Project Rationale

For this project, I analyzed the World Bank's Logistics Performance Index (LPI) from 2007–2023 to understand how global logistics has changed over the last sixteen years. My main question was simple: is the world converging or diverging in logistics performance? My working thesis ended up being: technology accelerates, geography constrains, and bureaucracy stalls progress.

This topic matters because logistics shapes almost everything we care about in development and global trade. Countries with strong logistics grow faster, trade more, and integrate more into the global economy. This connects directly to policy, international business strategy, and the broader question of economic equity.

From a STAT 3280 perspective, this project let me pull together everything we learned: design principles, clean storytelling, and multiple visualization modalities. I used a mix of R, ggplot2, Plotly, Shiny, gganimate, and one Tableau map to show that I can combine static, interactive, animated, and geospatial tools in a single narrative.

I structured the project like a small documentary. The first section sets up the LPI and why it matters. From there, the visualizations move in a logical arc: what the world looks like today, how regions trend over time, what components actually drive those trends, and then into more exploration tools where the user can compare countries directly. The last pieces move into animation and geography, which show why some countries rise and others stagnate.

Visualization 1 is a histogram of 2023 LPI scores. This establishes the baseline: most countries fall between 2.5 and 3.5. I used a simple binwidth, neutral colors, and limited gridlines to keep it readable. This sets up the whole idea of a "logistics middle class" and why the tails matter.

Visualization 2 is an interactive Plotly line chart of regional trends. Interactivity matters here because there are only seven survey years, and hover text helps the user explore without clutter. The story is pretty clear: Europe stays on top, Asia rises, and Africa stays flat.

Visualization 3 breaks the index into its six components. Using six lines in one plot risks clutter, but I wanted the reader to be able to compare components directly on the same scale. The main insight is that tracking and infrastructure have improved the most, while customs procedures lag badly.

Visualization 4 is a horizontal bar chart ranking component improvements. This is the "summary" version of the previous plot.

Visualization 5 is a Shiny leaderboard that lets the user filter by region and year. Without interactivity, this would require 35 separate charts. This tool makes it easy to see which countries are rising stars and which ones fall behind.

Visualization 6 is a Shiny comparison tool that lets the user benchmark any country against its region across all components. This exposes some surprising outliers like Rwanda and Vietnam, and also underperformers like Venezuela.

Visualization 7 is an animated scatterplot showing the relationship between infrastructure and LPI over time. This is one of the clearest "causal feeling" visuals in the project. As infrastructure scores rise, LPI scores rise. Asia moves steadily upward; Africa barely moves.

Visualization 8 is a Tableau choropleth map showing global LPI values. The spatial patterns jump out immediately: coastal nations tend to perform better, and major trade hubs outperform everyone. Tableau was the right tool here since geospatial visuals in ggplot require much more manual setup.

I used consistency across all eight visuals through one color palette for regions, one for components, applied everywhere. Everything uses a minimal theme bold titles, smaller subtitles, and readable axis text.

Color accessibility was a major focus. I tested both palettes through a color-blindness simulator and avoided red-green contrasts. All subtitles provide interpretation, not repetition, and captions serve as alt-text so screen readers can still follow the narrative.

For interaction, the Shiny apps use simple controls and clear instructions. The design avoids unnecessary complexity unless the user chooses to engage with it. This keeps the cognitive load manageable and matches the progressive disclosure ideas from class.

The data wrangling relied heavily on purrr, dplyr, and tidyr to pull each year's sheet into one tidy dataset. Some countries required manual region adjustments. For Shiny, I used cascading dropdowns, reactive filtering, and animated sliders. The animation uses gganimate with a linear easing so the transitions look natural.

The biggest technical challenge was reproducibility. Shiny doesn't knit to HTML, and animations require the GIF to live in the same folder as the Rmd. Tableau is also just a screenshot in the final output. I handled these by explaining the limitations up front.

There are several limitations worth mentioning. The LPI relies on surveys, which means perceptions may not perfectly match reality. The data is also national, so it misses differences within countries.

In the future, expanding this project to city-level data or to freight cost metrics could give a more realistic picture of how trade actually works. There's also potential for machine learning models that forecast which countries will rise based on current reforms or investments.

Overall, this project showed that global logistics performance has improved, but unevenly. Wealthy and coastal regions continue to dominate, Asia is catching up fast, and Africa is stuck.

Putting this together through eight different visualizations forced me to think in terms of narrative. Each plot answered a different slice of the main question, and the interactive and animated tools helped bring the dynamics to life. The final picture is that logistics is both a technical and geographic system, one where the gap between leaders and laggards is narrowing slowly, but not as much as global trade would suggest.

Sources: World Bank. (2023). *Logistics Performance Index (LPI)*. Retrieved from

https://lpi.worldbank.org/

AI Prompts:

"Collect all my prompts from the entire thread. Remove redundancies to minimize clutter.

"I have a multi-sheet Excel file but read_excel() isn't loading. Can you remind me how installation vs. library loading works so I understand why the function isn't available?"

"My R code is only reading the first sheet. Can you explain how excel_sheets() works and why iterating over sheets is the clean way to import all of them?"

"After importing all sheets, I want to add each sheet's year as a new column. Which part of the mapping pipeline is best for inserting that year information?"

"My pivot_longer isn't working because column names are messy. What does clean_names() do and why does it improve consistency?"

"In my histogram, I accidentally put a string inside aes(). Why does that make ggplot treat the variable as discrete instead of referencing the column?"

"My histogram throws a stat_bin() error about discrete x aesthetics. Why does this happen and how does ggplot decide whether a variable is continuous?"

"For my region line chart, how do named palettes map to factor levels to keep colors consistent?"

"How do I recode long variable names into readable labels without breaking the underlying data structure?"

"Why does naming a vector 'palette' sometimes conflict with the base R palette() function, and how can I avoid naming conflicts?"

"How do I control legend titles, labels, keys, and spacing so all my plots have clean, consistent legends?"

"How should I calculate growth rates for each LPI component between 2007 and 2023 using grouped data in a reliable way?"

"How do I make sure each visualization adds something new so the project doesn't include redundant plots?"

"How does the basic architecture of a Shiny app work, especially how reactive expressions feed outputs when inputs change?"

"For my Shiny bar chart, where should the reactive filtering and plotting live inside the server so it updates correctly?"

"My Shiny plot isn't updating when the region changes. How do reactive dependencies work conceptually?"

"I need a dependent dropdown (region to country). When is renderUI() better than updateSelectInput() conceptually?"

"For my second Shiny app, how do I compute component-level growth only when inputs change instead of recalculating everything?"

“How can I keep the visual theme consistent between Shiny plots and my static ggplots?”

“How should I choose between sidebarLayout, fluidPage, and other UI structures to keep the Shiny app usable?”

“For my animation, how do I compute ranks per year and use transition_time() so movement looks smooth and understandable?”

“My Tableau installation doesn’t show Share. Why are screenshots acceptable for an R Markdown HTML document, and how do I insert them?”

“What’s the correct way to include images in R Markdown when paths contain spaces or come from Windows folders?”

“Is HTML the right output format since Shiny and gganimate won’t work properly in a PDF?”

“Do my nine visuals meet the assignment requirements without redundancy?”

“Does my narrative—distribution → geography → region trends → component trends → growth → country comparisons → animation—flow logically?”

“How do theme elements (titles, axis text, legends, margins, colors, gridlines) work so I can apply one consistent visual theme?”

“What are general strategies for simplifying long dplyr pipelines to keep them readable and efficient?”

“How can I rewrite my component-trends pipeline to avoid creating too many intermediate objects?”

“What are the key conceptual debugging steps to ensure my Shiny server logic and reactivity are structured correctly?”

“How do I choose frame rate, duration, and animation settings that keep the animated plot readable and not chaotic?”

“How can I list all the places where color decisions happen so I can keep my palette consistent?”

“Is using .groups = 'drop' the right way to stop leftover grouping from causing issues later in Plotly?”

“How do I make a Plotly line chart with type = 'scatter', mode = 'lines+markers', and hover text showing region, year, and avg LPI?”

“How does Plotly match my named region_palette to factor values so colors stay aligned?”

“What’s the cleanest way to format multi-line Plotly titles and use <sub> for a subtitle?”

“Why is across() better than manually summarizing each LPI component column?”

“Why is pivot_longer(-year, ...) safe, and when might I need to list columns instead?”

“When I use recode(..., !!!component_labels), what does !!! actually do in simple terms?”

“How should I think about choosing line widths and point sizes for multi-series line charts?”

“What assumptions am I making when I grab the 2007 and 2023 values, and how can I make this safer if more years get added?”

“How do I decide where to place labels in flipped bar charts so they don’t overlap?”

“How does Shiny run slider animations, and why do I need to snap the slider to the closest real survey year?”

“What edge cases should I consider when selecting the top N countries?”

“How does expand_limits(y = 5) work with the LPI scale, and when is it better than manually setting limits?”

“When does Shiny rebuild UI elements created with renderUI()?”

“Why is long format with a ‘type’ column the best structure for comparing country vs. regional data?”

“Is using the min and max year safe when countries might have missing or irregular years?”

“What other ways can I highlight the country besides alpha, and why is alpha a good choice?”

“How does gganimate handle transitions between discrete years, and what does ease_aes('linear') control?”

“How do fps and nframes interact, and how do I choose values that balance smoothness and file size?”

“Why should I include the saved GIF with include_graphics() instead of running animate() inside the R Markdown?”

“Why are relative paths better than absolute ones when inserting images like my Tableau screenshot?”

“What do input$ and output$ actually mean in Shiny, and why do we use the dollar sign in the first place?”

“How do I write the code so a selectInput for region actually filters the data with filter(region == input$region) inside renderPlot?”

“How do I use a sliderInput for year and plug input$year into my dplyr pipeline correctly?”

“What does reactive() really return, and why do I need parentheses when I call a reactive data frame?”

“What does req(input$region) do at the top of renderPlot, and why does it stop errors?”

“How do I update a country dropdown based on the selected region using updateSelectInput()?”

“How do I use aes(x = .data[[input$x_var]]) when the user picks the x-variable from a dropdown?”

“What does isolate(input$year) actually change in how the plot updates?”

“How do I make the plot only re-run when the user clicks an action button using eventReactive?”

“When I get ‘object not found’ inside renderPlot, what input$/reactive mistakes should I check first?”