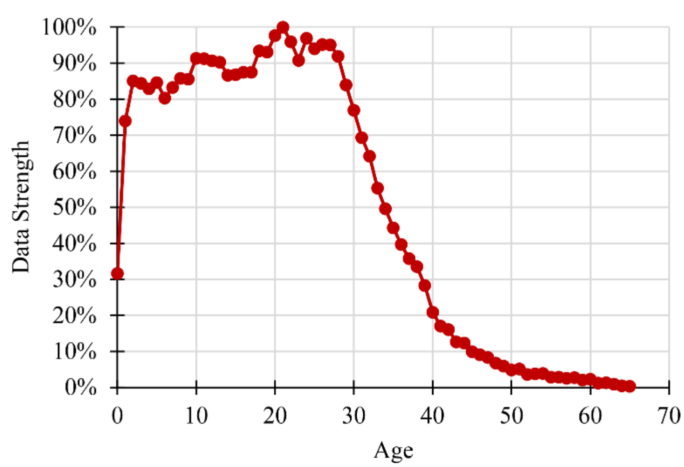

Figure 36. Data Strengths, State Box Beam AS WS

State>Box Beam Conc Deck

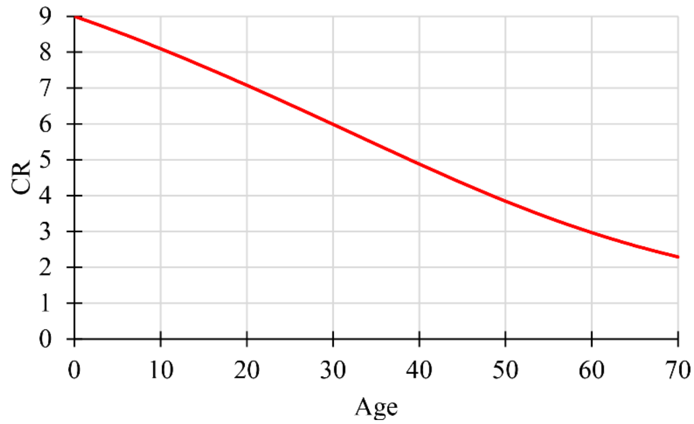

Figure 37. RNO Deterioration Curve, State Box Beam Conc Deck

Table 10. Transition Matrix, State Box Beam Conc Deck

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9163	0.0836	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8976	0.1023	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.8842	0.1157	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.8824	0.1168	0.0008	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9058	0.0940	0.0002	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9254	0.0710	0.0036	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2469	0.6817	0.0714
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0102	0.9898
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

Figure 38. Data Strengths, State Box Beam Conc Deck

State>Slab

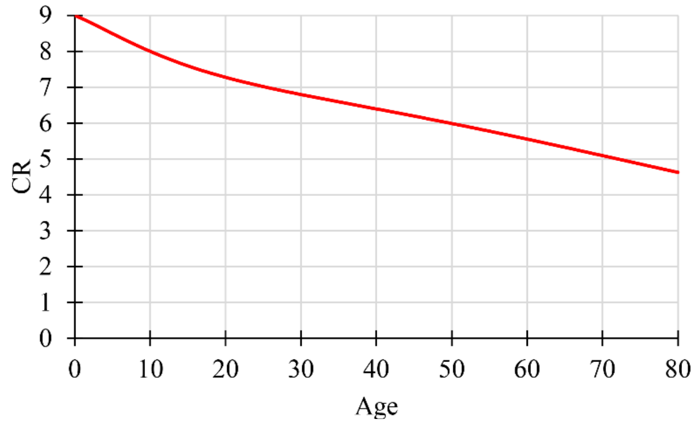

Figure 39. RNO Deterioration Curve, State Slab

Table 11. Transition Matrix, State Slab

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9081	0.0919	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8274	0.1726	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9780	0.0219	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9670	0.0329	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9408	0.0592	0.0000	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9110	0.0854	0.0036	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2713	0.6582	0.0706
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0102	0.9898
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

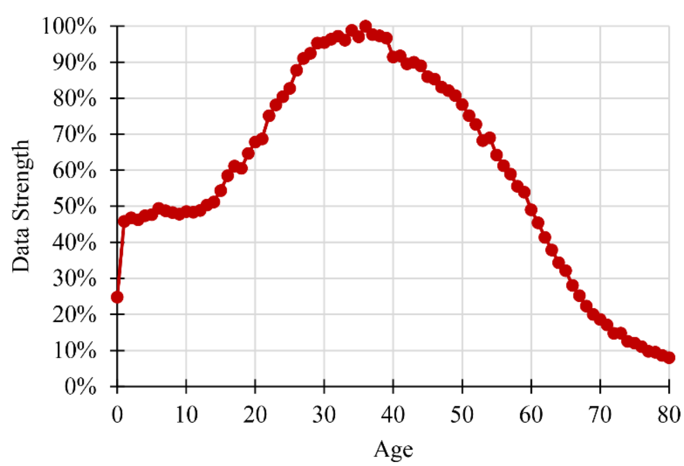

Figure 40. Data Strengths, State Slab

State>Frame

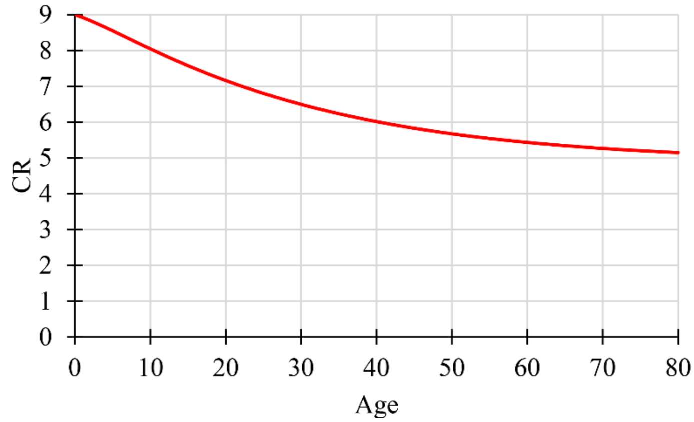

Figure 41. RNO Deterioration Curve, State Frame

Table 12. Transition Matrix, State Frame

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9241	0.0759	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8022	0.1977	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9307	0.0693	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9540	0.0459	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9992	0.0008	0.0000	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9869	0.0099	0.0032	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2751	0.6544	0.0705
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0103	0.9897
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

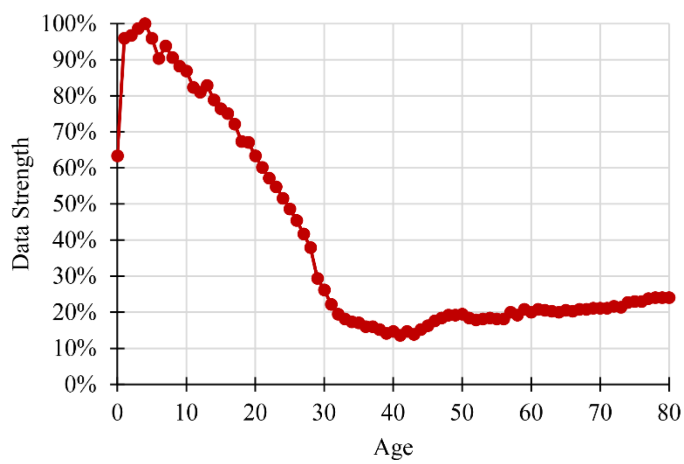

Figure 42. Data Strengths, State Frame

County>Stringer PS Beam

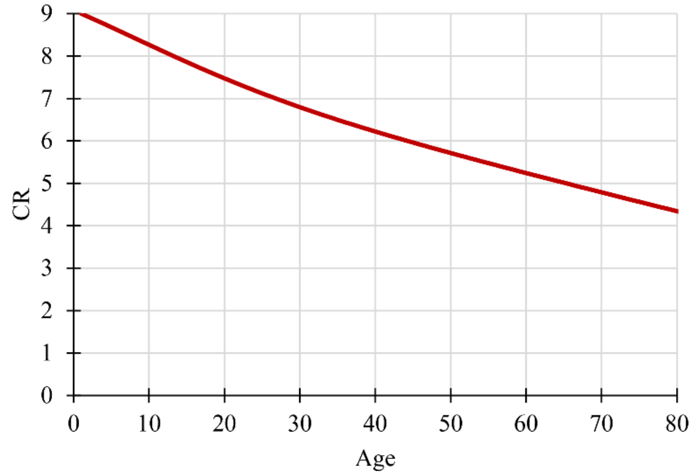

Figure 43. RNO Deterioration Curve, County Stringer PS Beam

Table 13. Transition Matrix, County Stringer PS Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9276	0.0721	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8884	0.1110	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9464	0.0536	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9709	0.0291	0.0000	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9464	0.0535	0.0000	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9051	0.0948	0.0001	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.8620	0.1379	0.0001
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0201	0.9799
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

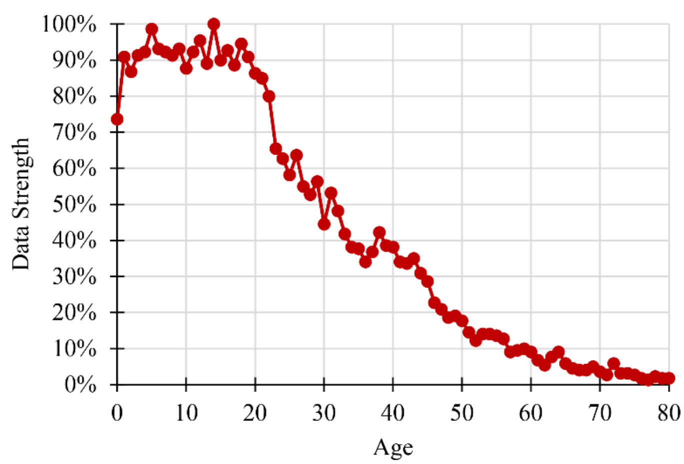

Figure 44. Data Strengths, County Stringer PS Beam

County>Stringer Steel Beam

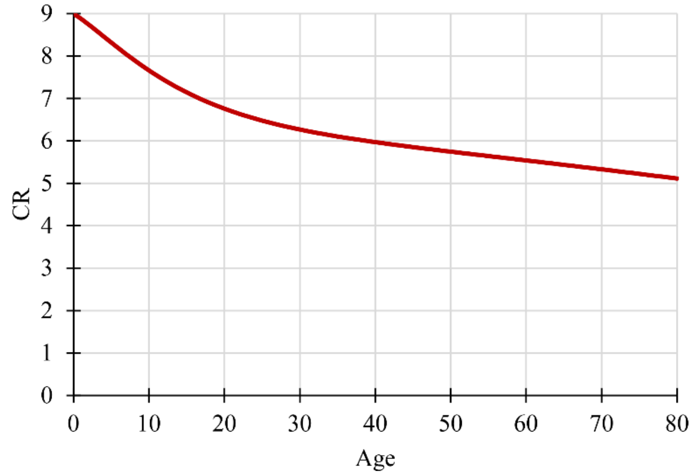

Figure 45. RNO Deterioration Curve, County Stringer Steel Beam

Table 14. Transition Matrix, County Stringer Steel Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.8771	0.1229	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.7890	0.2109	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9193	0.0807	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9925	0.0074	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9651	0.0337	0.0012	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9498	0.0356	0.0147	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0526	0.9474	0.0000
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0201	0.9799
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

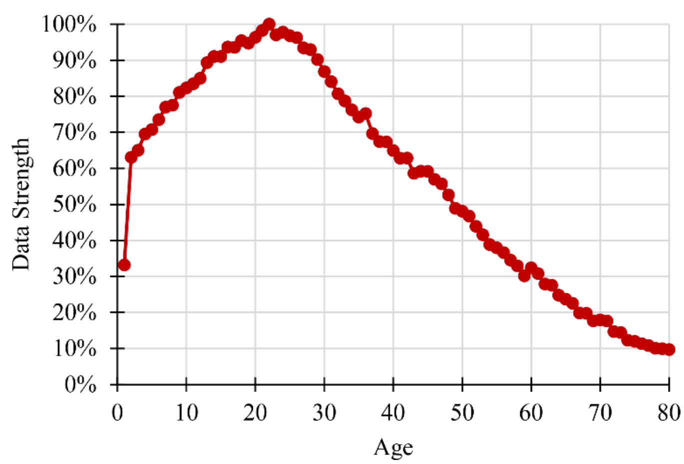

Figure 46. Data Strengths, County Stringer Steel Beam

County>Box Beam AS WS

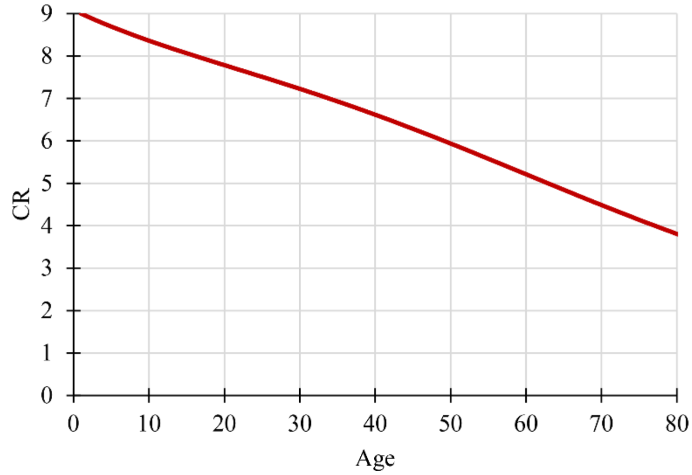

Figure 47. RNO Deterioration Curve, County Box Beam AS WS

Table 15. Transition Matrix, County Box Beam AS WS

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9194	0.0806	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9566	0.0433	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9518	0.0482	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9349	0.0649	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9095	0.0890	0.0015	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8832	0.0985	0.0183	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2096	0.7415	0.0490
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.9999
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

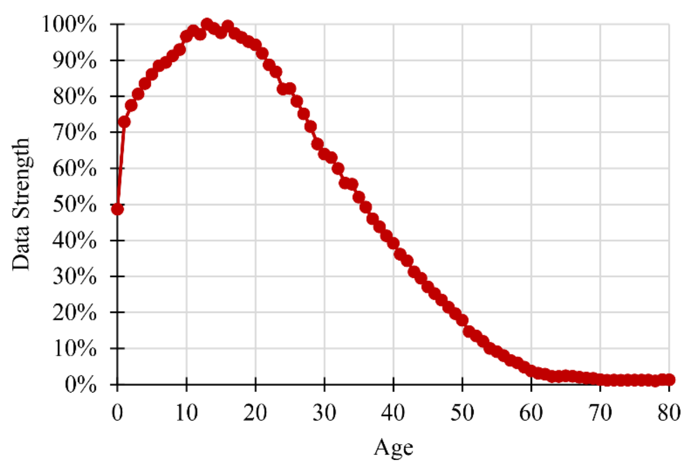

Figure 48. Data Strengths, County Box Beam AS WS

County>Box Beam Conc Deck

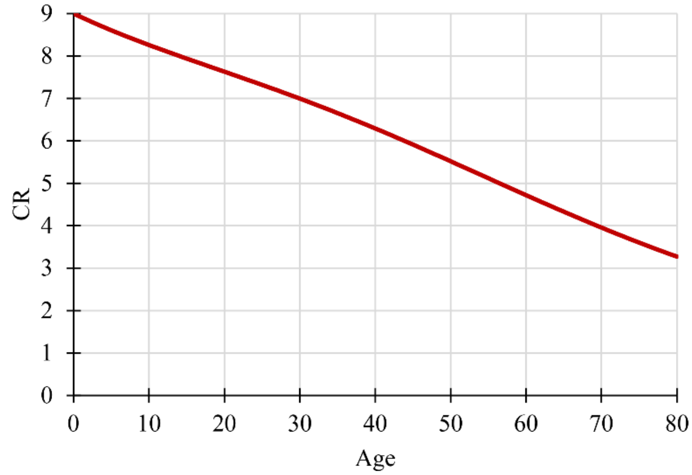

Figure 49. RNO Deterioration Curve, County Box Beam Conc Deck

Table 16. Transition Matrix, County Box Beam Conc Deck

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9163	0.0837	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9487	0.0512	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9464	0.0536	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9300	0.0700	0.0000	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9060	0.0940	0.0000	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8799	0.1165	0.0036	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2585	0.6706	0.0710
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0102	0.9898
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

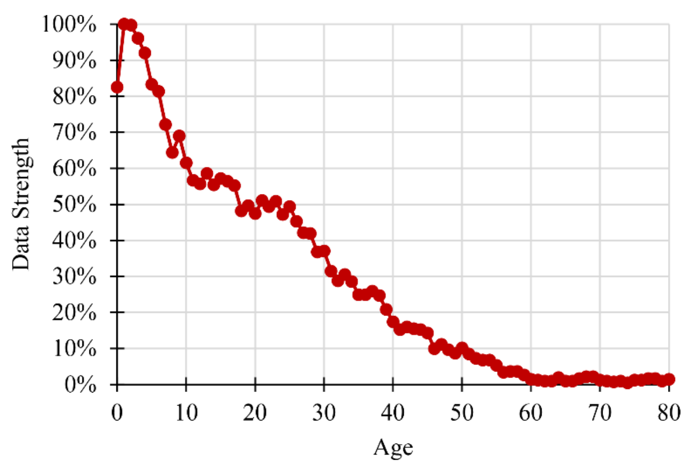

Figure 50. Data Strengths, County Box Beam Conc Deck

County>Slab

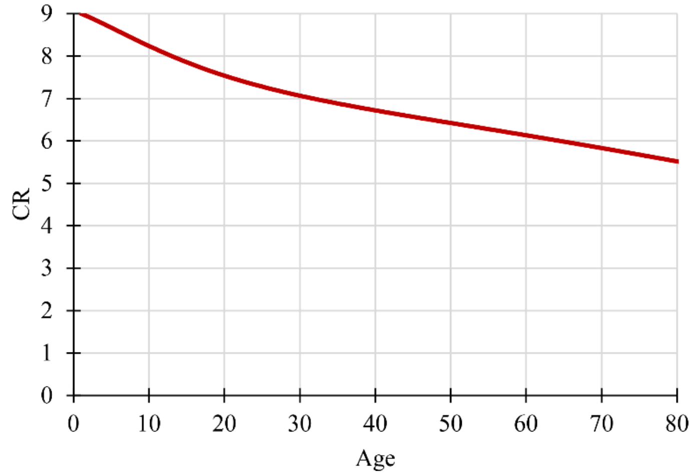

Figure 51. RNO Deterioration Curve, County Slab

Table 17. Transition Matrix, County Slab

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9242	0.0758	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8516	0.1483	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9808	0.0192	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9818	0.0181	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9618	0.0369	0.0014	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9061	0.0782	0.0157	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2164	0.7348	0.0488
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.9999
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

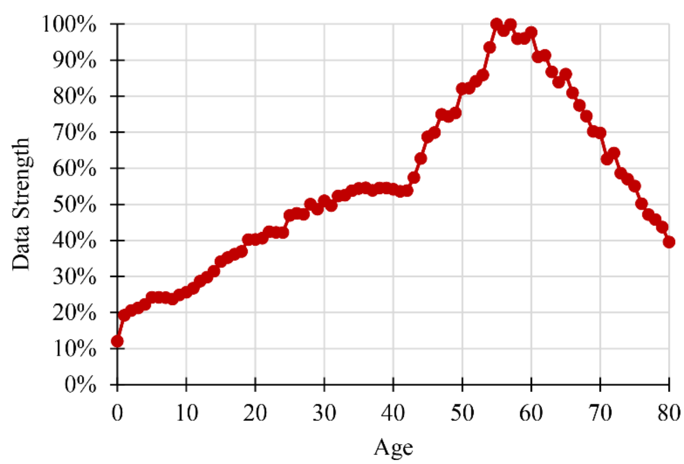

Figure 52. Data Strengths, County Slab

County>Frame

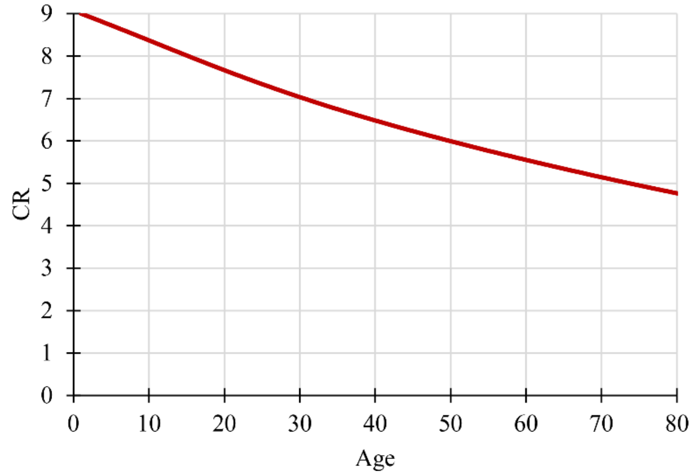

Figure 53. RNO Deterioration Curve, County Frame

Table 18. Transition Matrix, County Frame

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9319	0.0681	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9262	0.0737	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9016	0.0984	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9898	0.0100	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.8602	0.1375	0.0023	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8789	0.0962	0.0248	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0099	0.9801	0.0101
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0099	0.9901
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

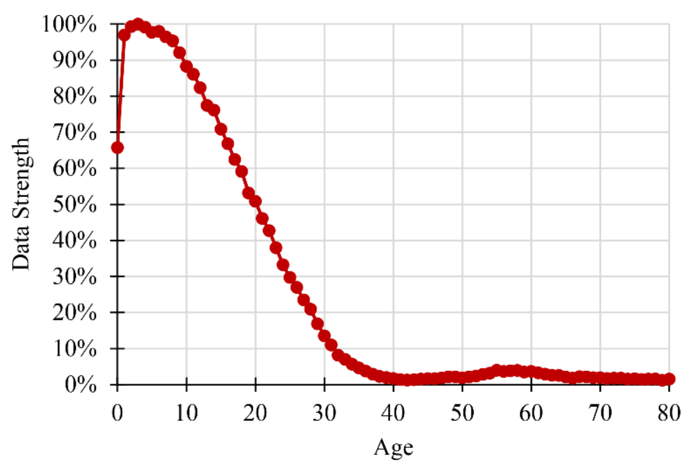

Figure 54. Data Strengths, County Frame

City/Municipality>Stringer PS Beam

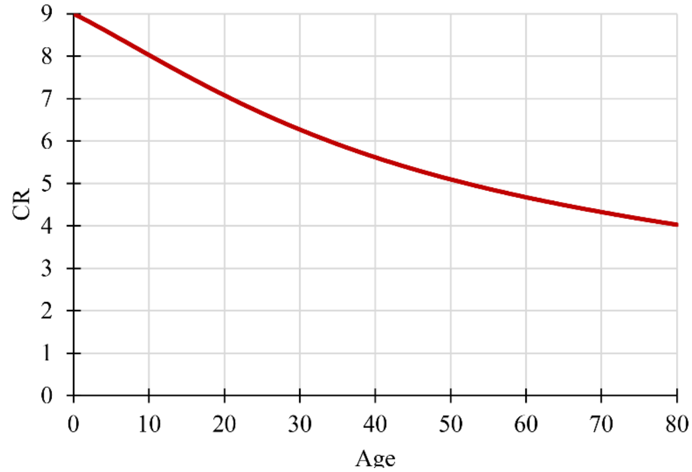

Figure 55. RNO Deterioration Curve, City/Municipality Stringer PS Beam

Table 19. Transition Matrix, City/Municipality Stringer PS Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9093	0.0907	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8791	0.1208	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9114	0.0886	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9180	0.0819	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9915	0.0073	0.0012	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0198	0.9646	0.0156	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9800	0.0100
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9900
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

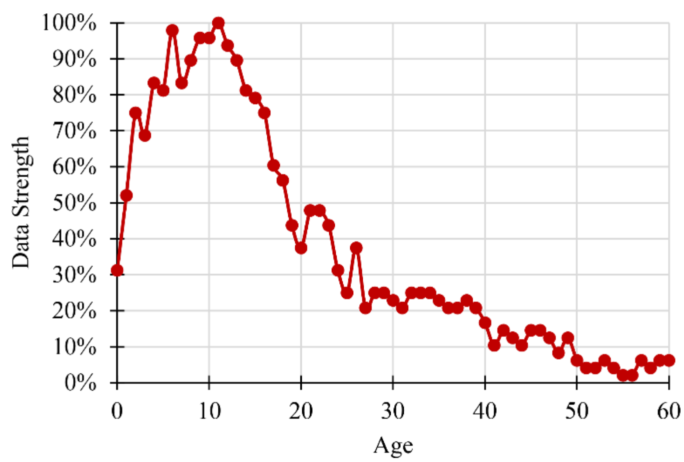

Figure 56. Data Strengths, City/Municipality Stringer PS Beam

<u>City/Municipality>Stringer Steel Beam</u>

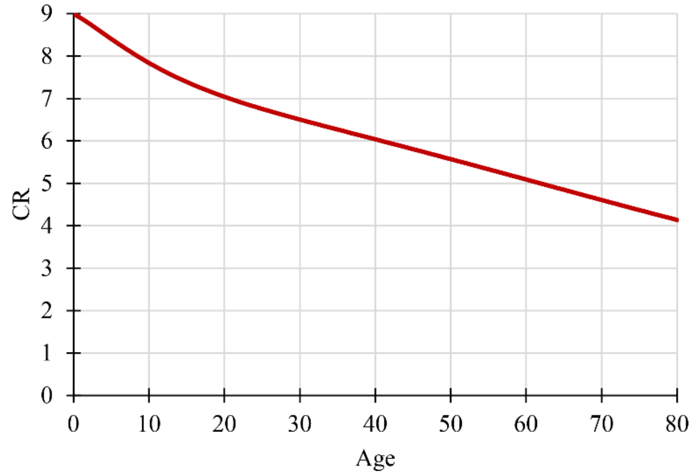

Figure 57. RNO Deterioration Curve, City/Municipality Stringer Steel Beam

Table 20. Transition Matrix, City/Municipality Stringer Steel Beam

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.8930	0.1070	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8008	0.1991	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9638	0.0362	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9735	0.0264	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9652	0.0337	0.0011	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0201	0.9644	0.0155	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9800	0.0100
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9900
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

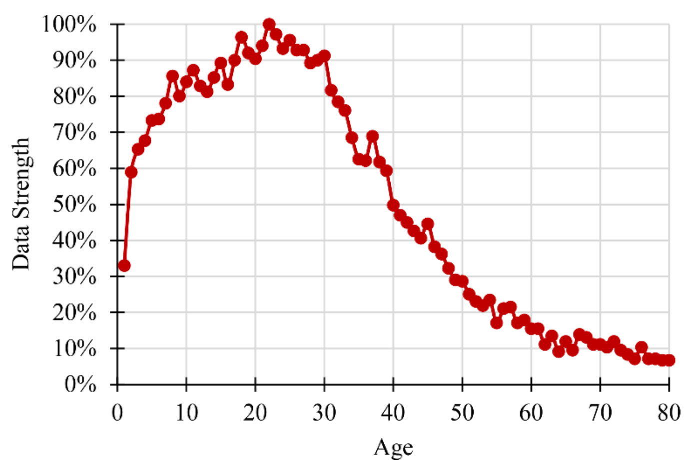

Figure 58. Data Strengths, City/Municipality Stringer Steel Beam

City/Municipality>Box Beam AS WS

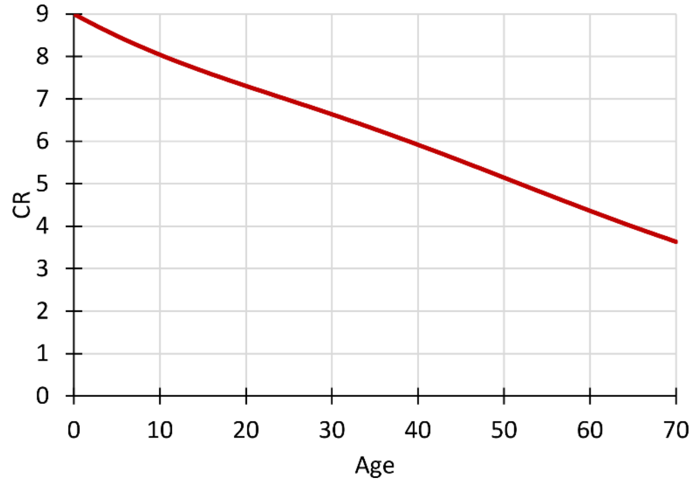

Figure 59. RNO Deterioration Curve, City/Municipality Box Beam AS WS

Table 21. Transition Matrix, City/Municipality Box Beam AS WS

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.8908	0.1091	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9177	0.0822	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9612	0.0388	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9348	0.0651	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.8969	0.1027	0.0004	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8681	0.1299	0.0020	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2197	0.7528	0.0274
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0102	0.9898
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

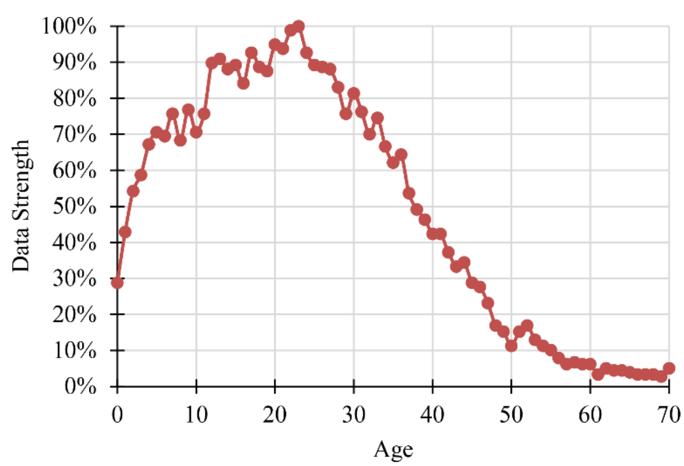

Figure 60. Data Strengths, City/Municipality Box Beam AS WS

City/Municipality>Box Beam Conc Deck

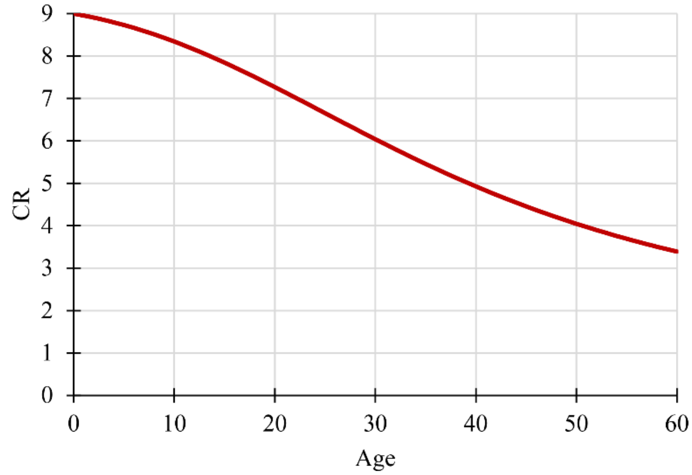

Figure 61. RNO Deterioration Curve, City/Municipality Box Beam Conc Deck

Table 22. Transition Matrix, City/Municipality Box Beam Conc Deck

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9546	0.0454	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.8682	0.1317	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.7529	0.2471	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.8382	0.1617	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.7506	0.2489	0.0005	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.8546	0.1219	0.0235	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9877	0.0036	0.0087
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0101	0.9899
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

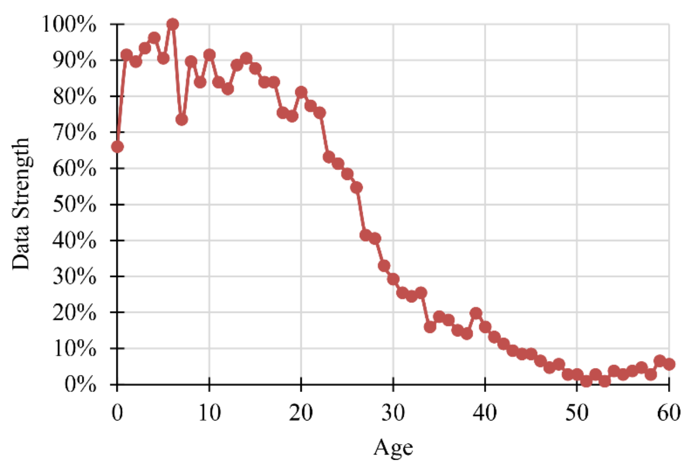

Figure 62. Data Strengths, City/Municipality Box Beam Conc Deck

City/Municipality>Slab

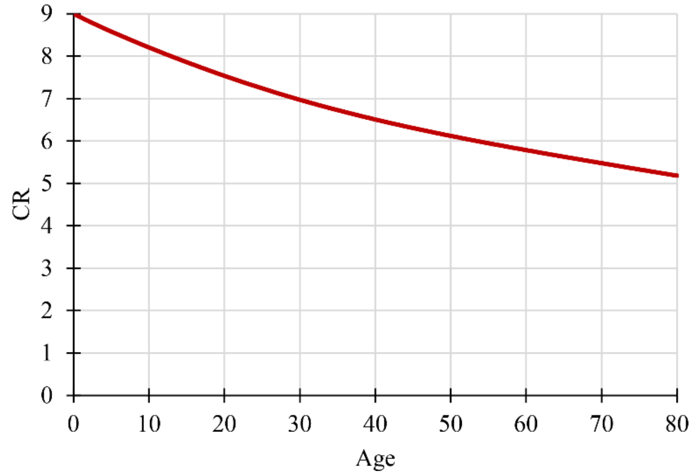

Figure 63. RNO Deterioration Curve, City/Municipality Slab

Table 23. Transition Matrix, City/Municipality Slab

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.9107	0.0893	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9460	0.0539	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9009	0.0991	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9846	0.0153	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.9794	0.0193	0.0014	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.9594	0.0280	0.0125	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9800	0.0100
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.9900
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

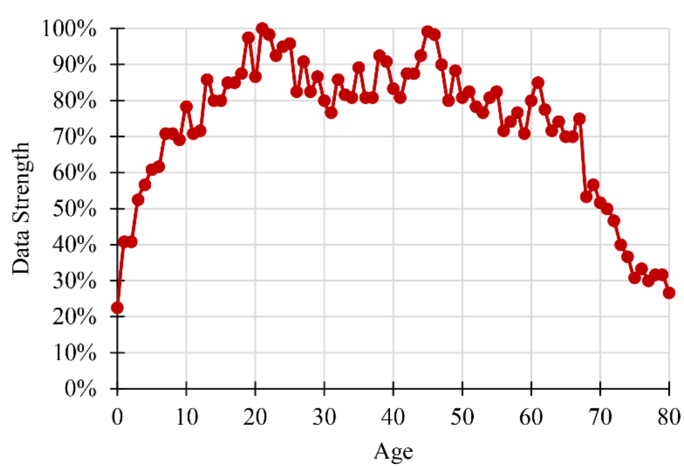

Figure 64. Data Strengths, City/Municipality Slab

City/Municipality>Frame

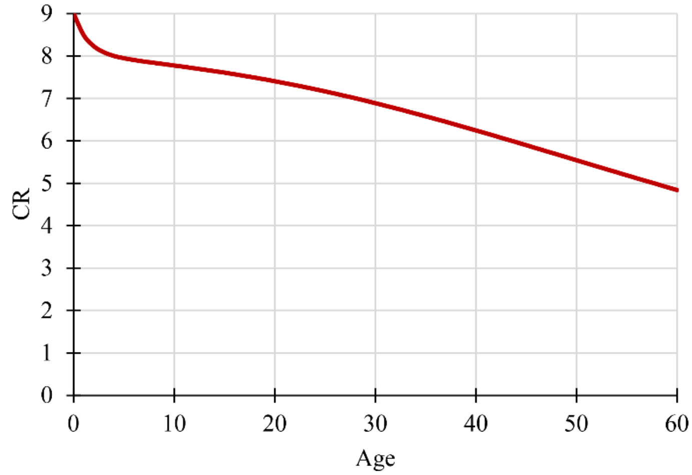

Figure 65. RNO Deterioration Curve, City/Municipality Frame

Table 24. Transition Matrix, City/Municipality Frame

		Condition State								
		9	8	7	6	5	4	3	2	1
Condition State	9	0.4939	0.5061	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	8	0.0000	0.9741	0.0259	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	0.0000	0.0000	0.9567	0.0433	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.9162	0.0837	0.0001	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.8916	0.1040	0.0044	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0104	0.0010	0.9886	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0100	0.0010	0.9890
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0101	0.9899
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

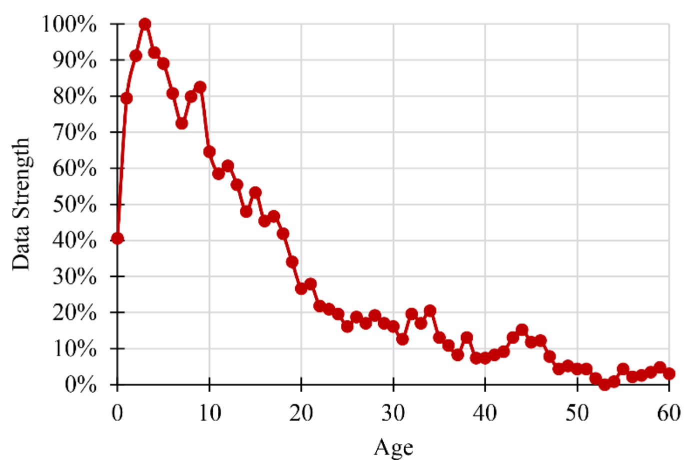

Figure 66. Data Strengths, City/Municipality Frame

7.4.3. Deterioration Curve Comparison among Superstructure Designs

State Superstructures

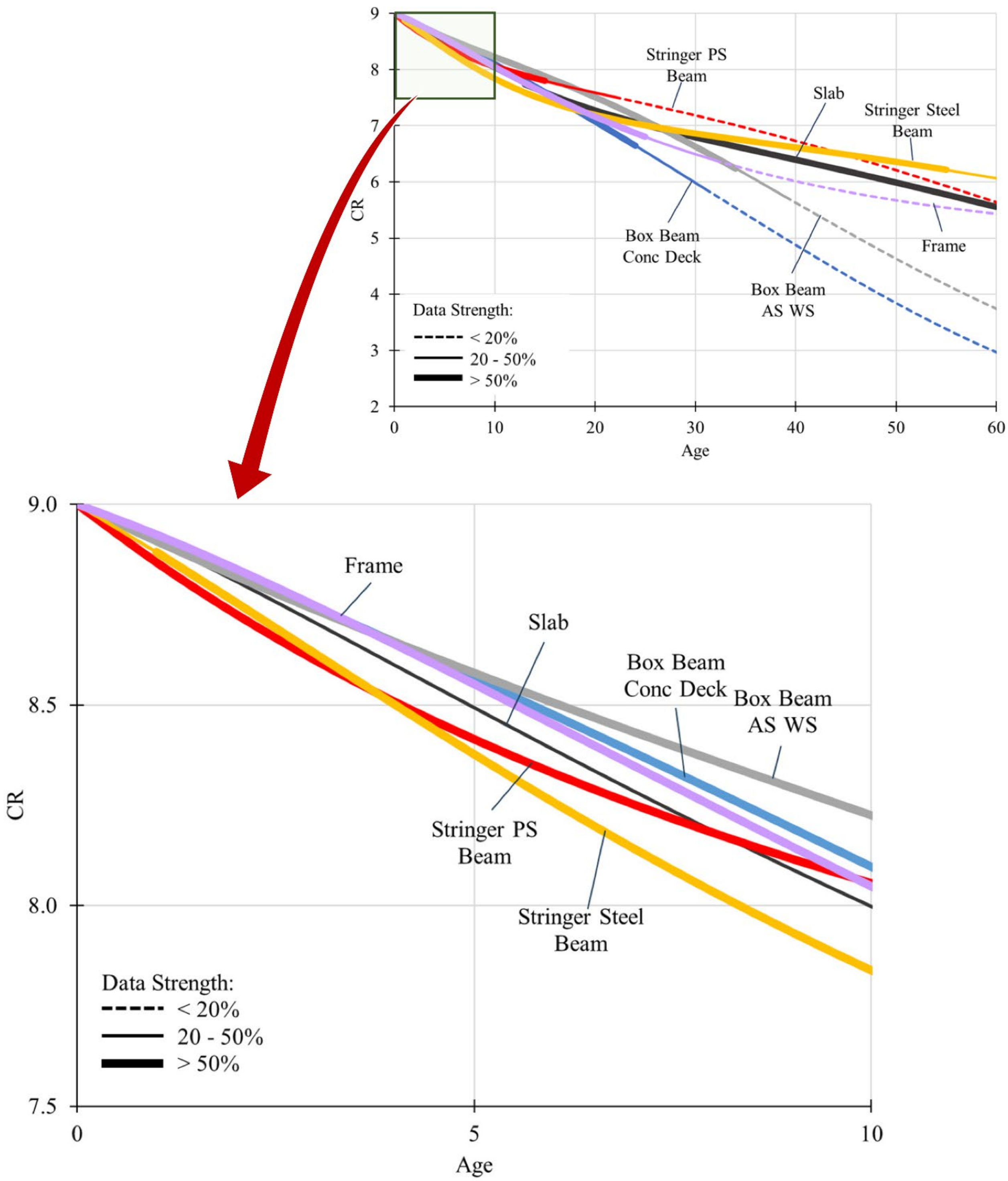

Figure 67. Comparison of State Superstructure Deterioration Curves in the Age Range of 0-10

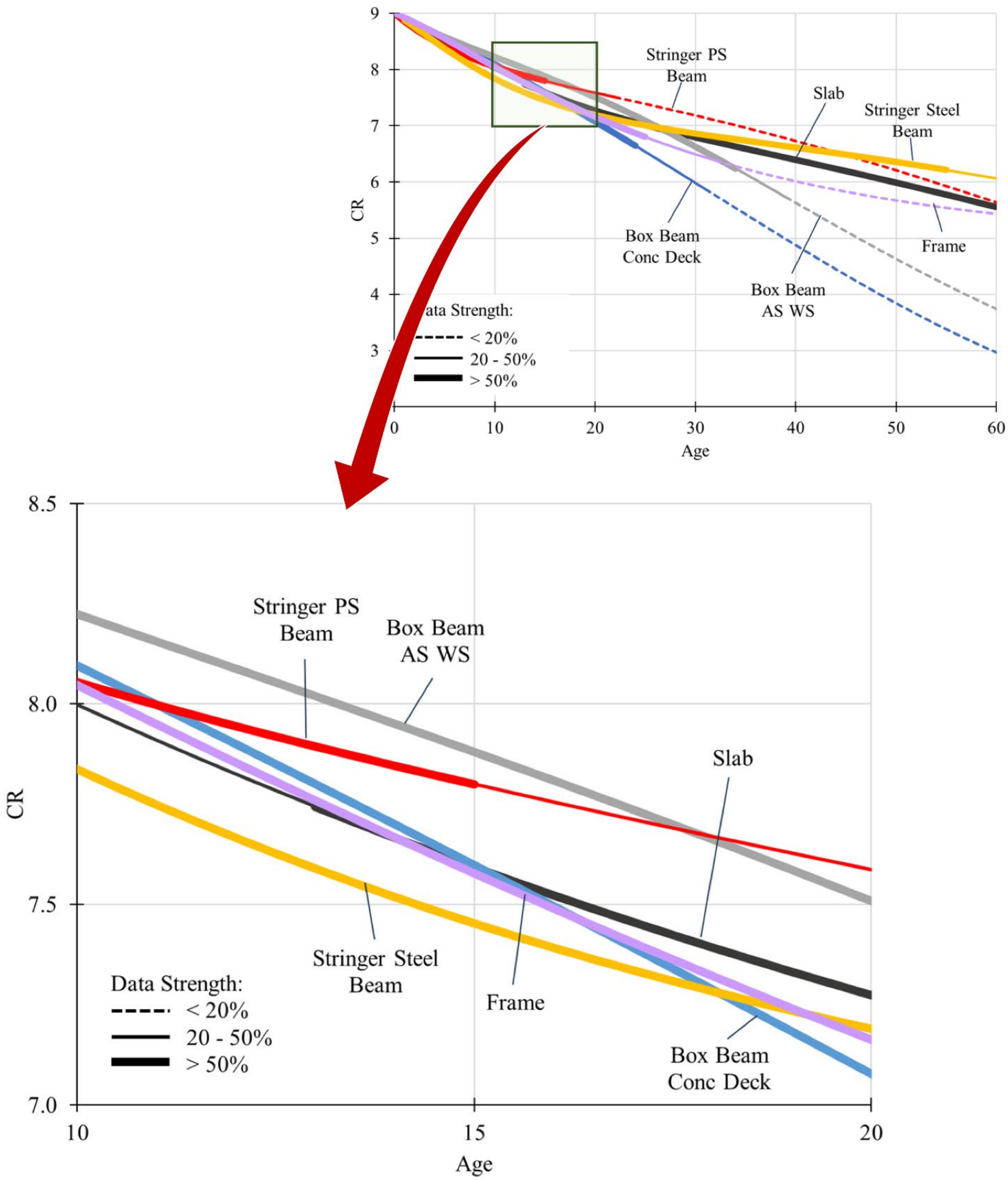

Figure 68. Comparison of State Superstructure Deterioration Curves in the Age Range of 10-20

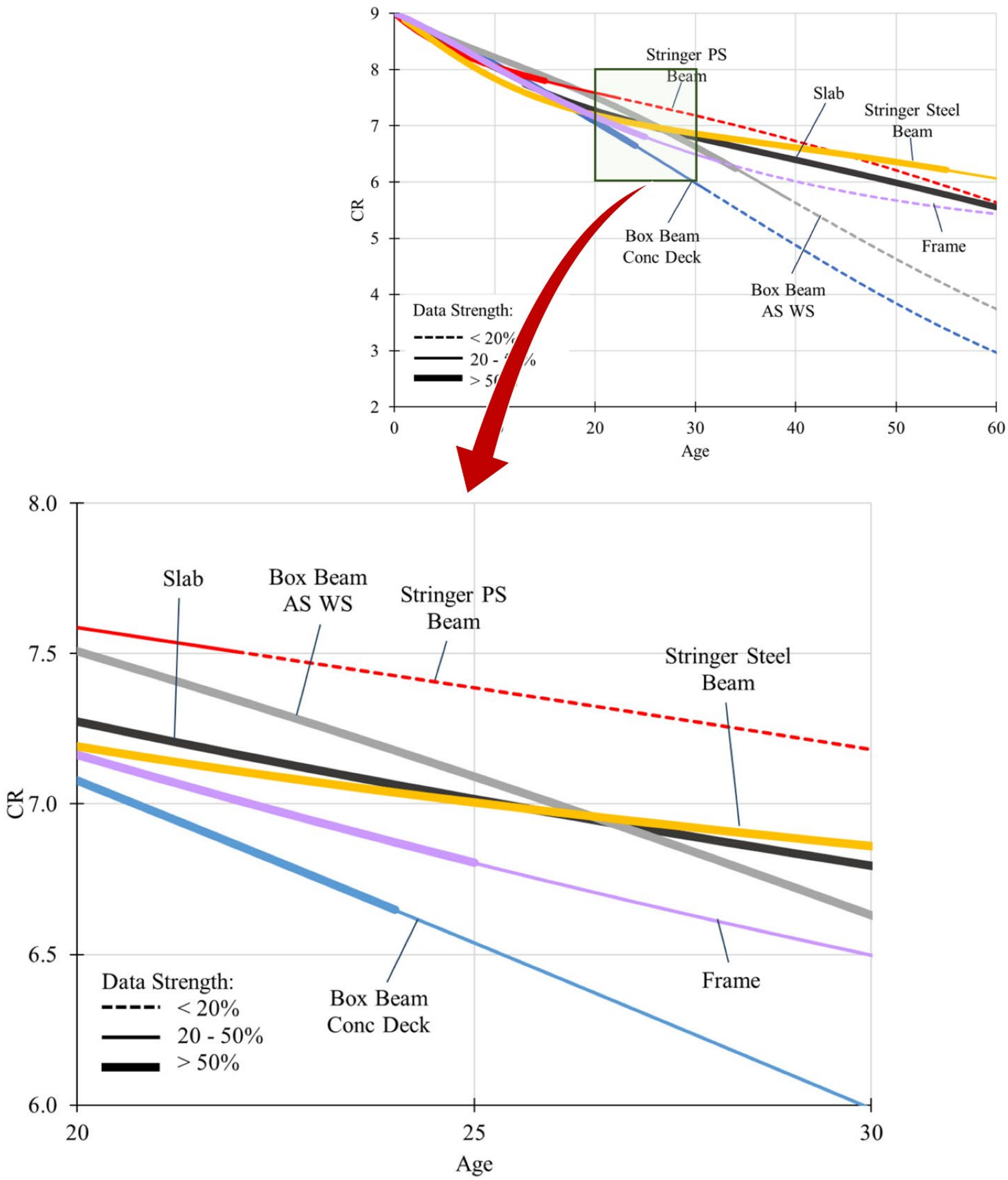

Figure 69. Comparison of State Superstructure Deterioration Curves in the Age Range of 20-30

County Superstructures

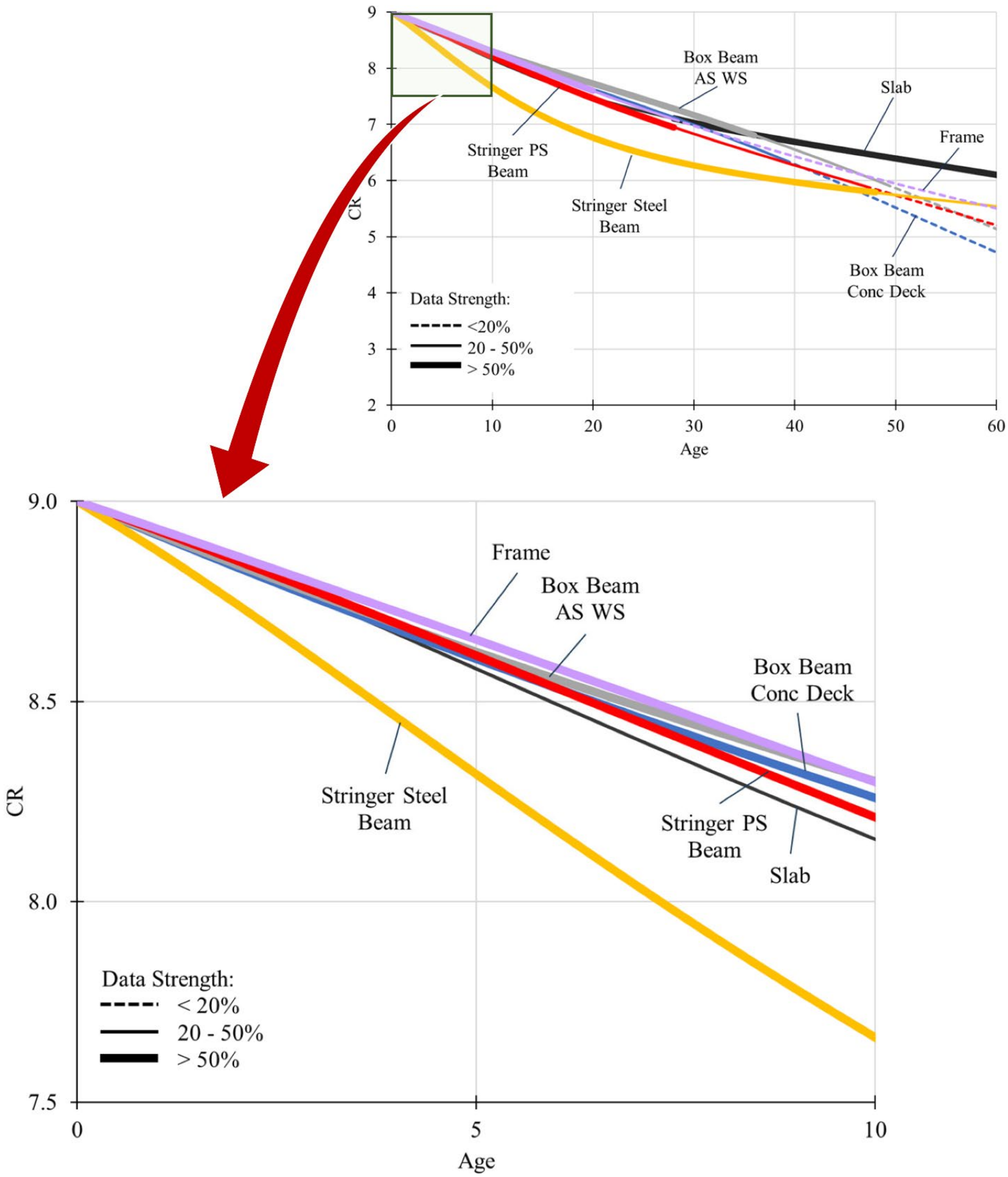

Figure 70. Comparison of County Superstructure Deterioration Curves in the Age Range of 0-10

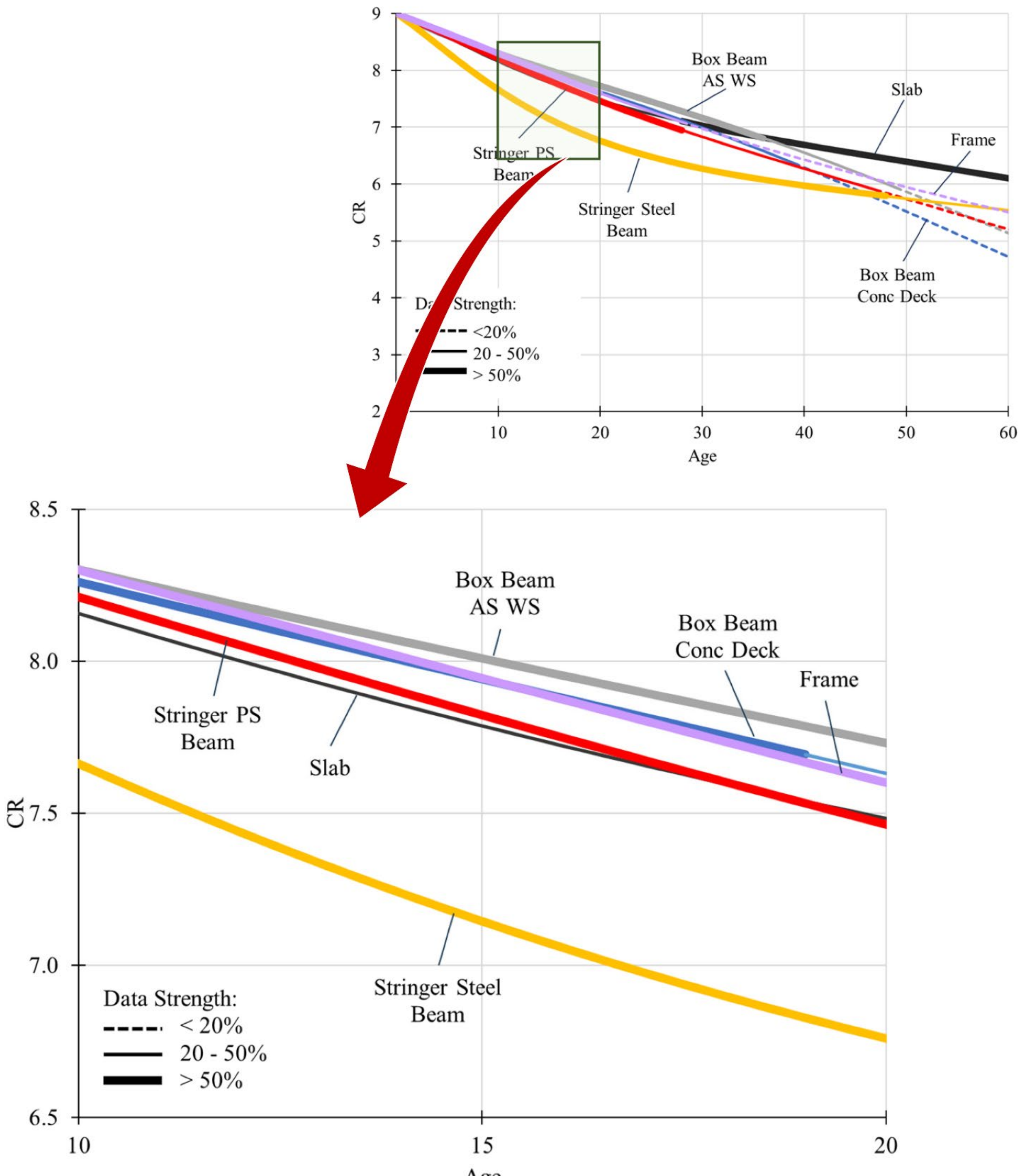

Figure 71. Comparison of County Superstructure Deterioration Curves in the Age Range of 10-20

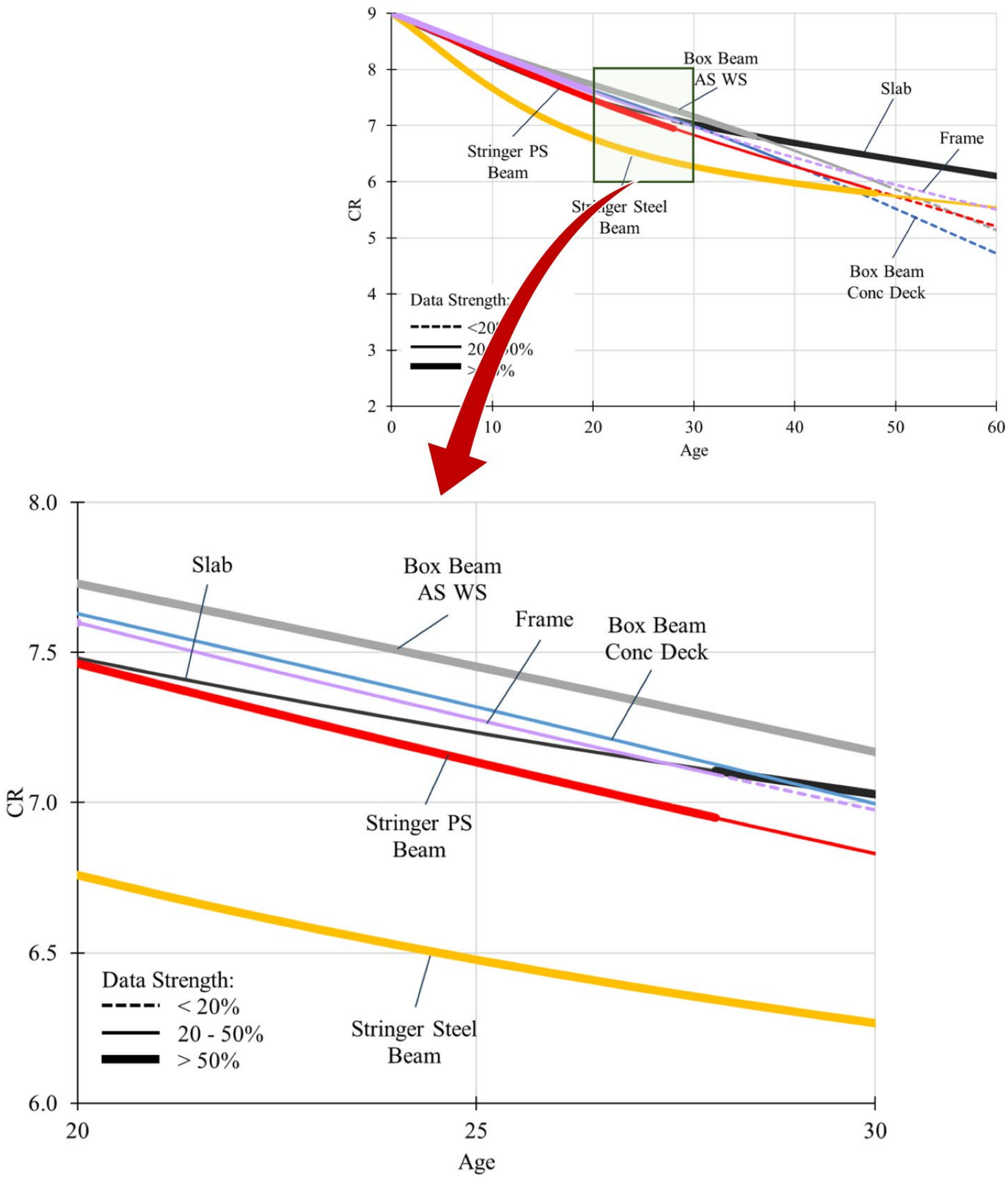

Figure 72. Comparison of County Superstructure Deterioration Curves in the Age Range of 20-30

City/Municipality Superstructures

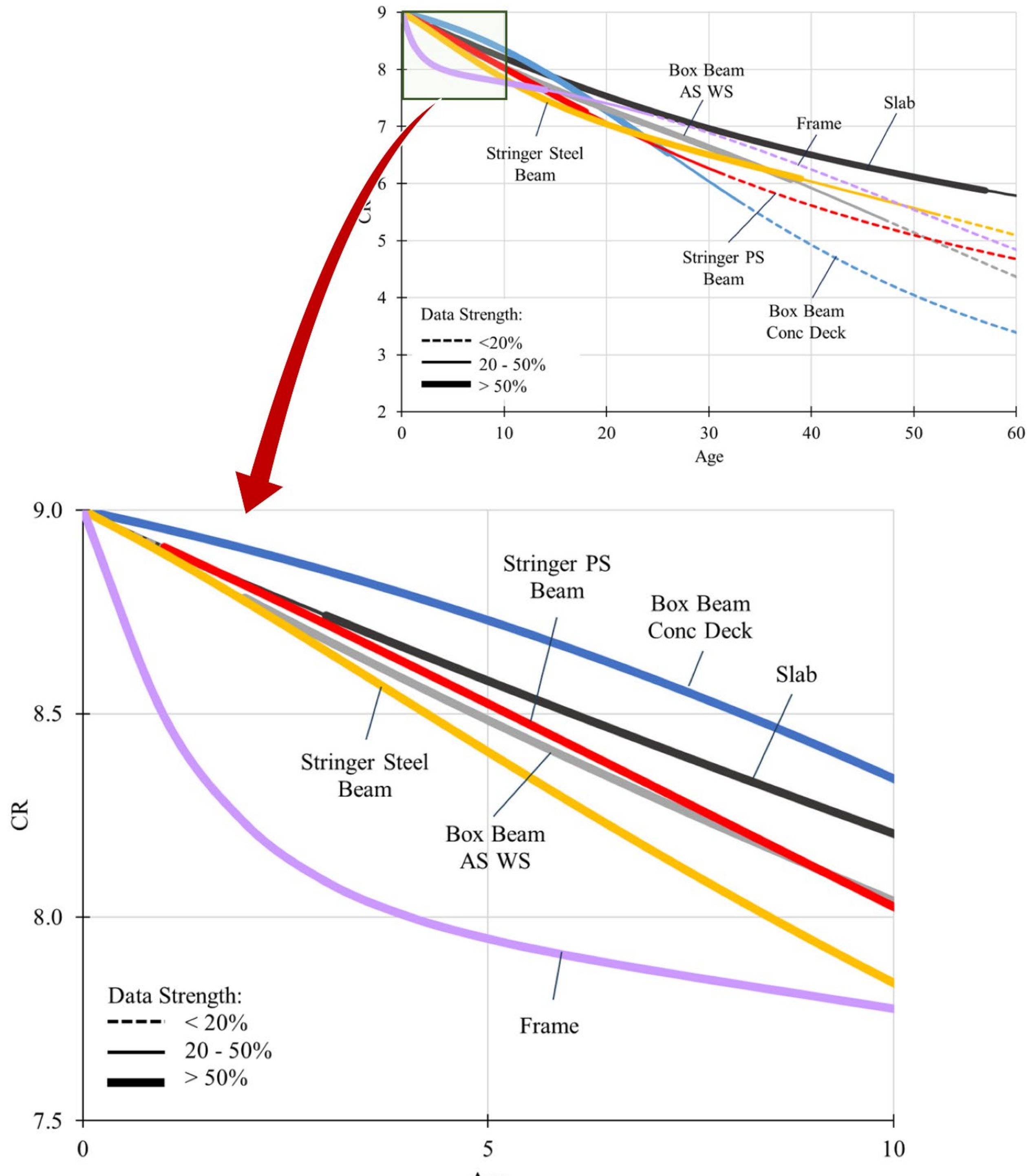

Figure 73. Comparison of City/Municipality Superstructure Deterioration Curves in the Age Range of 0-10

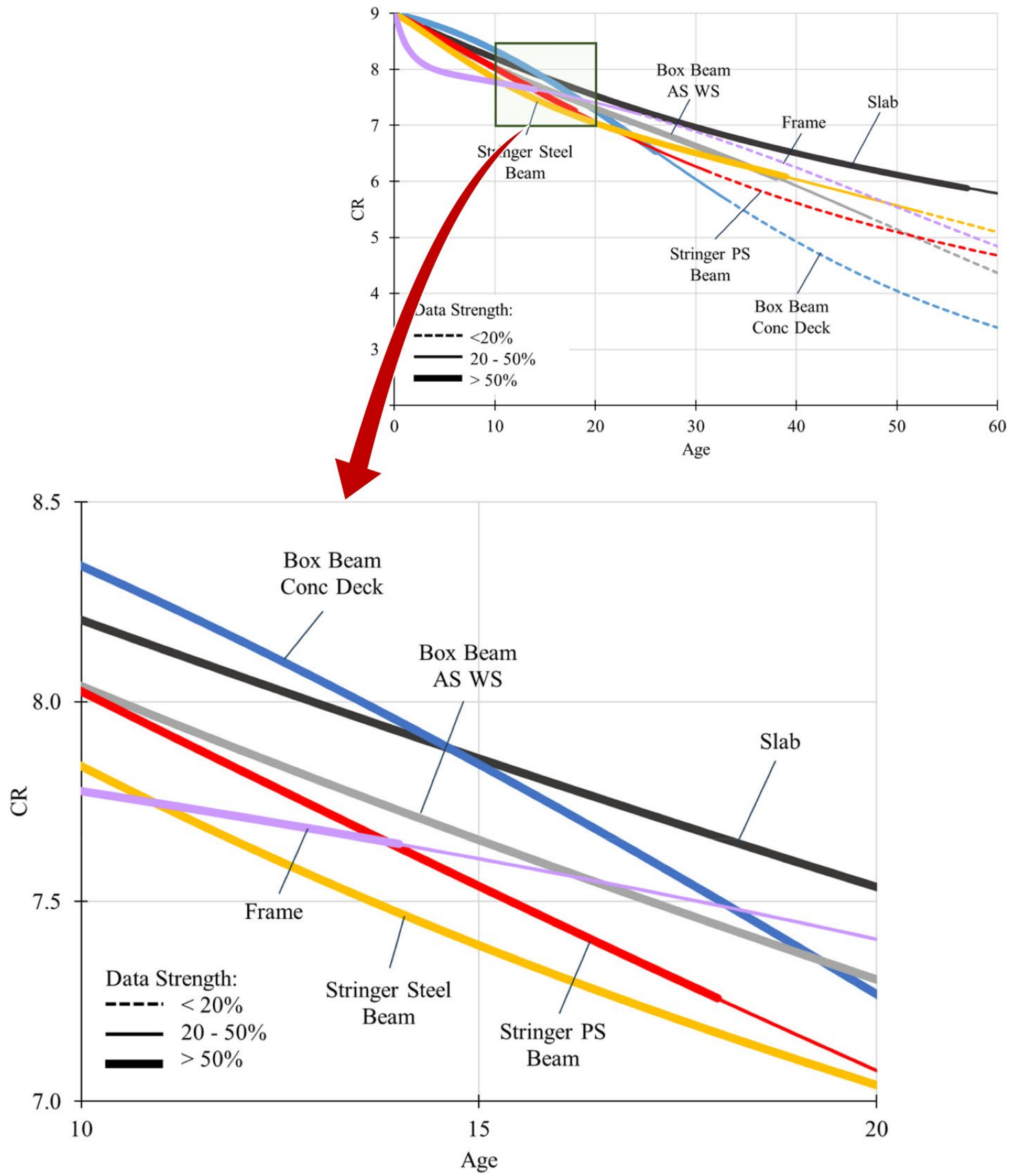

Figure 74. Comparison of City/Municipality Superstructure Deterioration Curves in the Age Range of 10-20

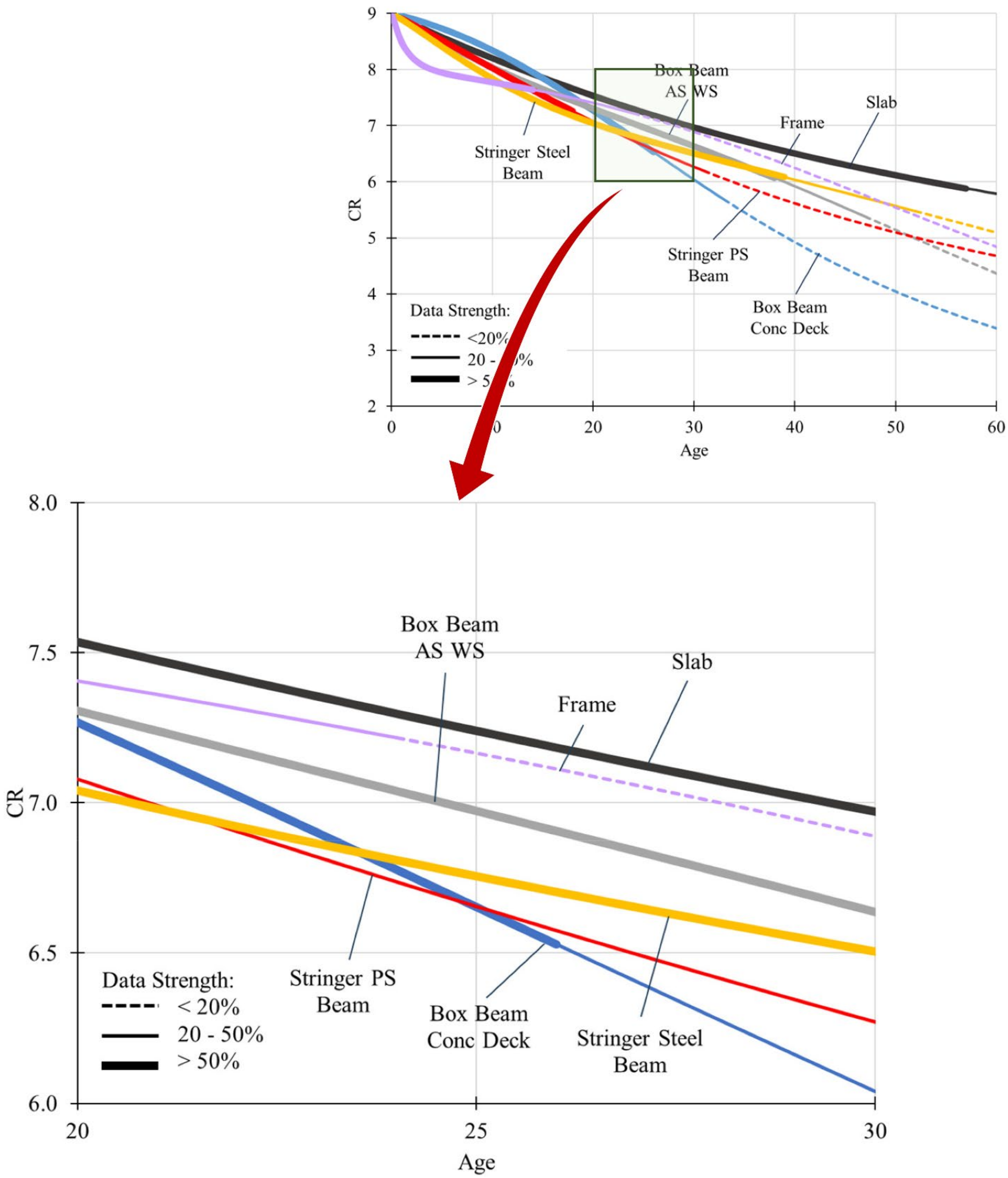

Figure 75. Comparison of City/Municipality Superstructure Deterioration Curves in the Age Range of 20-30

7.4.4. Average Condition Ratings by 5-Year Range

State Superstructures

Table 25. Average Condition Ratings by 5-Year Range over 60 Years, State Superstructures

Age Range	Slab	Box Beam Conc Deck	Box Beam AS WS	Stringer PS Beam	Stringer Steel Beam	Frame
0-5	8.752	8.786	8.783	8.685	8.690	8.785
6-10	8.190	8.287	8.364	8.187	8.041	8.248
11-15	7.748	7.799	8.019	7.897	7.596	7.761
16-20	7.396	7.287	7.661	7.670	7.288	7.324
21-25	7.116	6.756	7.262	7.467	7.074	6.943
26-30	6.881	6.209	6.818	7.264	6.915	6.616
31-35	6.672	5.654	6.337	7.051	6.782	6.338
36-40	6.473	5.100	5.833	6.823	6.659	6.102
41-45	6.274	4.559	5.325	6.577	6.535	5.902
46-50	6.071	4.043	4.826	6.315	6.406	5.735
51-55	5.860	3.565	4.351	6.039	6.270	5.594
56-60	5.641	3.131	3.907	5.752	6.126	5.477
61-65	5.415	2.746	3.501	5.459	5.974	5.378

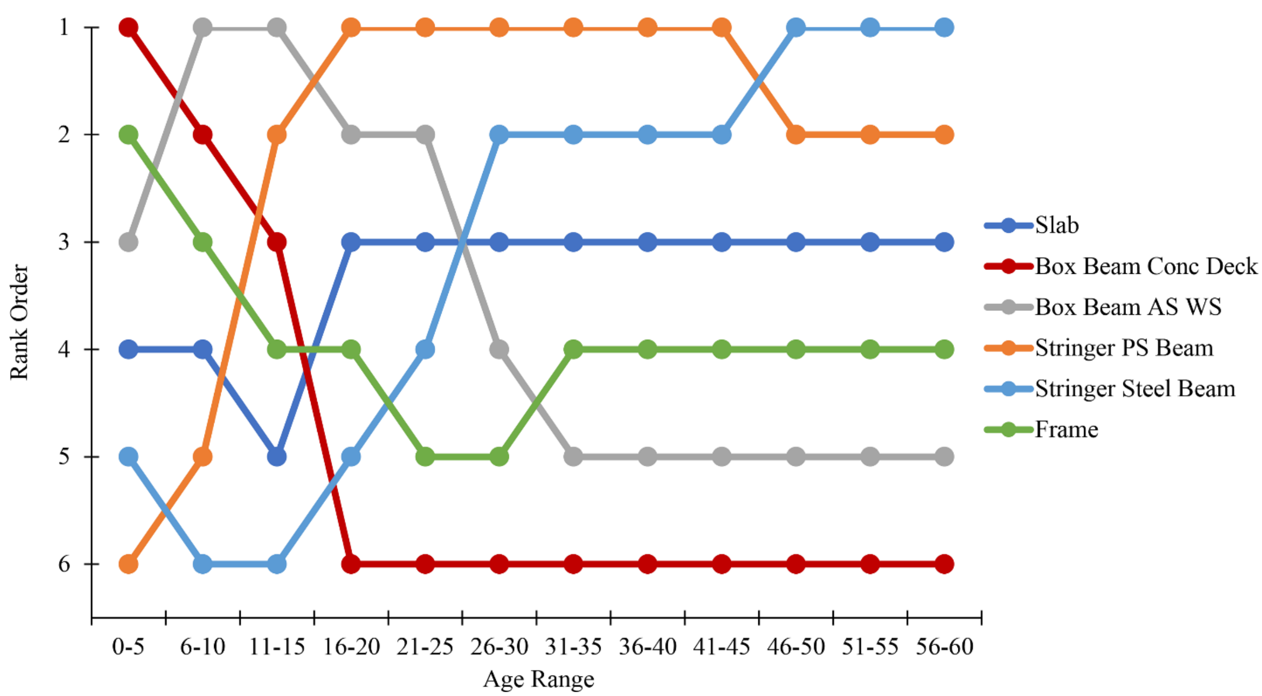

Figure 76. State Superstructures Ranked by 5-Year Average Condition Ratings

County Superstructures

Table 26. Average Condition Ratings by 5-Year Range over 60 Years, County Superstructures

Age Range	Slab	Box Beam Conc Deck	Box Beam AS WS	Stringer PS Beam	Stringer Steel Beam	Frame
0-5	8.796	8.799	8.808	8.811	8.667	8.828
6-10	8.324	8.395	8.427	8.372	7.915	8.442
11-15	7.929	8.066	8.123	7.976	7.339	8.085
16-20	7.599	7.754	7.841	7.604	6.903	7.736
21-25	7.330	7.445	7.565	7.263	6.583	7.404
26-30	7.107	7.127	7.284	6.950	6.346	7.093
31-35	6.919	6.793	6.991	6.658	6.165	6.804
36-40	6.752	6.439	6.681	6.380	6.021	6.534
41-45	6.599	6.066	6.353	6.110	5.898	6.281
46-50	6.452	5.677	6.008	5.843	5.787	6.041
51-55	6.308	5.279	5.651	5.577	5.682	5.813
56-60	6.162	4.880	5.287	5.309	5.580	5.595

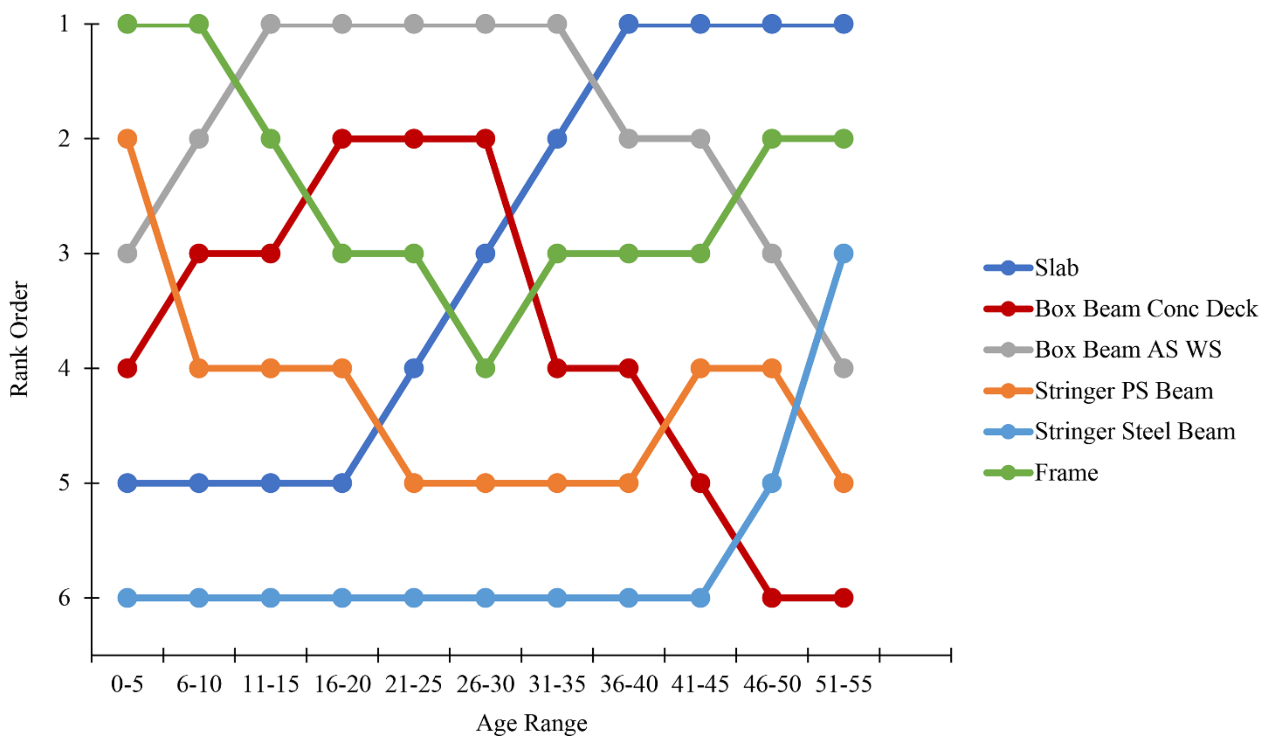

Figure 77. County Superstructures Ranked by 5-Year Average Condition Ratings

City/Municipality Superstructures

Table 27. Average Condition Ratings by 5-Year Range over 60 Years, City/Municipality Superstructures

Age Range	Slab	Box Beam Conc Deck	Box Beam AS WS	Stringer PS Beam	Stringer Steel Beam	Frame
0-5	8.786	8.872	8.737	8.765	8.710	8.294
6-10	8.353	8.506	8.212	8.225	8.057	7.840
11-15	7.995	8.050	7.805	7.731	7.561	7.677
16-20	7.663	7.503	7.443	7.259	7.174	7.490
21-25	7.357	6.901	7.105	6.820	6.866	7.265
26-30	7.077	6.284	6.771	6.420	6.603	7.002
31-35	6.823	5.687	6.427	6.060	6.363	6.705
36-40	6.593	5.134	6.067	5.737	6.131	6.381
41-45	6.384	4.639	5.691	5.449	5.900	6.038
46-50	6.193	4.205	5.302	5.192	5.666	5.685
51-55	6.016	3.829	4.908	4.962	5.428	5.329
56-60	5.849	3.507	4.517	4.754	5.187	4.979

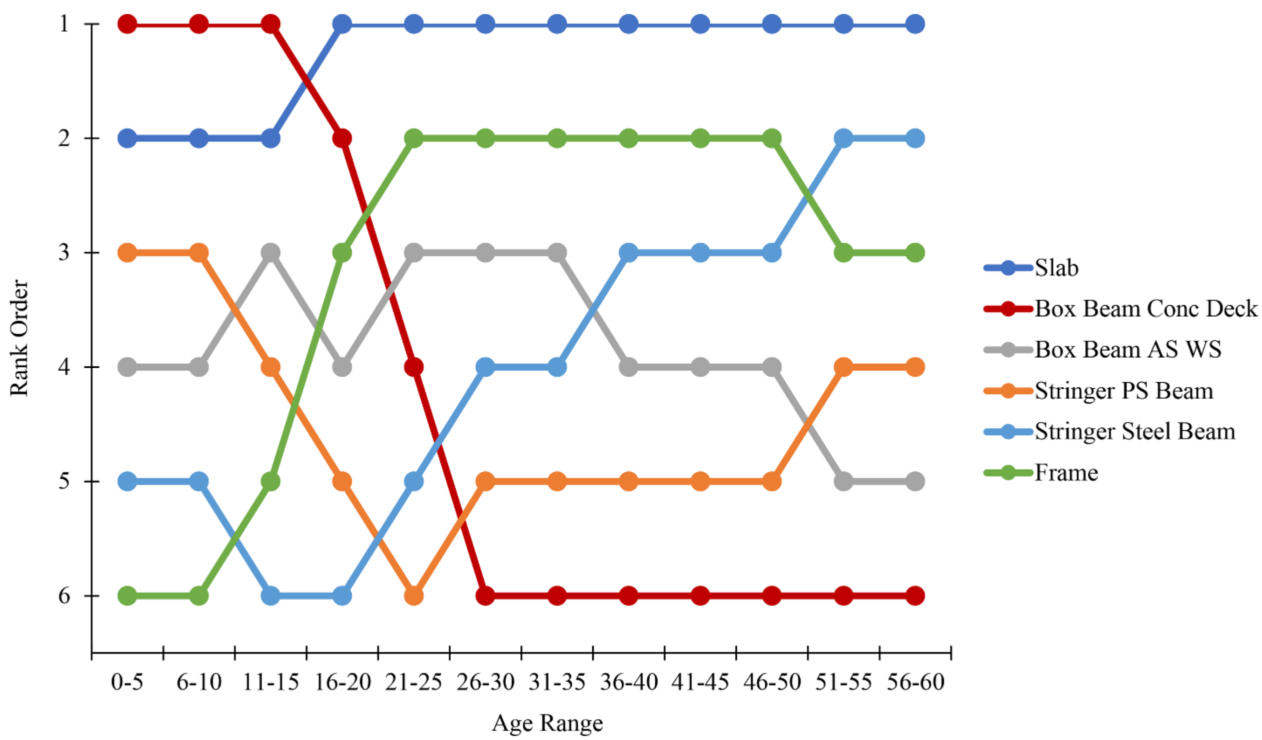

Figure 78. City/Municipality Superstructures Ranked by 5-Year Average Condition Ratings

7.4.5. Deterioration Curves between Box Beam AS WS and Box Beam Conc Deck

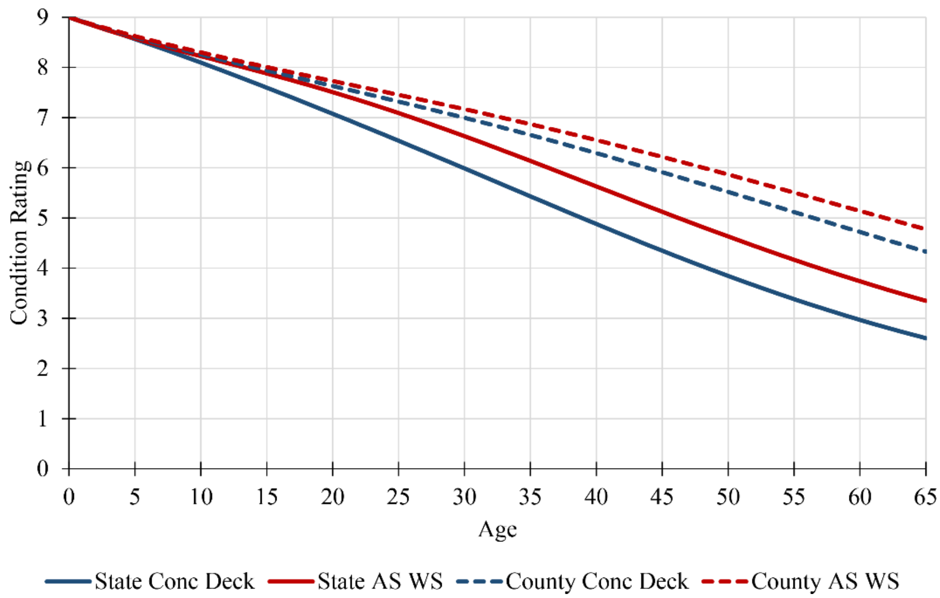

Figure 79. Deterioration Curves of Box Beam AS WS vs. Conc Deck

Table 28. ADT for State and County Box Beam AS WS and Conc Deck

Maintenance Responsibility	Superstructure Design	ADT		
		Average	Maximum	Minimum
State	Box Beam AS WS	3,896	156,804	86
	Box Beam Conc Deck	6,986	126,679	86
County	Box Beam AS WS	963	35,234	0
	Box Beam Conc Deck	2,272	40,281	10

Table 29. Average Deck Areas for State and County Box Beam AS WS and Conc Deck

Maintenance Responsibility	Superstructure Design	Average Deck Area (ft^2)
State	Box Beam AS WS	2,643.57
	Box Beam Conc Deck	5,189.96
County	Box Beam AS WS	1,512.05
	Box Beam Conc Deck	2,553.99