Cattle Movement

Tanin Rajamand

2024-08-09

before we begin:

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tibble)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(stringr)
library(RColorBrewer)
library(wesanderson)
```

```
## Warning: package 'wesanderson' was built under R version 4.3.3
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(readxl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v readr   2.1.5
## v purrr   1.0.2     v tidyr   1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.3.3
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.3.3
```

```
## Linking to GEOS 3.11.2, GDAL 3.8.2, PROJ 9.3.1; sf_use_s2() is TRUE
```

```
library(maps)
```

```
## Warning: package 'maps' was built under R version 4.3.3
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
print(getwd())
```

```
## [1] "C:/Users/tanin/OneDrive/Documents/Lab R/Cattle movement/Data Presentation"
```

```
# Set the working directory (this is already set in your case)
setwd("C:/Users/tanin/OneDrive/Documents/Lab R/Cattle movement/Data Presentation")

# Load the dataset using just the file name since the working directory is already set correctly
movements_data <- read_csv("st_full_dairy.csv")
```

```
## Rows: 1146 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): spp, type, org_st, dest_st
## dbl (1): nb_shp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

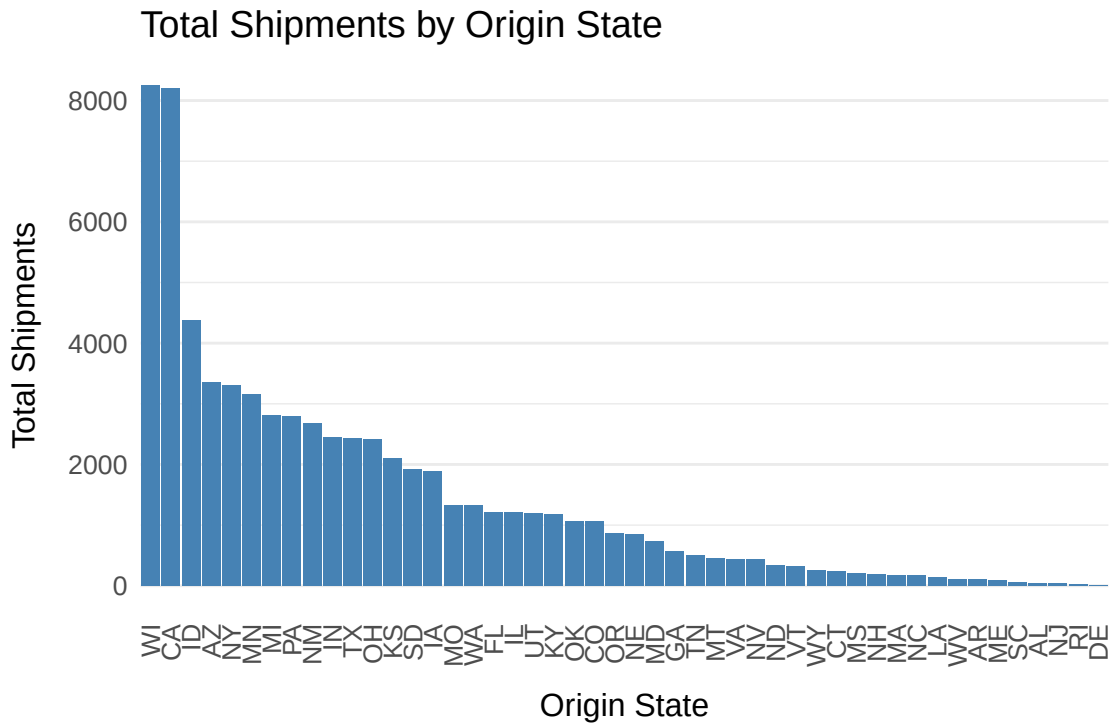
```
colnames(movements_data)[colnames(movements_data) == "nb_shp"] <- "Counts of Shipment"
colnames(movements_data)[colnames(movements_data) == "org_st"] <- "Origin"
colnames(movements_data)[colnames(movements_data) == "dest_st"] <- "Destination"

colnames(movements_data)
```

```
## [1] "spp"                "type"               "Counts of Shipment"
## [4] "Origin"             "Destination"
```

I want to make a graph that shows me the states of origin based on counts - regardless of the destination state

```
origin_summary <- movements_data %>%
  group_by(Origin) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  arrange(desc(Total_Shipments))


ggplot(origin_summary, aes(x = reorder(Origin, -Total_Shipments), y = Total_Shipments)) +
```

```
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Total Shipments by Origin State",
       x = "Origin State",
       y = "Total Shipments") +
  theme_minimal() +
  theme(text = element_text(size = 12),  # Increase text size for readability
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),  # Rotate and align x-axis labe
        axis.title.x = element_text(margin = margin(t = 10)),  # Add margin to x-axis title
        axis.title.y = element_text(margin = margin(r = 10)),  # Add margin to y-axis title
        plot.margin = margin(1, 1, 1, 1, "cm"),  # Adjust plot margins
        panel.grid.major.x = element_blank(),  # Remove vertical grid lines for clarity
        panel.grid.minor.x = element_blank()) +  # Remove minor vertical grid lines
  scale_x_discrete(expand = expansion(add = 0.5))  # Add space around x-axis
```



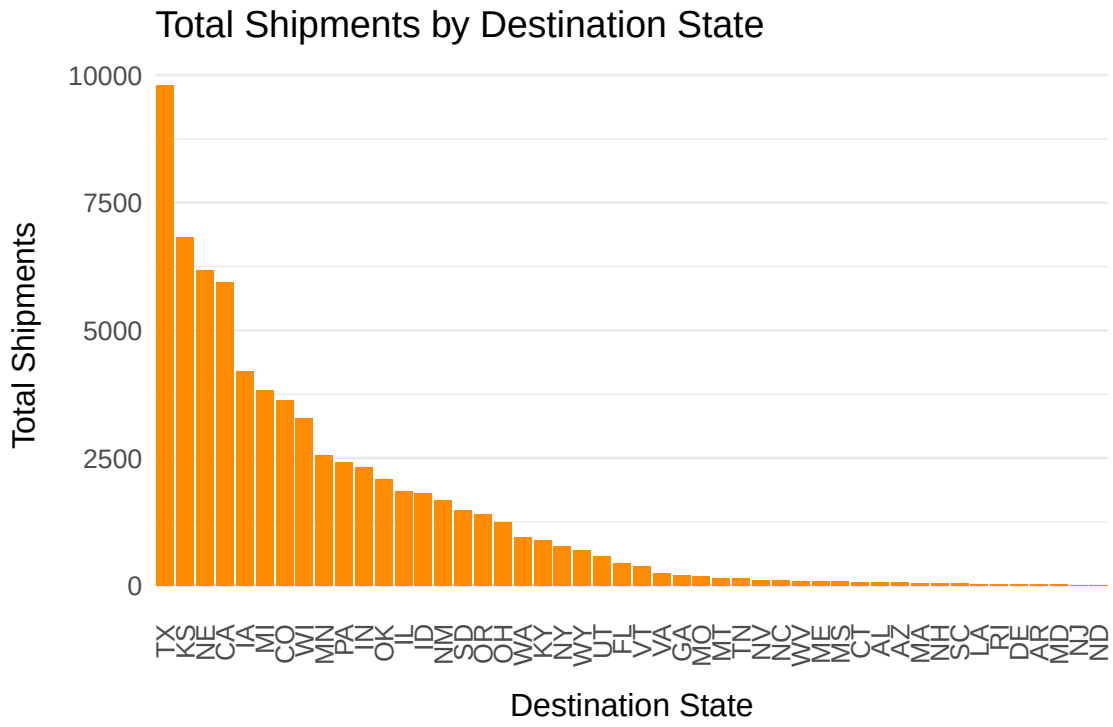
to make this better for reading, ill make the table version of this as well:

```
write.csv(origin_summary, "total_shipments_by_origin_state.csv", row.names = FALSE)
```

now i want to see what states are the recipent of shipment the most:

```
destination_summary <- movements_data %>%
  group_by(Destination) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  arrange(desc(Total_Shipments))

# Plotting the bar plot for destination states
ggplot(destination_summary, aes(x = reorder(Destination, -Total_Shipments), y = Total_Shipments)) +
```

```
  geom_bar(stat = "identity", fill = "darkorange") +
  labs(title = "Total Shipments by Destination State",
       x = "Destination State",
       y = "Total Shipments") +
  theme_minimal() +
  theme(text = element_text(size = 12),  # Increase text size for readability
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),  # Rotate and align x-axis labe
        axis.title.x = element_text(margin = margin(t = 10)),  # Add margin to x-axis title
        axis.title.y = element_text(margin = margin(r = 10)),  # Add margin to y-axis title
        plot.margin = margin(1, 1, 1, 1, "cm"),  # Adjust plot margins
        panel.grid.major.x = element_blank(),  # Remove vertical grid lines for clarity
        panel.grid.minor.x = element_blank()) +  # Remove minor vertical grid lines
  scale_x_discrete(expand = expansion(add = 0.5))  # Add space around x-axis
```



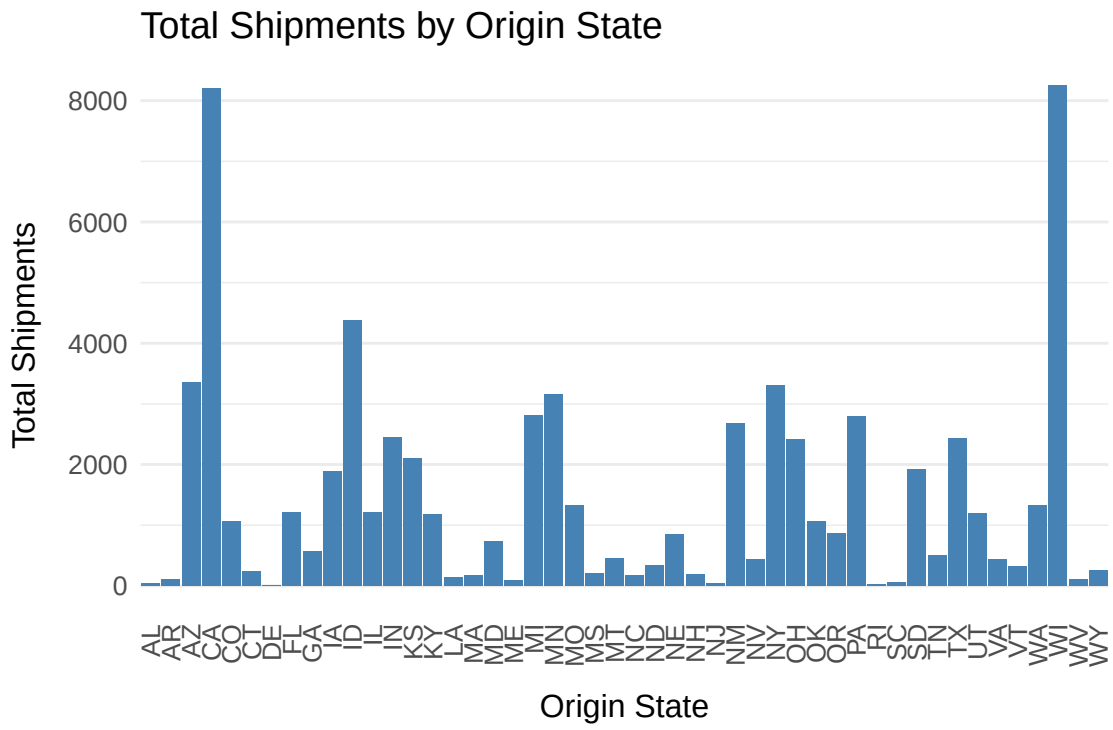
now the table:

```
write.csv(destination_summary, "total_shipments_by_destination_state.csv", row.names = FALSE)
```

now lets see if we can make it read better. so now we are going to reorder them alphabetically.

```
origin_summary <- movements_data %>%
  group_by(Origin) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  arrange(Origin)  # Arrange alphabetically

ggplot(origin_summary, aes(x = Origin, y = Total_Shipments)) +
  geom_bar(stat = "identity", fill = "steelblue") +
```

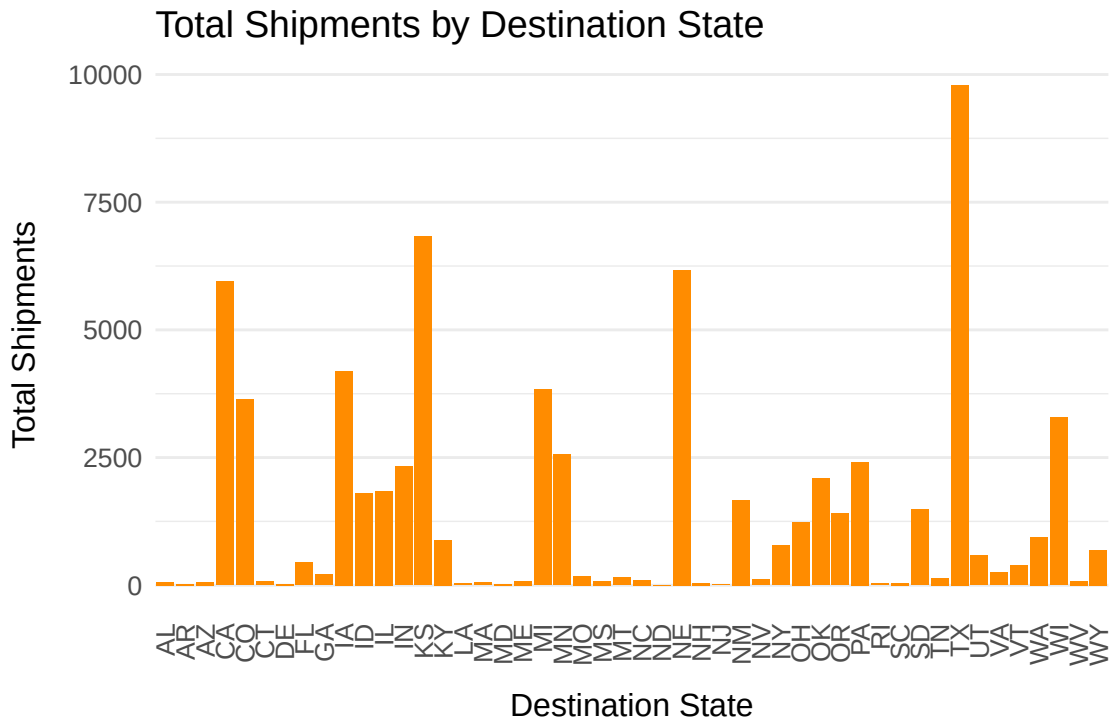
```
  labs(title = "Total Shipments by Origin State",
       x = "Origin State",
       y = "Total Shipments") +
  theme_minimal() +
  theme(text = element_text(size = 12),  # Increase text size for readability
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),  # Rotate and align x-axis labe
        axis.title.x = element_text(margin = margin(t = 10)),  # Add margin to x-axis title
        axis.title.y = element_text(margin = margin(r = 10)),  # Add margin to y-axis title
        plot.margin = margin(1, 1, 1, 1, "cm"),  # Adjust plot margins
        panel.grid.major.x = element_blank(),  # Remove vertical grid lines for clarity
        panel.grid.minor.x = element_blank()) +  # Remove minor vertical grid lines
  scale_x_discrete(expand = expansion(add = 0.5))  # Add space around x-axis
```



now the same for destination states:

```
destination_summary <- movements_data %>%
  group_by(Destination) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  arrange(Destination)  # Arrange alphabetically

# Plotting the bar plot for destination states
ggplot(destination_summary, aes(x = Destination, y = Total_Shipments)) +
  geom_bar(stat = "identity", fill = "darkorange") +
  labs(title = "Total Shipments by Destination State",
       x = "Destination State",
       y = "Total Shipments") +
```

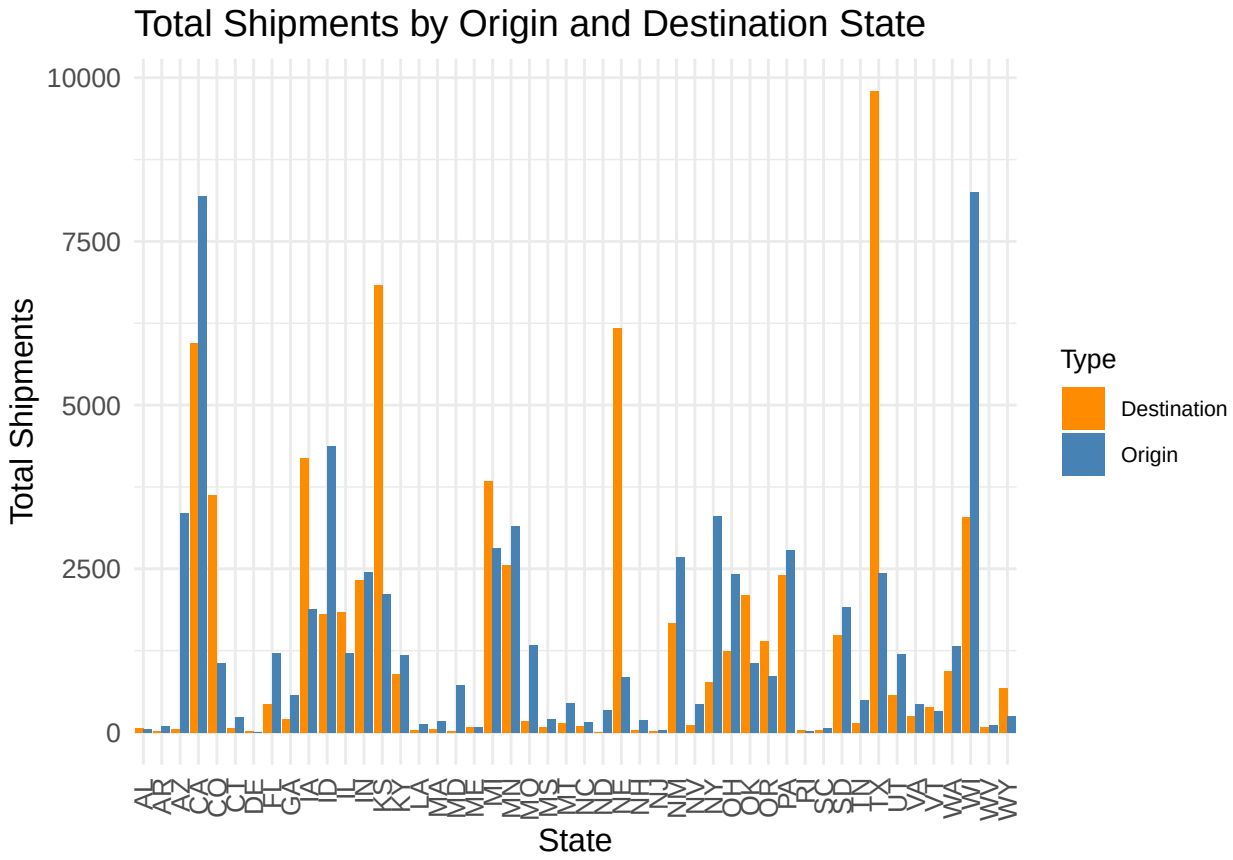
```
  theme_minimal() +
  theme(text = element_text(size = 12),  # Increase text size for readability
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),  # Rotate and align x-axis labe
        axis.title.x = element_text(margin = margin(t = 10)),  # Add margin to x-axis title
        axis.title.y = element_text(margin = margin(r = 10)),  # Add margin to y-axis title
        plot.margin = margin(1, 1, 1, 1, "cm"),  # Adjust plot margins
        panel.grid.major.x = element_blank(),  # Remove vertical grid lines for clarity
        panel.grid.minor.x = element_blank()) +  # Remove minor vertical grid lines
  scale_x_discrete(expand = expansion(add = 0.5))  # Add space around x-axis
```



this is pretty good but i can make this better i think. i can combine them and get two columns and see everything better.

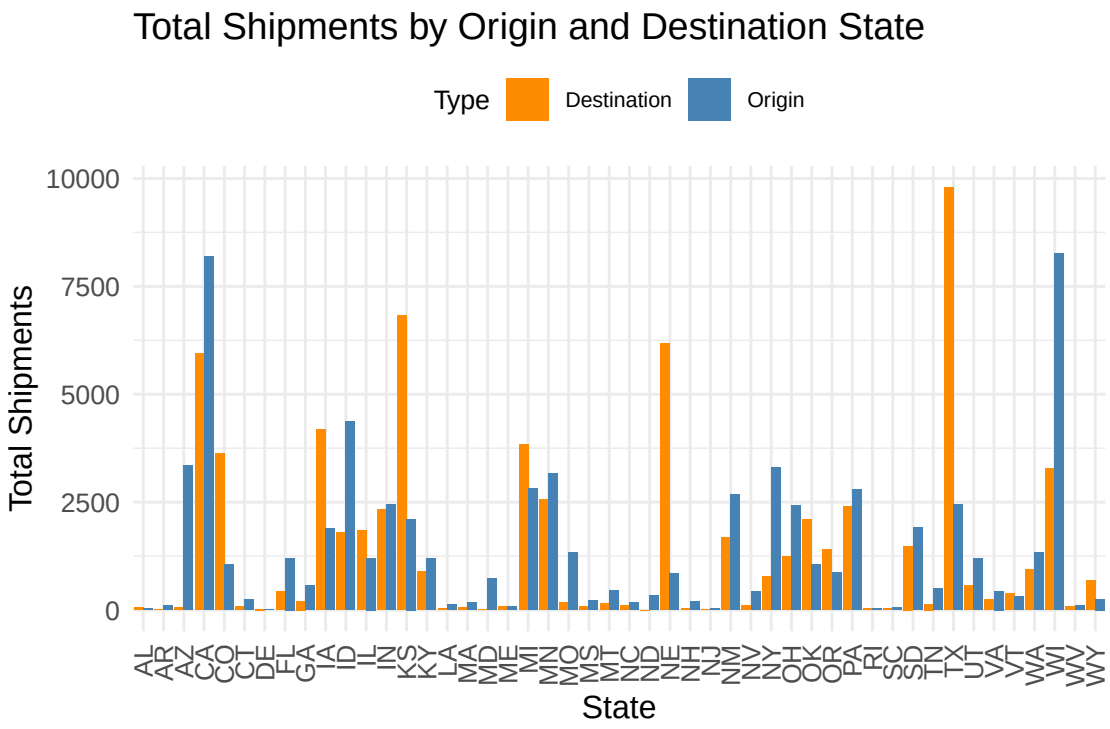
```
library(ggplot2)
library(dplyr)

# Combine the summaries for origin and destination
combined_summary <- movements_data %>%
  group_by(State = Origin) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  mutate(Type = "Origin") %>%
  bind_rows(
    movements_data %>%
      group_by(State = Destination) %>%
      summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
      mutate(Type = "Destination")
  ) %>%
```

```
  arrange(State, Type)

# Plotting the combined bar plot
ggplot(combined_summary, aes(x = State, y = Total_Shipments, fill = Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("Origin" = "steelblue", "Destination" = "darkorange")) +
  labs(title = "Total Shipments by Origin and Destination State",
       x = "State",
       y = "Total Shipments") +
  theme_minimal() +
  theme(text = element_text(size = 12),
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
        plot.margin = margin(.1, .1, .1, .1, "cm"),  # Increased right margin for legend space
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.position = "right") +  # Keep the legend on the right
  scale_x_discrete(expand = expansion(add = 0.5))  # Add space around x-axis
```




```
# Combine the summaries for origin and destination
combined_summary <- movements_data %>%
  group_by(State = Origin) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
  mutate(Type = "Origin") %>%
  bind_rows(
    movements_data %>%
      group_by(State = Destination) %>%
```

```
      summarise(Total_Shipments = sum(`Counts of Shipment`)) %>%
      mutate(Type = "Destination")
  ) %>%
  arrange(State, Type)

# Plotting the combined bar plot
ggplot(combined_summary, aes(x = State, y = Total_Shipments, fill = Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("Origin" = "steelblue", "Destination" = "darkorange")) +
  labs(title = "Total Shipments by Origin and Destination State",
       x = "State",
       y = "Total Shipments") +
  theme_minimal() +
  theme(text = element_text(size = 12),
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),  # Adjust vertical alignment
        plot.margin = margin(1, 1, 1, 1, "cm"),
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 8),
        legend.position = "top") +
  scale_x_discrete(expand = expansion(add = 0.5))  # Add extra space around the x-axis
```



remember that i said i dont want to think about the destination when im looking at origin at first? now i want to consider both the destination and the origin.

for this i came up with three main data presentaion tools: Chord Diagram: Visualizes connections between states with arcs, showing the volume of shipments. Sankey Diagram: Illustrates the flow from origin to

destination, with widths representing shipment volumes. Heatmap: Shows the intensity of shipments between each origin and destination pair.

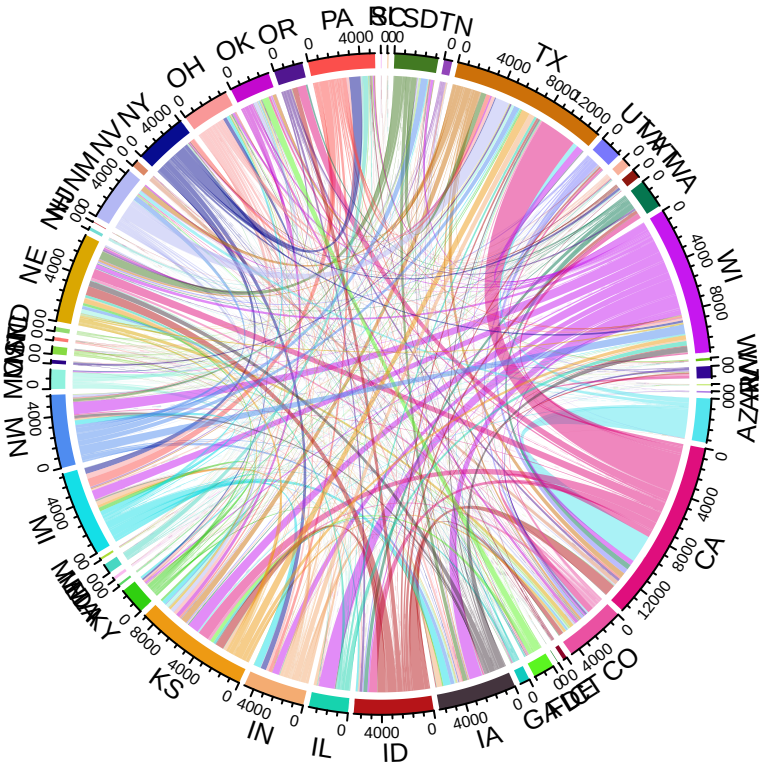
1. Chord Diagram

```
library(circlize)
```

```
## Warning: package 'circlize' was built under R version 4.3.3
```

```
## ========================================
## circlize version 0.4.16
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
origin_destination_summary <- movements_data %>%
  group_by(Origin, Destination) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')

shipment_matrix <- reshape2::acast(origin_destination_summary, Origin ~ Destination, value.var = "Total_

chordDiagram(shipment_matrix, transparency = 0.5)
```



THIS WORKS, I JUST MADE IT INTO A COMMENT SO THE FILE KNIT FASTER

```
# png("chord_diagram.png", width = 3000, height = 3000, res = 200)
# chordDiagram(shipment_matrix, transparency = 0.5)
# dev.off()
```

2. Sankey Diagram

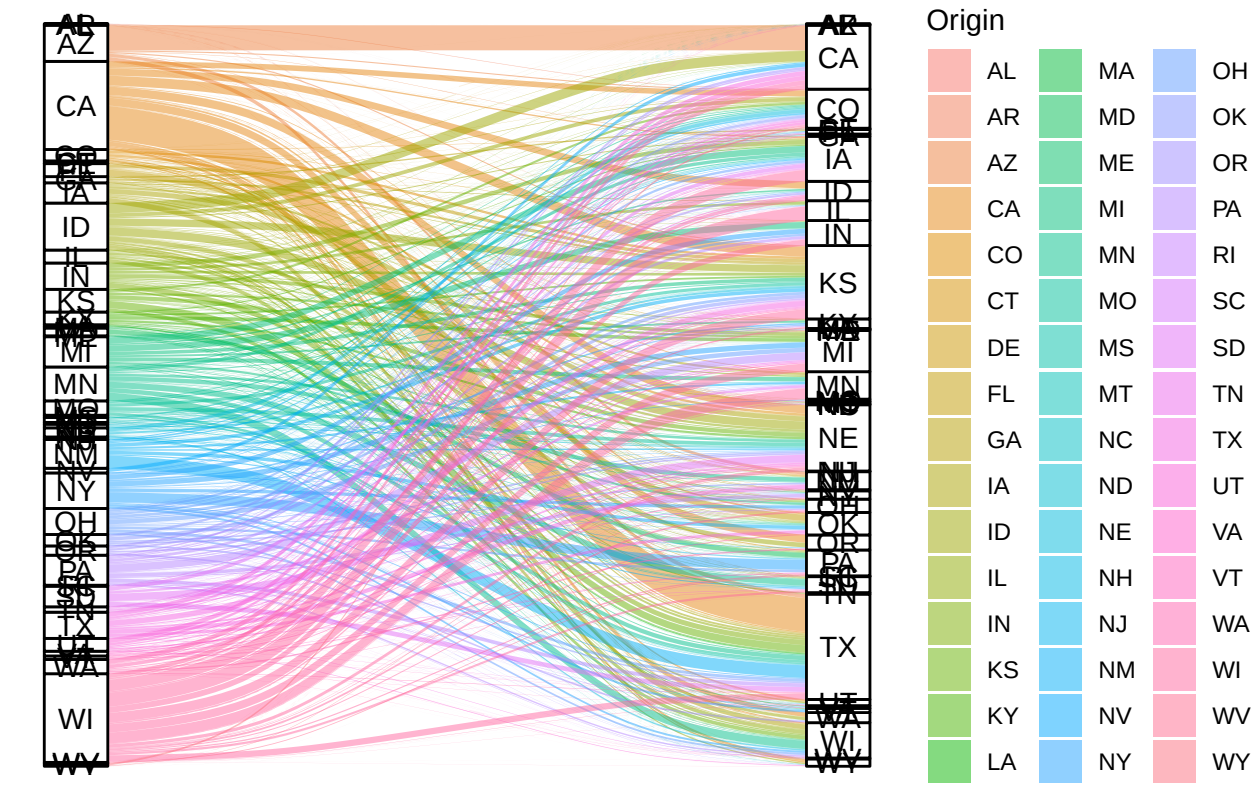
```
library(ggalluvial)
```

```
## Warning: package 'ggalluvial' was built under R version 4.3.3
```

```
sankey_data <- movements_data %>%
  group_by(Origin, Destination) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')



sankey_plot <- ggplot(sankey_data,
                      aes(axis1 = Origin, axis2 = Destination, y = Total_Shipments)) +
  geom_alluvium(aes(fill = Origin), width = 1/12) +
  geom_stratum(width = 1/12) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  theme_void() +
  ggtitle("Flow of Shipments from Origin to Destination States")

print(sankey_plot)
```

```
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
```

```
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
```



I MADE THIS ONE AS A COMMENT BC IT TAKES SO MUCH TIME - BUT THE CODE IS CORRECT AND IT WORKS

lets save the image now

```
# png("sankey_diagram_temp.png", width = 10000, height = 8000, res = 500)
# print(sankey_plot)
# dev.off()
```

3: Heatmap

```
heatmap_data <- movements_data %>%
  group_by(Origin, Destination) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')


heatmap_plot <- ggplot(heatmap_data, aes(x = Origin, y = Destination, fill = Total_Shipments)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Heatmap of Shipments from Origin to Destination States",
       x = "Origin State",
```

```
       y = "Destination State",
       fill = "Total Shipments") +
  theme_minimal()

print(heatmap_plot)
```

save the heat map:

```
# png("heatmap_diagram_temp.png", width = 6500, height = 4000, res = 500)
# print(heatmap_plot)
# dev.off()
```

now i see i have so many different colors - what im thinking right now is to group my data based on bird migratory(flyway) pathway.

then id only have 4 colors and id remake the diagrams.

According to U.S. Fish & Wildlife Service: https://fws.gov/partner/migratory-bird-program-administrative-flyways

Atlantic Flyway The Atlantic Flyway Council is composed of the states of Connecticut, Delaware, Florida, Georgia, Maine, Maryland, Massachusetts, New Hampshire, New Jersey, New York, North Carolina, Pennsylvania, Rhode Island, South Carolina, Vermont, Virginia, and West Virginia

Mississippi Flyway Administratively, the Mississippi Flyway is composed of the states of Alabama, Arkansas, Indiana, Illinois, Iowa, Kentucky, Louisiana, Michigan, Minnesota, Mississippi, Missouri, Ohio, Tennessee, and Wisconsin

Central Flyway The Central Flyway was formed in 1948 and is composed of the states of Montana, Wyoming, Colorado, New Mexico, Texas, Oklahoma, Kansas, Nebraska, South Dakota, and North Dakota,

Pacific Flyway U.S. members of the Pacific Flyway Council include Alaska, Arizona, California, Idaho, Nevada, Oregon, Utah, Washington, and those portions of Colorado, Montana, New Mexico, and Wyoming west of the Continental Divide.

Here:

```
# List of states in each flyway
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AK", "AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Filter the "Other" states
other_states <- movements_data %>%
  filter(!Origin %in% c(atlantic_states, mississippi_states, central_states, pacific_states)) %>%
  distinct(Origin) %>%
  pull(Origin)

# Display "Other" states
print(other_states)
```

```
## character(0)
```

```
movements_data <- movements_data %>%
  mutate(Flyway = case_when(
    Origin %in% c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC", "
    Origin %in% c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "WI") ~
    Origin %in% c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND") ~ "Central Flyway",
    Origin %in% c("AK", "AZ", "CA", "ID", "NV", "OR", "UT", "WA") ~ "Pacific Flyway",
    TRUE ~ "Other"  # For any states not listed
  ))

movements_data <- movements_data %>%
  mutate(Destination_Flyway = case_when(
    Destination %in% c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC
    Destination %in% c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "WI
    Destination %in% c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND") ~ "Central Flyway",
    Destination %in% c("AK", "AZ", "CA", "ID", "NV", "OR", "UT", "WA") ~ "Pacific Flyway",
    TRUE ~ "Other"  # For any states not listed
  ))
```

I am using both state assigning things to check for any states that are missing from the flyways and then ensure all states are categorized correctly in my dataset.

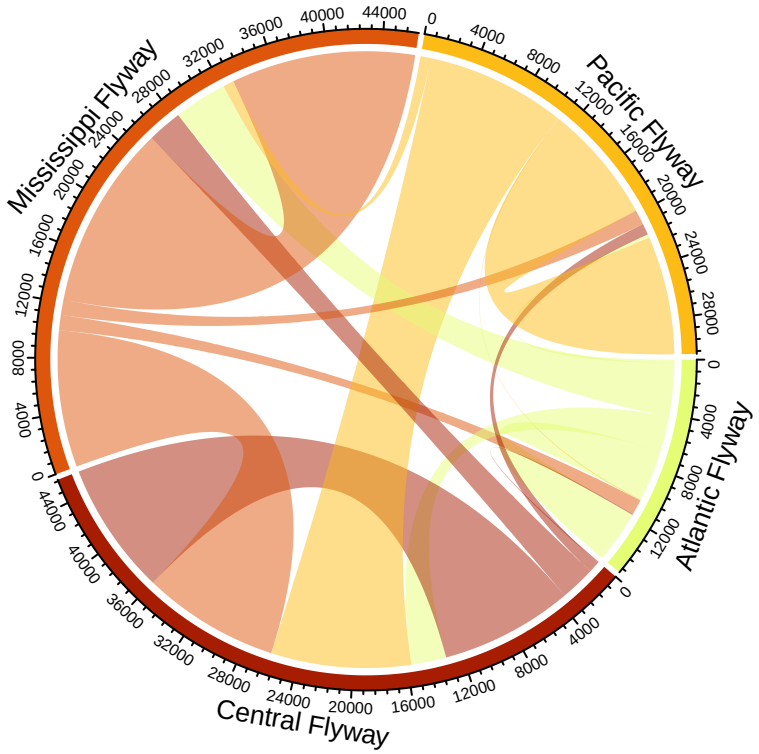
Chord Diagram:

```
library(circlize)
library(dplyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
origin_destination_summary <- movements_data %>%
  group_by(Flyway, Destination_Flyway) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')

# Create a matrix for the chord diagram
shipment_matrix_flyway <- reshape2::acast(origin_destination_summary, Flyway ~ Destination_Flyway, valu

# Create the chord diagram
chordDiagram(shipment_matrix_flyway, transparency = 0.5)
```



lets save this:

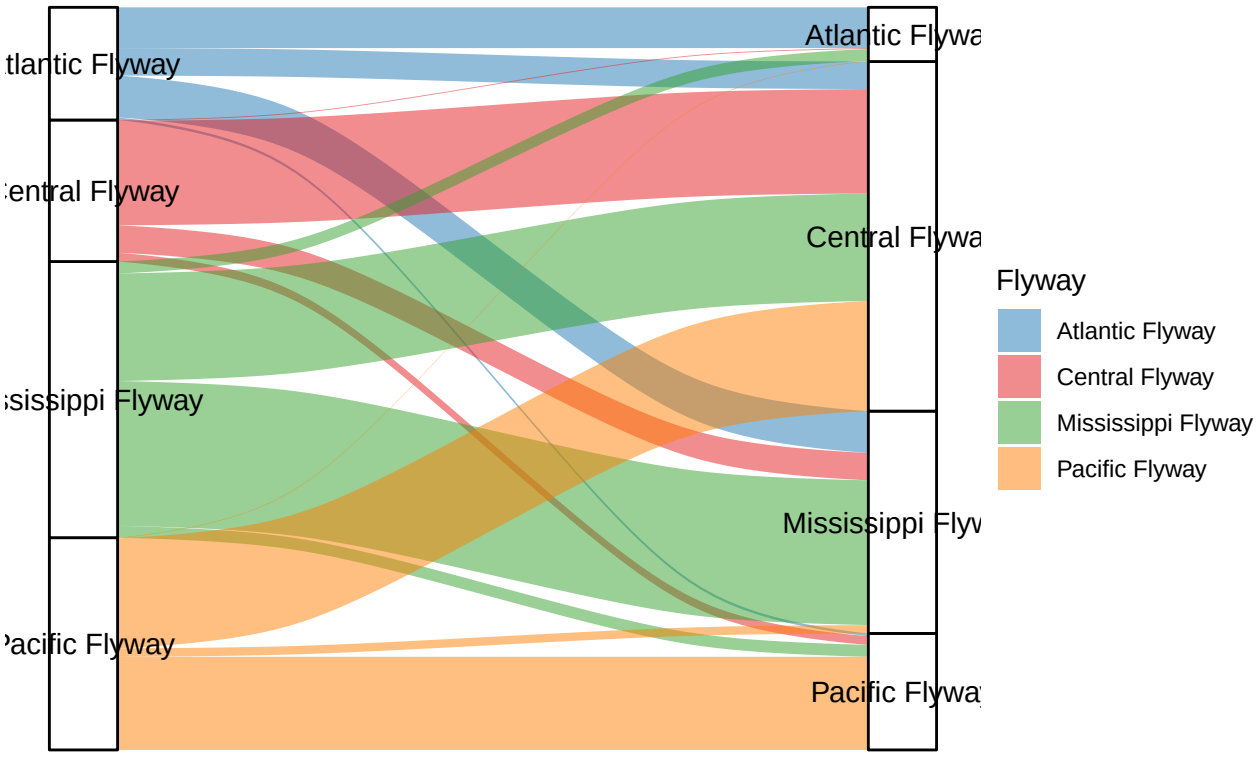
```r
# png("chord_diagram_flyways.png", width = 1500, height = 1500, res = 200)
# chordDiagram(shipment_matrix_flyway, transparency = 0.5)
# dev.off()
```

sankey:

```r
library(ggplot2)
library(ggalluvial)

sankey_data <- movements_data %>%
  group_by(Flyway, Destination_Flyway) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')


# Define nature-inspired colors
nature_colors <- c("Atlantic Flyway" = "#1f78b4",   # Blue
```

```
                    "Mississippi Flyway" = "#33a02c", # Green
                    "Central Flyway" = "#e31a1c",     # Red
                    "Pacific Flyway" = "#ff7f00")     # Orange

# Plot the Sankey diagram with nature-inspired colors
sankey_plot <- ggplot(sankey_data,
                      aes(axis1 = Flyway, axis2 = Destination_Flyway, y = Total_Shipments)) +
  geom_alluvium(aes(fill = Flyway), width = 1/12) +
  geom_stratum(width = 1/12) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum))) +
  theme_void() +
  ggtitle("Flow of Shipments Between Flyways") +
  scale_fill_manual(values = nature_colors)

# Display the Sankey diagram
print(sankey_plot)
```

```
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
## Warning in to_lodes_form(data = data, axes = axis_ind, discern =
## params$discern): Some strata appear at multiple axes.
```

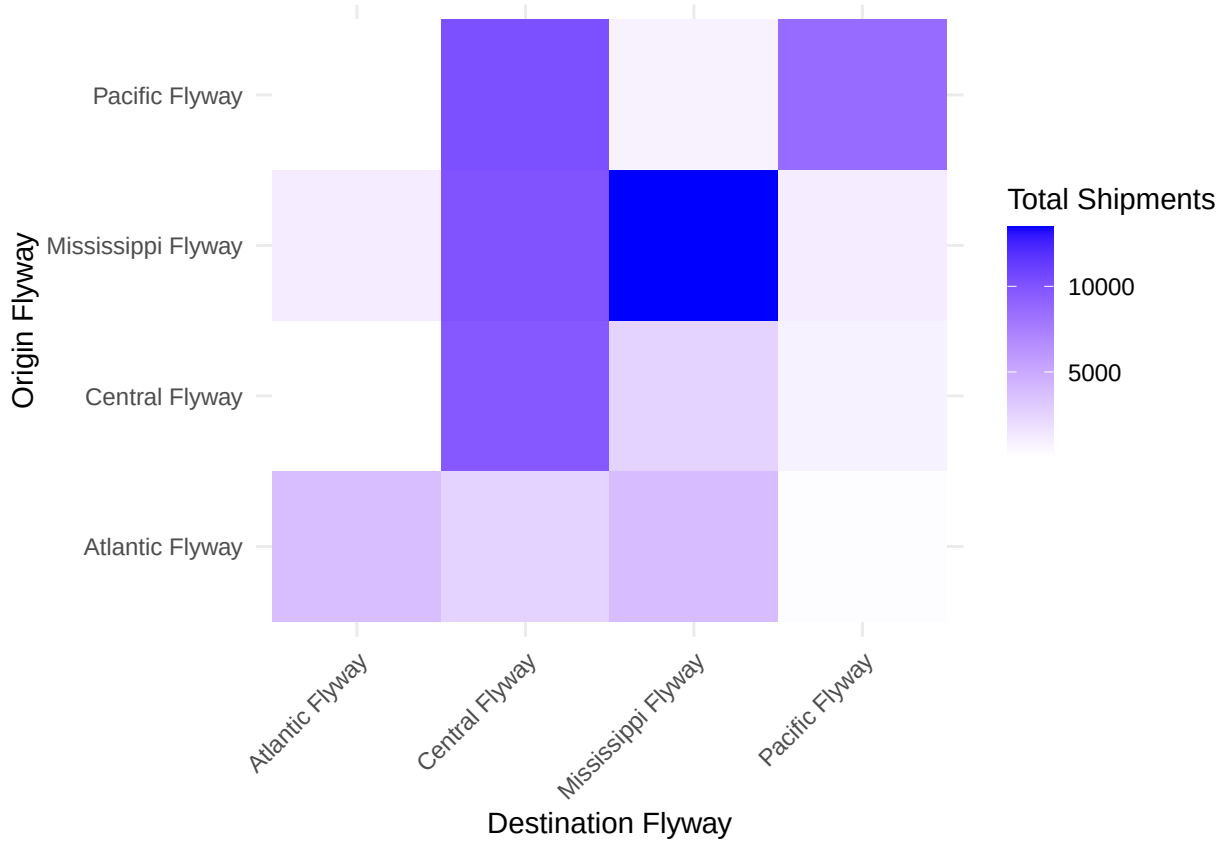


let's save it:

```
# png("sankey_diagram_flyways.png", width = 6500, height = 4000, res = 500)
# print(sankey_plot)
# dev.off()
```

heatmap:

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(reshape2)  # For melting the matrix

# Summarize the data
origin_destination_summary <- movements_data %>%
  group_by(Flyway, Destination_Flyway) %>%
  summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')

# Pivot the data for the heatmap
heatmap_data <- origin_destination_summary %>%
  pivot_wider(names_from = Destination_Flyway, values_from = Total_Shipments, values_fill = 0)

# Convert to matrix for heatmap
heatmap_matrix <- as.matrix(heatmap_data[,-1])
rownames(heatmap_matrix) <- heatmap_data$Flyway

# Create the heatmap plot
heatmap_plot <- ggplot(melt(heatmap_matrix), aes(x=Var2, y=Var1, fill=value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(x = "Destination Flyway", y = "Origin Flyway", fill = "Total Shipments") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display the heatmap plot
print(heatmap_plot)
```



lets save it:

```
# png("heatmap_flyways.png", width = 2500, height = 2000, res = 500)
# print(heatmap_plot)
# dev.off()
```

now we want to make maps: lets give it some long/lang:

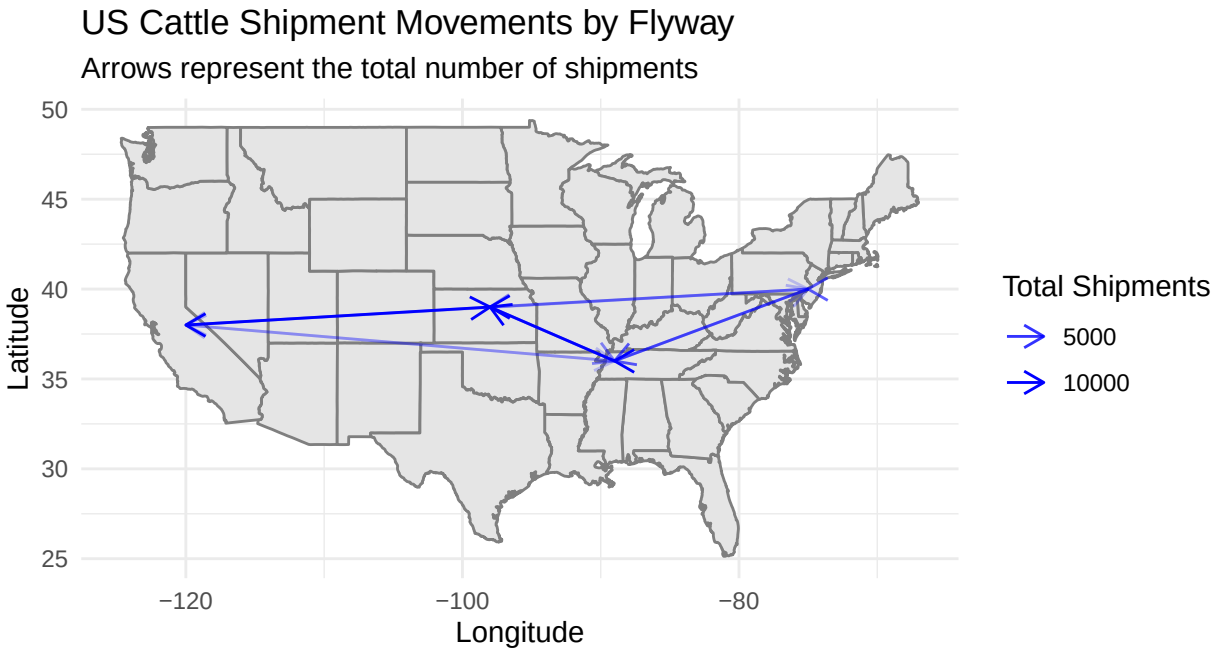
```
# State coordinates provided by the user
state_coords <- data.frame(
  state = c("ALABAMA", "ALASKA", "ARIZONA", "ARKANSAS", "CALIFORNIA", "COLORADO", "CONNECTICUT", "DELAWA
            "HAWAII", "IDAHO", "ILLINOIS", "INDIANA", "IOWA", "KANSAS", "KENTUCKY", "LOUISIANA", "MAINE
            "MICHIGAN", "MINNESOTA", "MISSISSIPPI", "MISSOURI", "MONTANA", "NEBRASKA", "NEVADA", "NEW HA
            "NEW MEXICO", "NEW YORK", "NORTH CAROLINA", "NORTH DAKOTA", "OHIO", "OKLAHOMA", "OREGON", "P
            "SOUTH CAROLINA", "SOUTH DAKOTA", "TENNESSEE", "TEXAS", "UTAH", "VERMONT", "VIRGINIA", "WASH
            "WISCONSIN", "WYOMING"),
  abbr = c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS
           "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "OH", "OK
           "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"),
  lat = c(32.806671, 61.370716, 33.729759, 34.969704, 36.116203, 39.059811, 41.597782, 39.318523, 27.766
          44.240459, 40.349457, 39.849426, 42.011539, 38.5266, 37.66814, 31.169546, 44.693947, 39.063946
          45.694454, 32.741646, 38.456085, 46.921925, 41.12537, 38.313515, 43.452492, 40.298904, 34.840
          47.528912, 40.388783, 35.565342, 44.572021, 40.590752, 41.680893, 33.856892, 44.299782, 35.747
          44.045876, 37.769337, 47.400902, 38.491226, 44.268543, 42.755966),
  lon = c(-86.79113, -152.404419, -111.431221, -92.373123, -119.681564, -105.311104, -72.755371, -75.507
          -157.498337, -114.478828, -88.986137, -86.258278, -93.210526, -96.726486, -84.670067, -91.8678
          -71.530106, -84.536095, -93.900192, -89.678696, -92.288368, -110.454353, -98.268082, -117.055
          -106.248482, -74.948051, -79.806419, -99.784012, -82.764915, -96.928917, -122.070938, -77.209
```

```
          -99.438828, -86.692345, -97.563461, -111.862434, -72.710686, -78.169968, -121.490494, -80.954
)
```

now lets see if we can make the map based on flyway patterns

```
flyway_coords <- data.frame(
  Flyway = c("Atlantic Flyway", "Central Flyway", "Mississippi Flyway", "Pacific Flyway"),
  lon = c(-75, -98, -89, -120),
  lat = c(40, 39, 36, 38)
)

movement_data <- origin_destination_summary %>%
  inner_join(flyway_coords, by = c("Flyway" = "Flyway")) %>%
  rename(lon_origin = lon, lat_origin = lat) %>%
  inner_join(flyway_coords, by = c("Destination_Flyway" = "Flyway")) %>%
  rename(lon_dest = lon, lat_dest = lat)

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(data = movement_data, aes(x = lon_origin, y = lat_origin, xend = lon_dest, yend = lat_des
                                         alpha = Total_Shipments),
               arrow = arrow(length = unit(0.3, "cm")), color = "blue") +
  scale_alpha_continuous(range = c(0.1, 2)) +
  coord_fixed(1.3) +
  theme_minimal() +
  labs(title = "US Cattle Shipment Movements by Flyway",
       subtitle = "Arrows represent the total number of shipments",
       x = "Longitude", y = "Latitude", alpha = "Total Shipments")
```

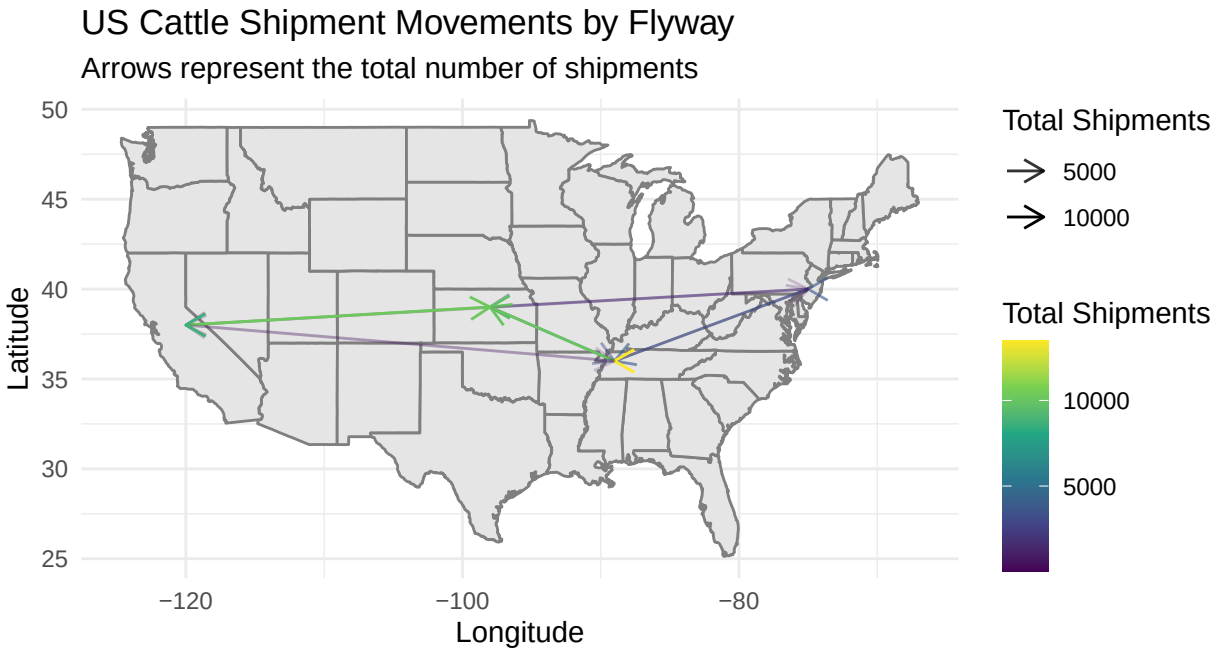
```
## Warning: Duplicated aesthetics after name standardisation: colour
```



another way: this one we are changing the pallet

```
#I made this into a comment, because this already exits in my code from line from line 108 to 262

# origin_destination_summary <- movements_data %>%
#   group_by(Flyway, Destination_Flyway) %>%
#   summarise(Total_Shipments = sum(`Counts of Shipment`), .groups = 'drop')


ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(data = movement_data, aes(x = lon_origin, y = lat_origin, xend = lon_dest, yend = lat_des
                                         alpha = Total_Shipments, color = Total_Shipments),
               arrow = arrow(length = unit(0.3, "cm"))) +
  scale_alpha_continuous(range = c(0.1, 2)) +
  scale_color_viridis_c() +  # Color scale for better distinction
  coord_fixed(1.3) +
  theme_minimal() +
  labs(title = "US Cattle Shipment Movements by Flyway",
       subtitle = "Arrows represent the total number of shipments",
       x = "Longitude", y = "Latitude", alpha = "Total Shipments", color = "Total Shipments")
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
```



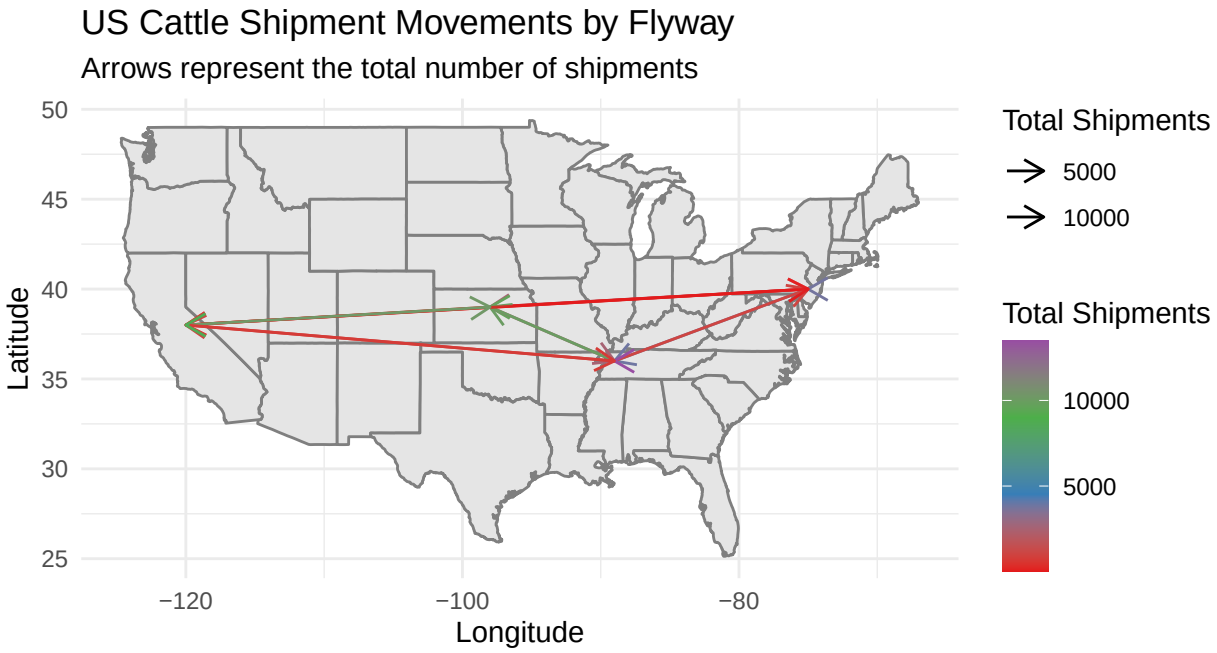

```
# Count the number of unique origin-destination pairs in your data
number_of_arrows <- nrow(movement_data)
print(paste("Number of arrows:", number_of_arrows))
```

```
## [1] "Number of arrows: 16"
```

a specific set

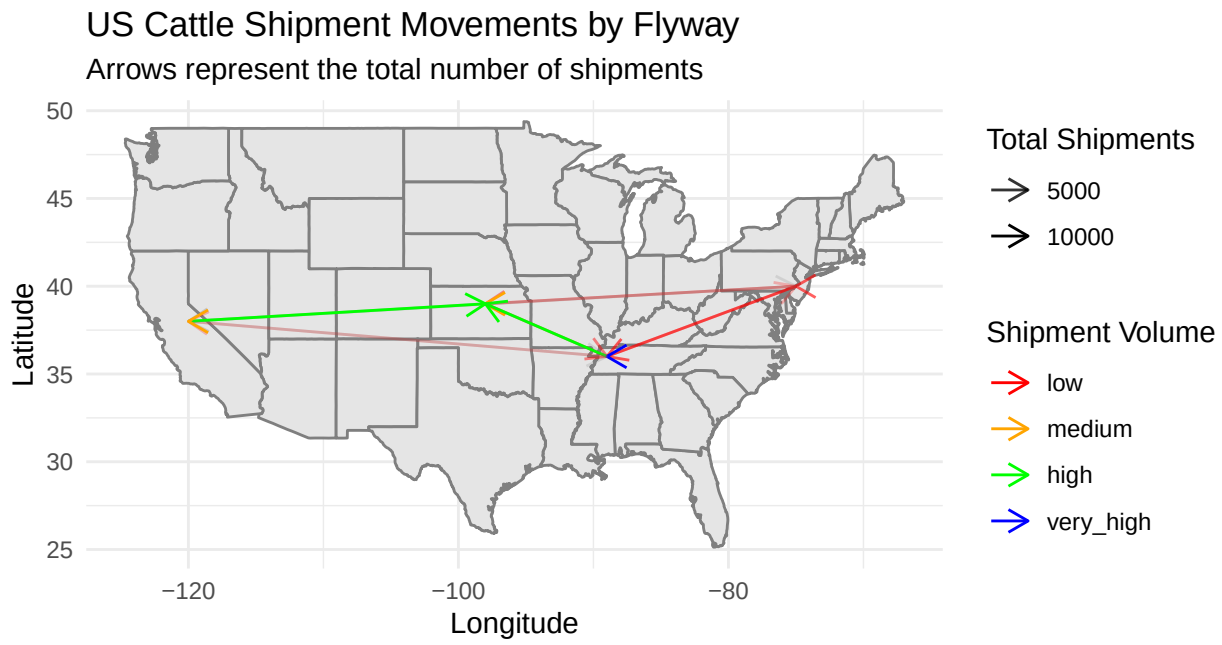
```
# Use a high-contrast palette from RColorBrewer
ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(data = movement_data, aes(x = lon_origin, y = lat_origin, xend = lon_dest, yend = lat_des
                                         alpha = Total_Shipments, color = Total_Shipments),
               arrow = arrow(length = unit(0.3, "cm"))) +
  scale_alpha_continuous(range = c(0.8, 5)) +
  scale_color_gradientn(colors = brewer.pal(4, "Set1")) +  # Apply a palette with more distinct colors
  coord_fixed(1.3) +
  theme_minimal() +
  labs(title = "US Cattle Shipment Movements by Flyway",
       subtitle = "Arrows represent the total number of shipments",
       x = "Longitude", y = "Latitude", alpha = "Total Shipments", color = "Total Shipments")
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
```




```
library(ggplot2)
library(sf)

# Define custom color scale with more distinguished colors
custom_colors <- c("low" = "red", "medium" = "orange", "high" = "green", "very_high" = "blue")

movement_data$Shipment_Category <- cut(movement_data$Total_Shipments,
                                       breaks = c(-Inf, 1000, 5000, 10000, 12000, Inf),
                                       labels = c("very_low", "low", "medium", "high", "very_high"))

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(data = movement_data, aes(x = lon_origin, y = lat_origin, xend = lon_dest, yend = lat_des
                                         alpha = Total_Shipments, color = Shipment_Category),
               arrow = arrow(length = unit(0.3, "cm"))) +
  scale_alpha_continuous(range = c(0.1, 2)) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  coord_fixed(1.3) +
  theme_minimal() +
  labs(title = "US Cattle Shipment Movements by Flyway",
       subtitle = "Arrows represent the total number of shipments",
       x = "Longitude", y = "Latitude", alpha = "Total Shipments", color = "Shipment Volume")
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
```

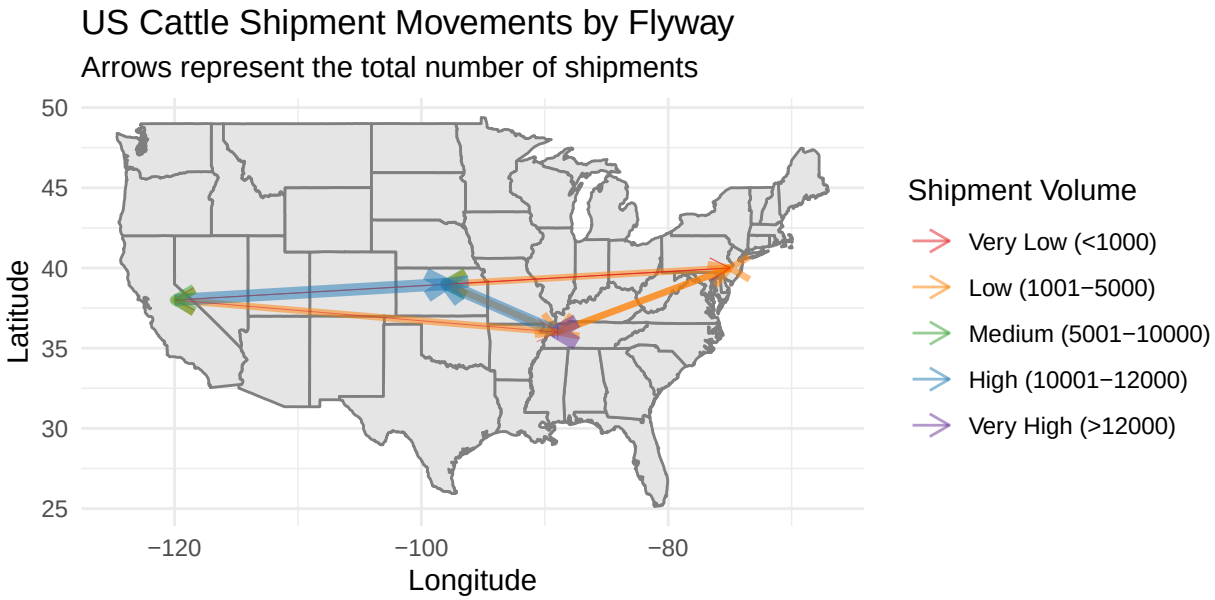



```r
# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(movement_data$Total_Shipments,
                                       breaks = c(0, 1000, 5000, 10000, 12000, Inf),
                                       labels = c("Very Low (<1000)", "Low (1001-5000)", "Medium (5001-
                                       right = TRUE)

# Define size mapping for each Shipment Category
size_mapping <- c("Very Low (<1000)" = 0.1,
                  "Low (1001-5000)" = 0.5,
                  "Medium (5001-10000)" = 1,
                  "High (10001-12000)" = 1.5,
                  "Very High (>12000)" = 2)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

ggplot() +
```

```
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(data = movement_data, aes(x = lon_origin, y = lat_origin, xend = lon_dest, yend = lat_des
                                         color = Shipment_Category, size = Arrow_Size),
               arrow = arrow(length = unit(0.3, "cm"))) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scalex,
  scale_size_continuous(range = c(0.1, 2.5), guide = "none") +  # Ensure sizes are set as per the mappi
  coord_fixed(1.3) +
  theme_minimal() +
  labs(title = "US Cattle Shipment Movements by Flyway",
       subtitle = "Arrows represent the total number of shipments",
       x = "Longitude", y = "Latitude", color = "Shipment Volume", size = "Total Shipments") +
  theme(legend.position = "right")  # Position the legend to the right for better visibility
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



to fix the error:

```
library(ggplot2)
library(sf)
library(scales) # To use the alpha function
```

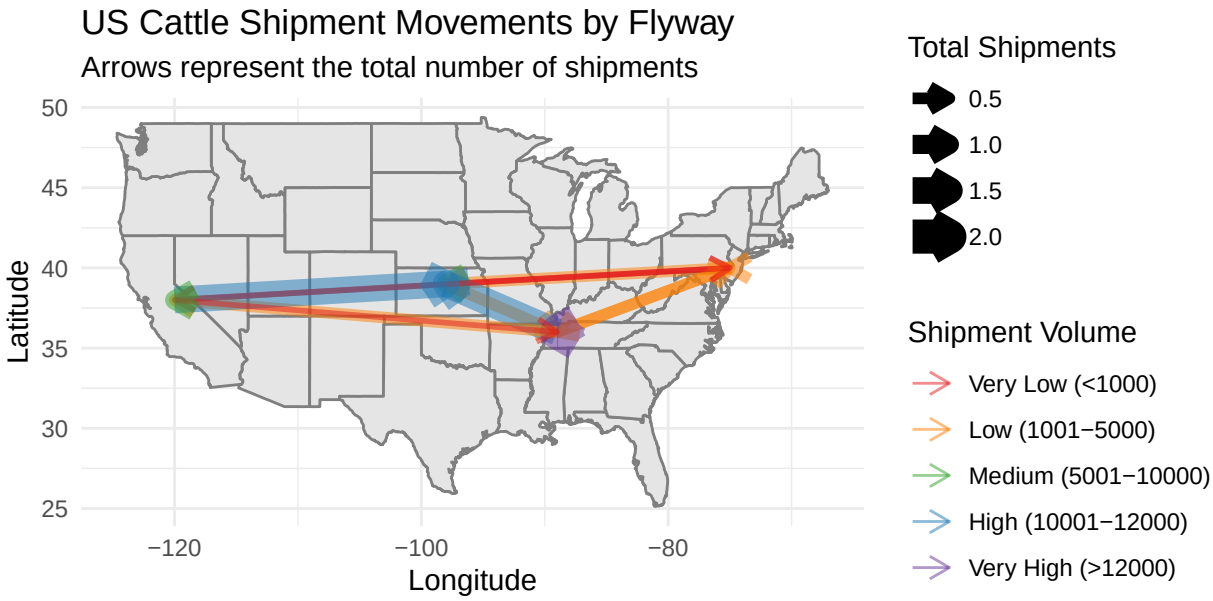
```
##
```

```
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.1,
  "Low (1001-5000)" = 0.5,
  "Medium (5001-10000)" = 1,
  "High (10001-12000)" = 1.5,
  "Very High (>12000)" = 2
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth instead of size
    ),
    arrow = arrow(length = unit(0.3, "cm"))
  ) +
```

```
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  scale_size_continuous(range = c(0.1, 2.5), guide = "none") +  # Removed duplicate size scale
  coord_fixed(1.3) +
  theme_minimal() +
  labs(
    title = "US Cattle Shipment Movements by Flyway",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume", linewidth = "Total Shipments"
  ) +
  theme(legend.position = "right")
```

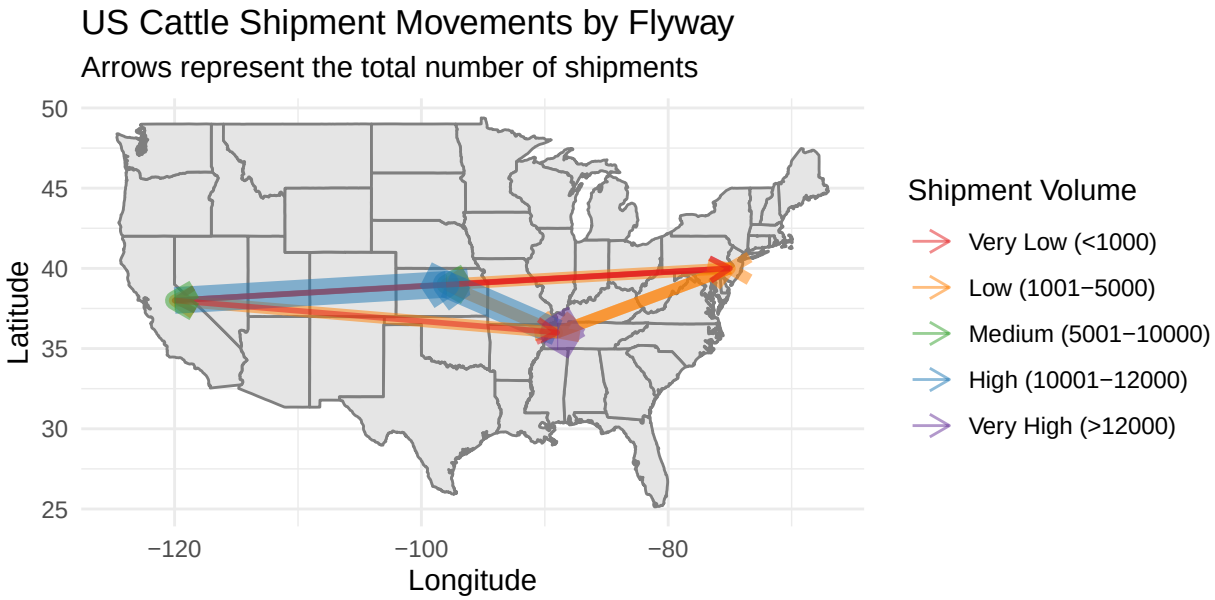
```
## Warning: Duplicated aesthetics after name standardisation: colour
```




```
library(ggplot2)
library(sf)
library(scales) # To use the alpha function

# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)
```

```r
# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.1,
  "Low (1001-5000)" = 0.5,
  "Medium (5001-10000)" = 1,
  "High (10001-12000)" = 1.5,
  "Very High (>12000)" = 2
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth instead of size
    ),
    arrow = arrow(length = unit(0.3, "cm")),
    show.legend = c(color = TRUE, linewidth = FALSE)  # Hide linewidth legend
  ) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  coord_fixed(1.3) +
  theme_minimal() +
  labs(
    title = "US Cattle Shipment Movements by Flyway",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume"
  ) +
  theme(legend.position = "right")
```

```
## Warning: Duplicated aesthetics after name standardisation: colour
```




```r
# Load necessary libraries
library(ggplot2)
library(maps)
library(dplyr)

# Define flyways as per your existing categorization
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AK", "AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Print the unique values in `region` and check if they are as expected
print(unique(us_states$region))
```

```
##  [1] "alabama"              "arizona"              "arkansas"
##  [4] "california"           "colorado"             "connecticut"
##  [7] "delaware"             "district of columbia" "florida"
## [10] "georgia"              "idaho"                "illinois"
## [13] "indiana"              "iowa"                 "kansas"
## [16] "kentucky"             "louisiana"            "maine"
## [19] "maryland"             "massachusetts"        "michigan"
## [22] "minnesota"            "mississippi"          "missouri"
## [25] "montana"              "nebraska"             "nevada"
```

```
## [28] "new hampshire"        "new jersey"           "new mexico"
## [31] "new york"             "north carolina"       "north dakota"
## [34] "ohio"                 "oklahoma"             "oregon"
## [37] "pennsylvania"         "rhode island"         "south carolina"
## [40] "south dakota"         "tennessee"            "texas"
## [43] "utah"                 "vermont"              "virginia"
## [46] "washington"           "west virginia"        "wisconsin"
## [49] "wyoming"
```

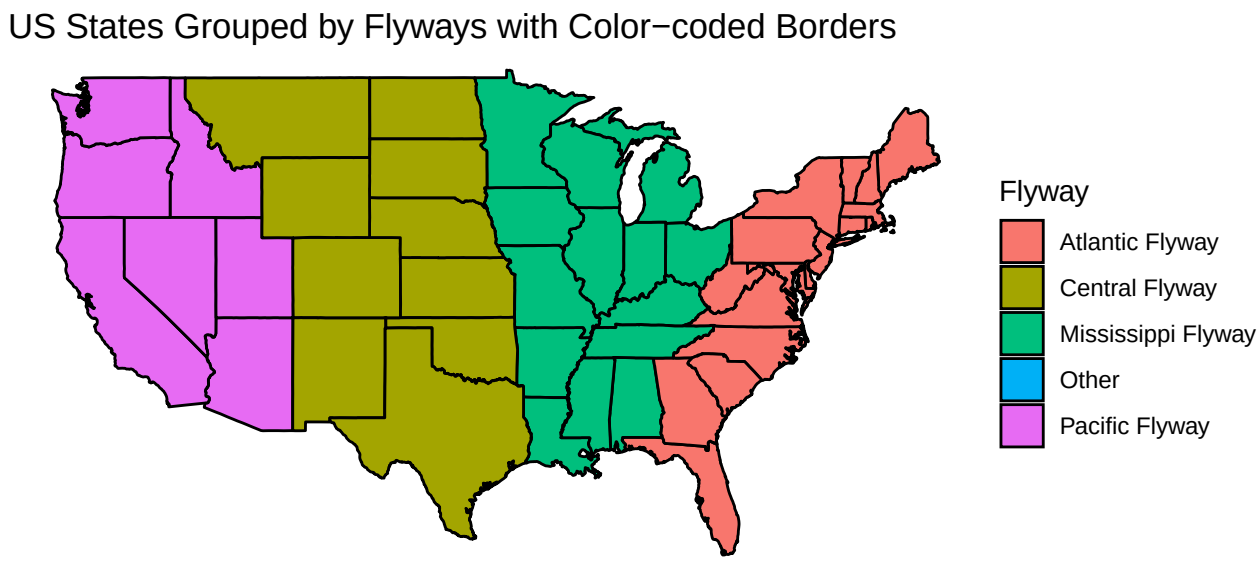
```r
# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% tolower(state.name[match(atlantic_states, state.abb)]) ~ "Atlantic Flyway",
  us_states$region %in% tolower(state.name[match(mississippi_states, state.abb)]) ~ "Mississippi Flyway
  us_states$region %in% tolower(state.name[match(central_states, state.abb)]) ~ "Central Flyway",
  us_states$region %in% tolower(state.name[match(pacific_states, state.abb)]) ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Verify the updated data
print(head(us_states))
```

```
##        long      lat group order  region subregion             Flyway
## 1 -87.46201 30.38968     1     1 alabama      <NA> Mississippi Flyway
## 2 -87.48493 30.37249     1     2 alabama      <NA> Mississippi Flyway
## 3 -87.52503 30.37249     1     3 alabama      <NA> Mississippi Flyway
## 4 -87.53076 30.33239     1     4 alabama      <NA> Mississippi Flyway
## 5 -87.57087 30.32665     1     5 alabama      <NA> Mississippi Flyway
## 6 -87.58806 30.32665     1     6 alabama      <NA> Mississippi Flyway
```

```r
print(table(us_states$Flyway))
```

```
##
##    Atlantic Flyway     Central Flyway Mississippi Flyway              Other
##               6074               2427               5218                 10
##     Pacific Flyway
##               1808
```

```r
# Plot the map
ggplot() +
  geom_polygon(data = us_states, aes(x = long, y = lat, group = group, fill = Flyway), color = "black")
  coord_fixed(1.3) +
  theme_void() +
  labs(title = "US States Grouped by Flyways with Color-coded Borders")
```

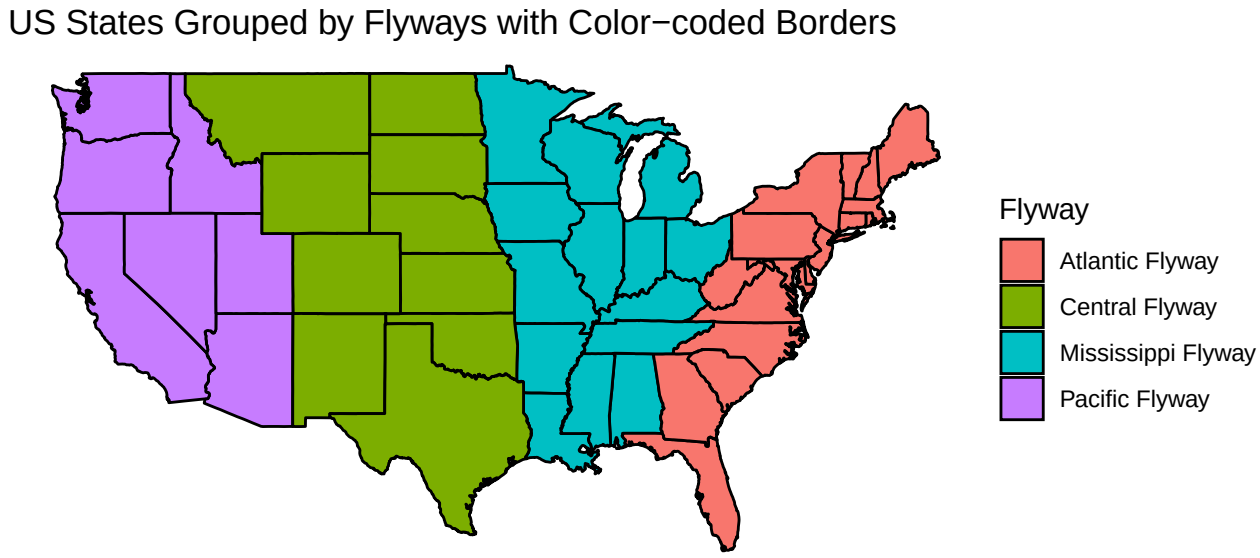



```r
# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))
```

```r
# Verify the updated data
print(table(us_states_filtered$Flyway))
```

```
##
##    Atlantic Flyway     Central Flyway Mississippi Flyway     Pacific Flyway
##               6074               2427               5218               1808
```

```r
# Plot the map
ggplot() +
  geom_polygon(data = us_states_filtered, aes(x = long, y = lat, group = group, fill = Flyway), color =
  coord_fixed(1.3) +
  theme_void() +
  labs(title = "US States Grouped by Flyways with Color-coded Borders")
```



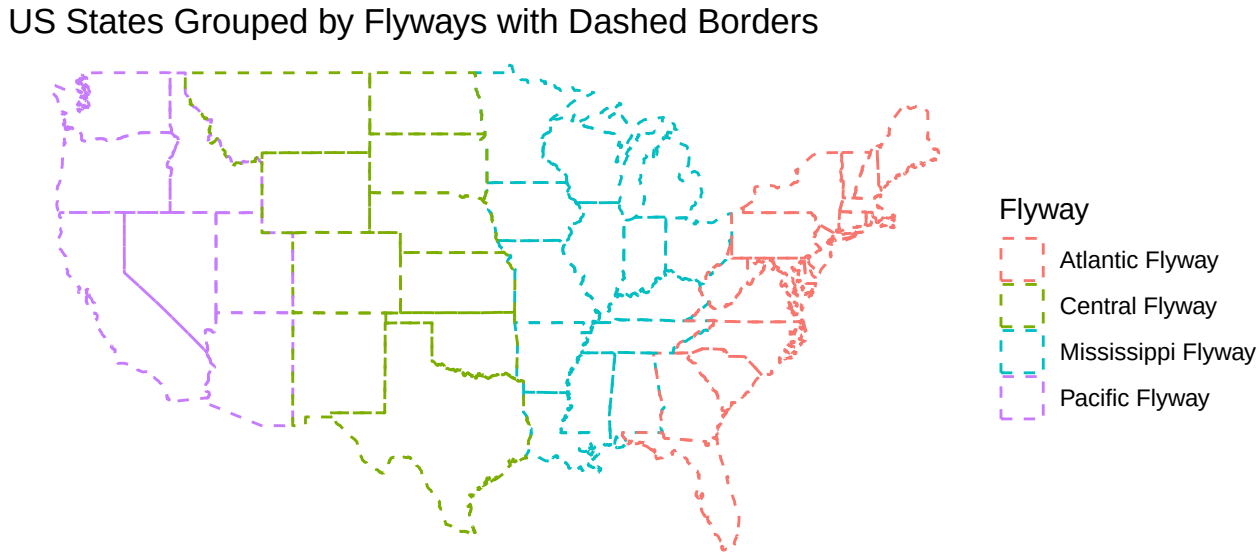
The previous codes worked. I liked the last one the best but now i want to color the states if i can so there will be a distinction between them.

```r
# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
```

```r
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))

# Verify the updated data
print(table(us_states_filtered$Flyway))
```

```
##
##    Atlantic Flyway     Central Flyway Mississippi Flyway     Pacific Flyway
##               6074               2427               5218               1808
```

```r
# Plot the map with dashed borders and no fill
ggplot() +
  geom_polygon(data = us_states_filtered, aes(x = long, y = lat, group = group, color = Flyway), fill =
  coord_fixed(1.3) +
  theme_void() +
  labs(title = "US States Grouped by Flyways with Dashed Borders")
```

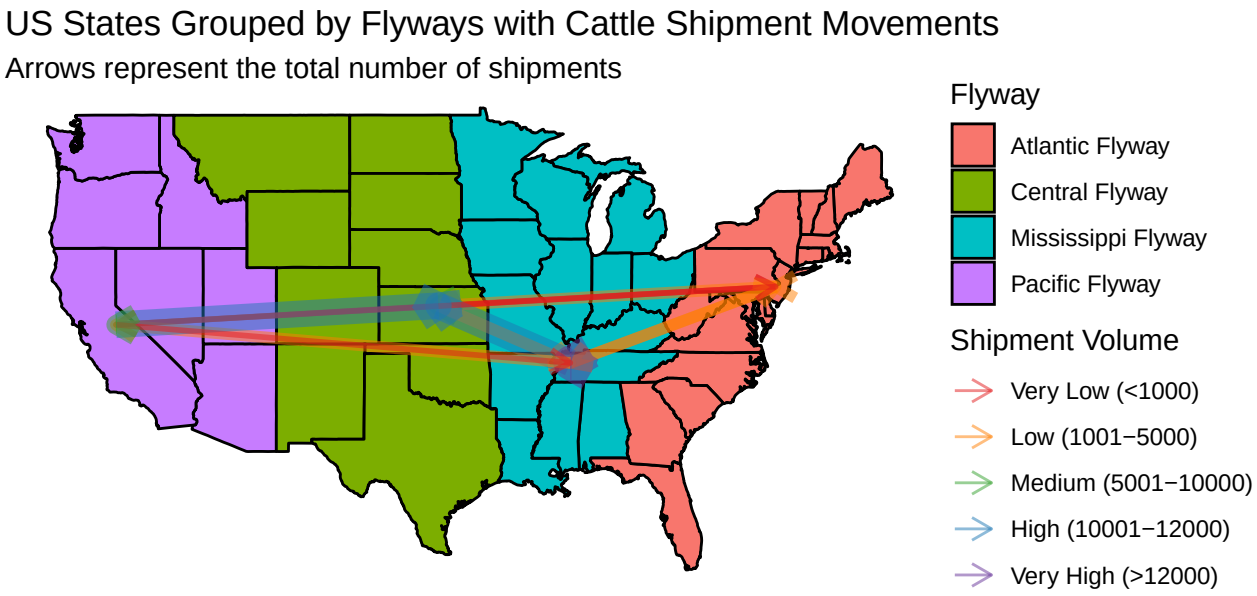


combine them now:

```
# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))
```

```r
# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.1,
  "Low (1001-5000)" = 0.5,
  "Medium (5001-10000)" = 1,
  "High (10001-12000)" = 1.5,
  "Very High (>12000)" = 2
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

# Plot the combined map
ggplot() +
  # Base map with flyways
  geom_polygon(data = us_states_filtered, aes(x = long, y = lat, group = group, fill = Flyway), color =
  # Shipment data
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth for arrow thickness
    ),
    arrow = arrow(length = unit(0.3, "cm")),
    show.legend = c(color = TRUE, linewidth = FALSE)  # Hide linewidth legend
  ) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  coord_fixed(1.3) +
  theme_void() +
  labs(
```

```
    title = "US States Grouped by Flyways with Cattle Shipment Movements",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume"
  ) +
  theme(legend.position = "right")
```



no color now

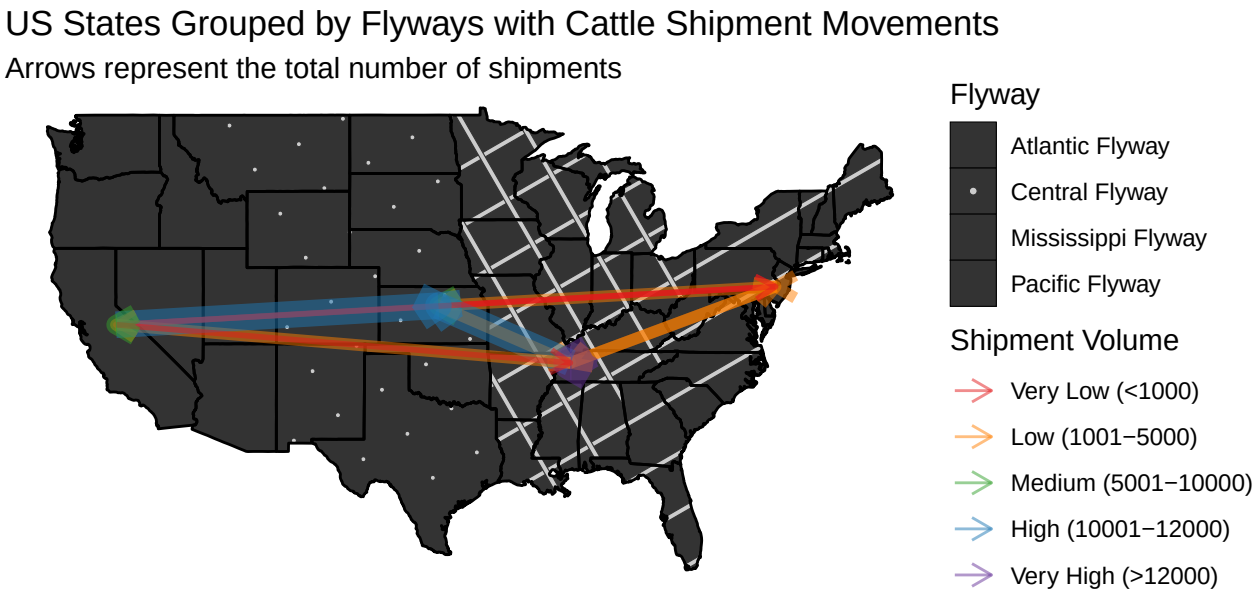
```
library(ggpattern)  # Load ggpattern for patterns
```

```
## Warning: package 'ggpattern' was built under R version 4.3.3
```

```
# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
```

```
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))


library(ggplot2)
library(ggpattern)

# Define pattern mapping for each Flyway
pattern_mapping <- c(
  "Atlantic Flyway" = "stripe",       # Diagonal stripes
  "Mississippi Flyway" = "crosshatch", # Crosshatch pattern
  "Central Flyway" = "circle",         # Circles
  "Pacific Flyway" = "none"            # No fill
)


ggplot() +
  # Base map with flyways using patterns
  geom_polygon_pattern(
    data = us_states_filtered,
    aes(x = long, y = lat, group = group, pattern = Flyway),
    color = "black",
    pattern_density = 0.1,    # Adjust density of patterns
    pattern_size = 0.2,      # Adjust size of patterns
    pattern_spacing = 0.1    # Adjust spacing of patterns
  ) +
  # Shipment data
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth for arrow thickness
    ),
    arrow = arrow(length = unit(0.3, "cm")),
    show.legend = c(color = TRUE, linewidth = FALSE)  # Hide linewidth legend
  ) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  scale_pattern_manual(values = pattern_mapping) +  # Apply pattern scale
  coord_fixed(1.3) +
  theme_void() +
  labs(
```

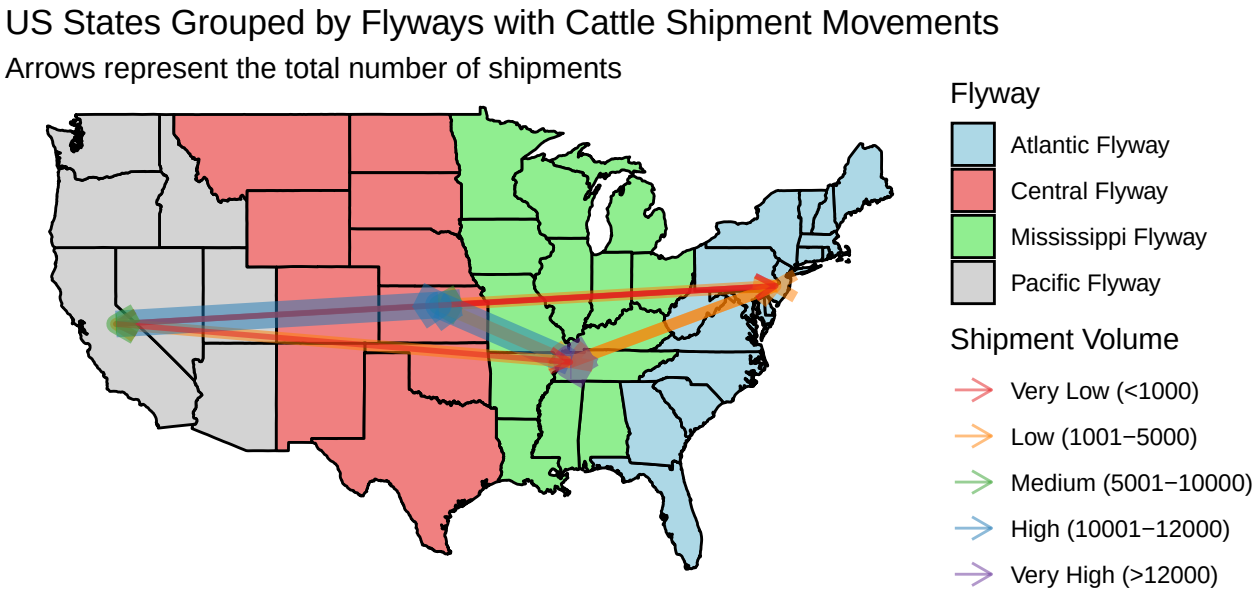
```
    title = "US States Grouped by Flyways with Cattle Shipment Movements",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume"
  ) +
  theme(legend.position = "right")
```




```
# Define fill colors for each Flyway
fill_colors <- c(
  "Atlantic Flyway" = "lightblue",      # Color for Atlantic Flyway
  "Mississippi Flyway" = "lightgreen",  # Color for Mississippi Flyway
  "Central Flyway" = "lightcoral",      # Color for Central Flyway
  "Pacific Flyway" = "lightgray"        # Color for Pacific Flyway
)

# Load US state map data
us_states <- map_data("state")

# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)
```

```r
# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))

# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),    # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),    # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Assuming movement_data is already defined and contains the required columns
# Example of how you might define it for completeness:
# movement_data <- data.frame(
#   lon_origin = runif(10, -125, -65),
#   lat_origin = runif(10, 25, 50),
#   lon_dest = runif(10, -125, -65),
#   lat_dest = runif(10, 25, 50),
#   Total_Shipments = sample(1:15000, 10),
#   Shipment_Category = cut(sample(1:15000, 10), breaks = c(0, 1000, 5000, 10000, 12000, Inf),
#                           labels = c("Very Low (<1000)", "Low (1001-5000)", "Medium (5001-10000)",
#                                      "High (10001-12000)", "Very High (>12000)")),
#   Arrow_Size = runif(10, 0.1, 2)
# )

# Plot the combined map with fill colors
ggplot() +
  # Base map with flyways using fill colors
  geom_polygon(
    data = us_states_filtered,
    aes(x = long, y = lat, group = group, fill = Flyway),
    color = "black",  # Border color
    size = 0.5        # Border size
  ) +
  # Shipment data
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
```

```
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth for arrow thickness
    ),
    arrow = arrow(length = unit(0.3, "cm")),
    show.legend = c(color = TRUE, linewidth = FALSE)  # Hide linewidth legend
  ) +
  scale_fill_manual(values = fill_colors) +  # Apply fill color scale
  scale_color_manual(values = custom_colors) +  # Apply custom color scale for shipments
  coord_fixed(1.3) +
  theme_void() +
  labs(
    title = "US States Grouped by Flyways with Cattle Shipment Movements",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume", fill = "Flyway"
  ) +
  theme(legend.position = "right")
```

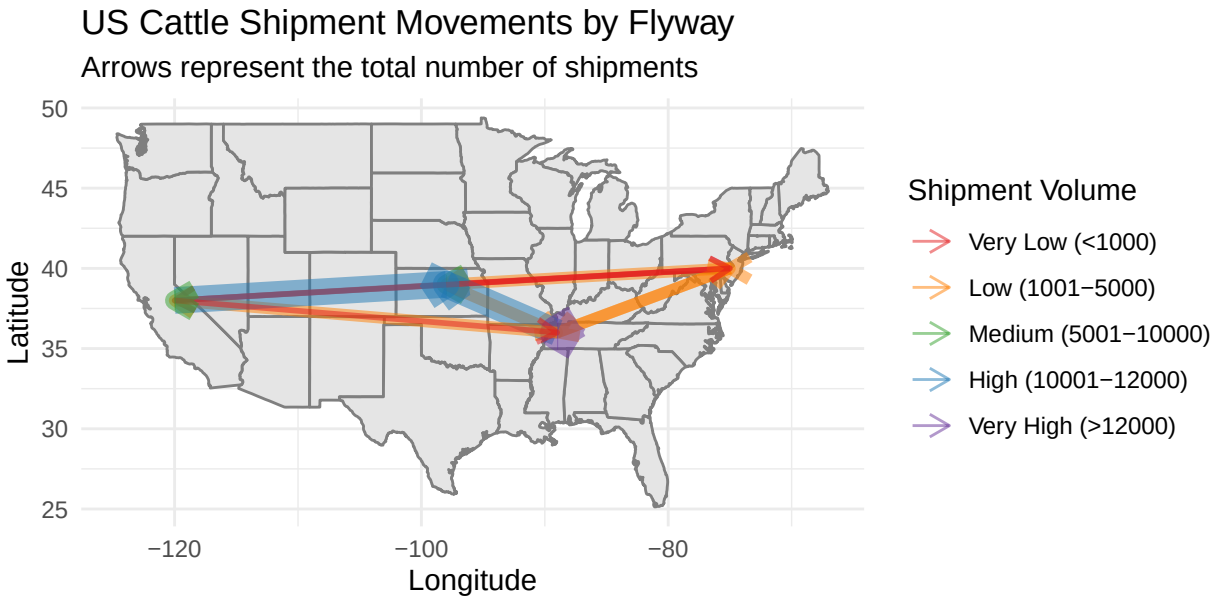


lets see this again and see if we can get them curved

```
library(scales) # To use the alpha function

# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
```

```
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.1,
  "Low (1001-5000)" = 0.5,
  "Medium (5001-10000)" = 1,
  "High (10001-12000)" = 1.5,
  "Very High (>12000)" = 2
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_segment(
    data = movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_dest,
      yend = lat_dest,
      color = Shipment_Category,
      linewidth = Arrow_Size  # Use linewidth instead of size
    ),
    arrow = arrow(length = unit(0.3, "cm")),
    show.legend = c(color = TRUE, linewidth = FALSE)  # Hide linewidth legend
  ) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  coord_fixed(1.3) +
  theme_minimal() +
  labs(
    title = "US Cattle Shipment Movements by Flyway",
    subtitle = "Arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume"
  ) +
  theme(legend.position = "right")
```

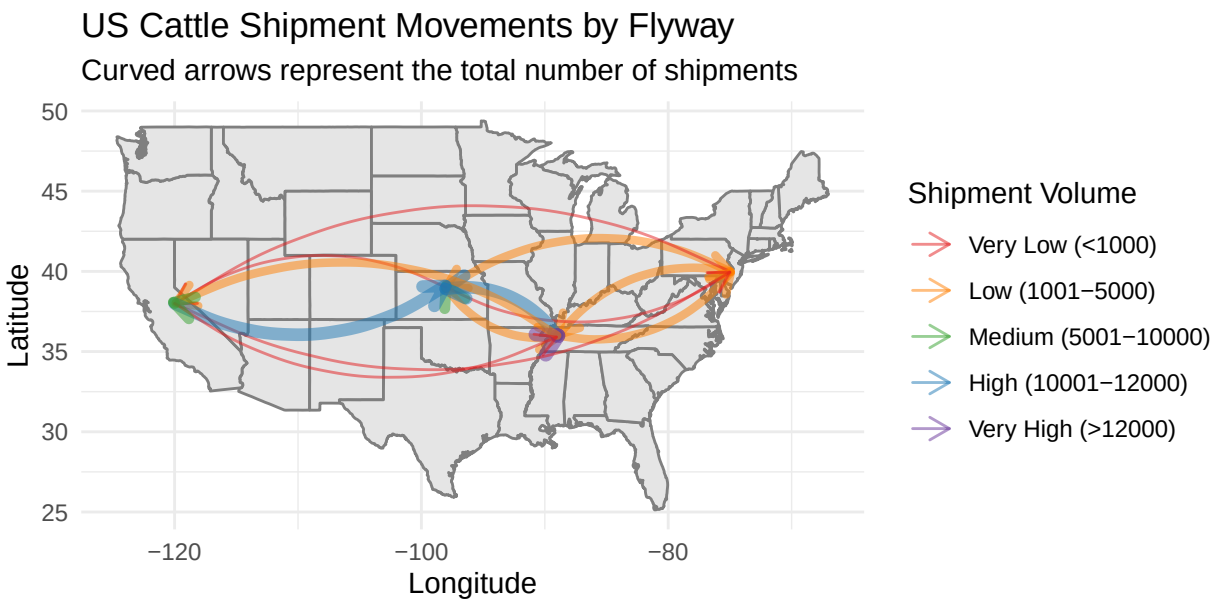
```
## Warning: Duplicated aesthetics after name standardisation: colour
```




```r
library(scales) # To use the alpha function

# Define custom color scale with transparency (alpha)
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.5,
  "Low (1001-5000)" = 1.0,
  "Medium (5001-10000)" = 1.5,
```

```
  "High (10001-12000)" = 2,
  "Very High (>12000)" = 2.5
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

# Slightly adjust endpoints to avoid identical points
adjusted_movement_data <- movement_data %>%
  mutate(
    lon_offset = lon_dest + runif(n(), -0.1, 0.1),
    lat_offset = lat_dest + runif(n(), -0.1, 0.1)
  )

ggplot() +
  borders("state", fill = "gray90", color = "white") +  # Base map
  geom_curve(
    data = adjusted_movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_offset,
      yend = lat_offset,
      color = Shipment_Category,
      size = Arrow_Size  # Use size for arrow thickness
    ),
    curvature = 0.3,  # Adjust curvature to make arrows bend
    arrow = arrow(length = unit(0.3, "cm")),
    lineend = "round",  # Smooth end of the curves
    show.legend = c(color = TRUE, size = FALSE)  # Hide size legend
  ) +
  scale_color_manual(values = custom_colors) +  # Apply custom color scale
  scale_size_continuous(range = c(0.5, 2.5)) +   # Set size range for arrows
  coord_fixed(1.3) +
  theme_minimal() +
  labs(
    title = "US Cattle Shipment Movements by Flyway",
    subtitle = "Curved arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume"
  ) +
  theme(legend.position = "right")
```

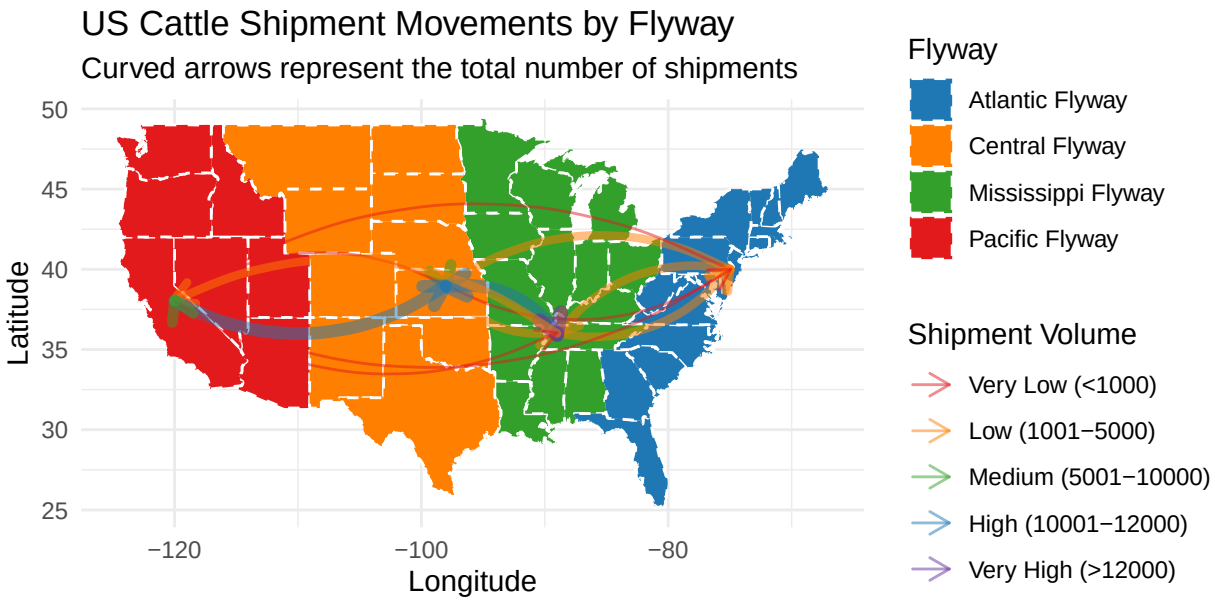
```
## Warning: Duplicated aesthetics after name standardisation: colour
```




```
library(scales) # To use the alpha function

# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
```

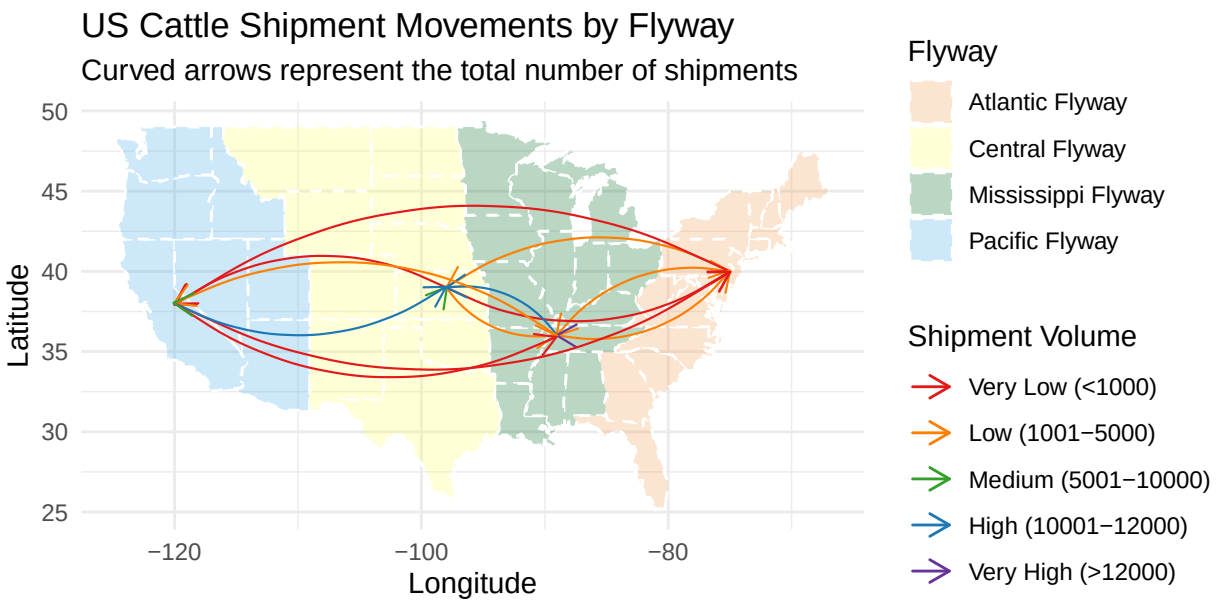
```r
  filter(!region %in% c("alaska", "hawaii"))

# Define custom color scale with transparency (alpha) for arrows
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c", 0.5),     # 50% opacity
  "Low (1001-5000)" = alpha("#ff7f00", 0.5),     # 50% opacity
  "Medium (5001-10000)" = alpha("#33a02c", 0.5), # 50% opacity
  "High (10001-12000)" = alpha("#1f78b4", 0.5),  # 50% opacity
  "Very High (>12000)" = alpha("#663399", 0.5)   # 50% opacity
)

# Define color scale for flyways
flyway_colors <- c(
  "Atlantic Flyway" = "#1f78b4",  # Blue
  "Mississippi Flyway" = "#33a02c", # Green
  "Central Flyway" = "#ff7f00",     # Orange
  "Pacific Flyway" = "#e31a1c"      # Red
)

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.5,
  "Low (1001-5000)" = 1.0,
  "Medium (5001-10000)" = 1.5,
  "High (10001-12000)" = 2,
  "Very High (>12000)" = 2.5
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

# Slightly adjust endpoints to avoid identical points
adjusted_movement_data <- movement_data %>%
  mutate(
    lon_offset = lon_dest + runif(n(), -0.1, 0.1),
    lat_offset = lat_dest + runif(n(), -0.1, 0.1)
  )

# Plot the map with dashed borders and curved arrows
ggplot() +
  geom_polygon(data = us_states_filtered, aes(x = long, y = lat, group = group, fill = Flyway), color =
  geom_curve(
    data = adjusted_movement_data,
```

```
  aes(
    x = lon_origin,
    y = lat_origin,
    xend = lon_offset,
    yend = lat_offset,
    color = Shipment_Category,
    size = Arrow_Size  # Use size for arrow thickness
  ),
  curvature = 0.3,  # Adjust curvature to make arrows bend
  arrow = arrow(length = unit(0.3, "cm")),
  lineend = "round",  # Smooth end of the curves
  show.legend = c(color = TRUE, size = FALSE)  # Hide size legend
) +
scale_fill_manual(values = flyway_colors) +  # Custom fill colors for flyways
scale_color_manual(values = custom_colors) +  # Custom color scale for shipments
scale_size_continuous(range = c(0.5, 2.5)) +   # Set size range for arrows
coord_fixed(1.3) +
theme_minimal() +
labs(
  title = "US Cattle Shipment Movements by Flyway",
  subtitle = "Curved arrows represent the total number of shipments",
  x = "Longitude", y = "Latitude", color = "Shipment Volume", fill = "Flyway"
) +
theme(legend.position = "right")
```



```r
library(scales) # To use the alpha function

# Define flyways with state abbreviations
atlantic_states <- c("CT", "DE", "FL", "GA", "ME", "MD", "MA", "NH", "NJ", "NY", "NC", "PA", "RI", "SC"
mississippi_states <- c("AL", "AR", "IN", "IL", "IA", "KY", "LA", "MI", "MN", "MS", "MO", "OH", "TN", "
central_states <- c("MT", "WY", "CO", "NM", "TX", "OK", "KS", "NE", "SD", "ND")
pacific_states <- c("AZ", "CA", "ID", "NV", "OR", "UT", "WA")

# Load US state map data
us_states <- map_data("state")

# Convert state names to lowercase for consistency
state_name_lower <- tolower(state.name)

# Map states to Flyways
us_states$Flyway <- case_when(
  us_states$region %in% state_name_lower[match(atlantic_states, state.abb)] ~ "Atlantic Flyway",
  us_states$region %in% state_name_lower[match(mississippi_states, state.abb)] ~ "Mississippi Flyway",
  us_states$region %in% state_name_lower[match(central_states, state.abb)] ~ "Central Flyway",
  us_states$region %in% state_name_lower[match(pacific_states, state.abb)] ~ "Pacific Flyway",
  TRUE ~ "Other"
)

# Filter out 'Other' and ensure Alaska and Hawaii are excluded
us_states_filtered <- us_states %>%
  filter(Flyway != "Other") %>%
  filter(!region %in% c("alaska", "hawaii"))

# Define custom color scale with transparency (alpha) for arrows
custom_colors <- c(
  "Very Low (<1000)" = alpha("#e31a1c"),
  "Low (1001-5000)" = alpha("#ff7f00"),
  "Medium (5001-10000)" = alpha("#33a02c"),
  "High (10001-12000)" = alpha("#1f78b4"),
  "Very High (>12000)" = alpha("#663399")
)

# Define color scale for flyways with transparency
flyway_colors <- alpha(c(
  "Atlantic Flyway" = "#F4A460",  # Blue
  "Mississippi Flyway" = "#117733", # Green
  "Central Flyway" = "#ff7",    # Orange
  "Pacific Flyway" = "#56B4E9"     # Red
), 0.3)  # Set transparency for flyway colors

# Categorize Total_Shipments into Shipment_Category
movement_data$Shipment_Category <- cut(
  movement_data$Total_Shipments,
  breaks = c(0, 1000, 5000, 10000, 12000, Inf),
  labels = c("Very Low (<1000)", "Low (1001-5000)",
             "Medium (5001-10000)", "High (10001-12000)",
             "Very High (>12000)"),
  right = TRUE
```

```r
)

# Define size mapping for each Shipment Category
size_mapping <- c(
  "Very Low (<1000)" = 0.1,
  "Low (1001-5000)" = .1,
  "Medium (5001-10000)" = .1,
  "High (10001-12000)" = .1,
  "Very High (>12000)" = .1
)

# Map sizes to categories
movement_data$Arrow_Size <- size_mapping[as.character(movement_data$Shipment_Category)]

# Slightly adjust endpoints to avoid identical points
adjusted_movement_data <- movement_data %>%
  mutate(
    lon_offset = lon_dest + runif(n(), -0.1, 0.1),
    lat_offset = lat_dest + runif(n(), -0.1, 0.1)
  )

# Plot the map with dashed borders and curved arrows
ggplot() +
  geom_polygon(data = us_states_filtered, aes(x = long, y = lat, group = group, fill = Flyway), color =
  geom_curve(
    data = adjusted_movement_data,
    aes(
      x = lon_origin,
      y = lat_origin,
      xend = lon_offset,
      yend = lat_offset,
      color = Shipment_Category,
      size = Arrow_Size  # Use size for arrow thickness
    ),
    curvature = 0.3,  # Adjust curvature to make arrows bend
    arrow = arrow(length = unit(0.3, "cm")),
    lineend = "round",  # Smooth end of the curves
    show.legend = c(color = TRUE, size = FALSE)  # Hide size legend
  ) +
  scale_fill_manual(values = flyway_colors) +  # Custom fill colors for flyways
  scale_color_manual(values = custom_colors) +  # Custom color scale for shipments
  scale_size_continuous(range = c(0.1, .5)) +   # Set size range for arrows
  coord_fixed(1.3) +
  theme_minimal() +
  labs(
    title = "US Cattle Shipment Movements by Flyway",
    subtitle = "Curved arrows represent the total number of shipments",
    x = "Longitude", y = "Latitude", color = "Shipment Volume", fill = "Flyway"
  ) +
  theme(legend.position = "right")
```



from before, I've identified the major exporters and importers of shipment I want to make 12 different maps. 5 individual maps for each of the major exporter states - I want to see where their shipments end up. for example, if CA is exporting shipments, I want a us map that uses arrows to show me where the shipments end up. 5 individual maps for each of the major importer states. 1 that shows all 5 major importer states. and 1 that shows all 5 major exporter states.

the major exporters are WI, CA, ID, AZ, and NY the major importers are TX, KS, NE, CA, and IA

lets make sure they are right though. major exporters

```
# Aggregate the data to find the total number of shipments sent from each state
exporter_states <- movements_data %>%
  group_by(Origin) %>%
  summarize(Total_Shipments = sum(`Counts of Shipment`, na.rm = TRUE)) %>%
  arrange(desc(Total_Shipments))

# Display the top 5 major exporter states
top_exporters <- head(exporter_states, 5)

# Print the result
print(top_exporters)
```

```
## # A tibble: 5 x 2
##   Origin Total_Shipments
##   <chr>            <dbl>
## 1 WI                8251
## 2 CA                8200
## 3 ID                4373
```

```
## 4 AZ                3359
## 5 NY                3307
```

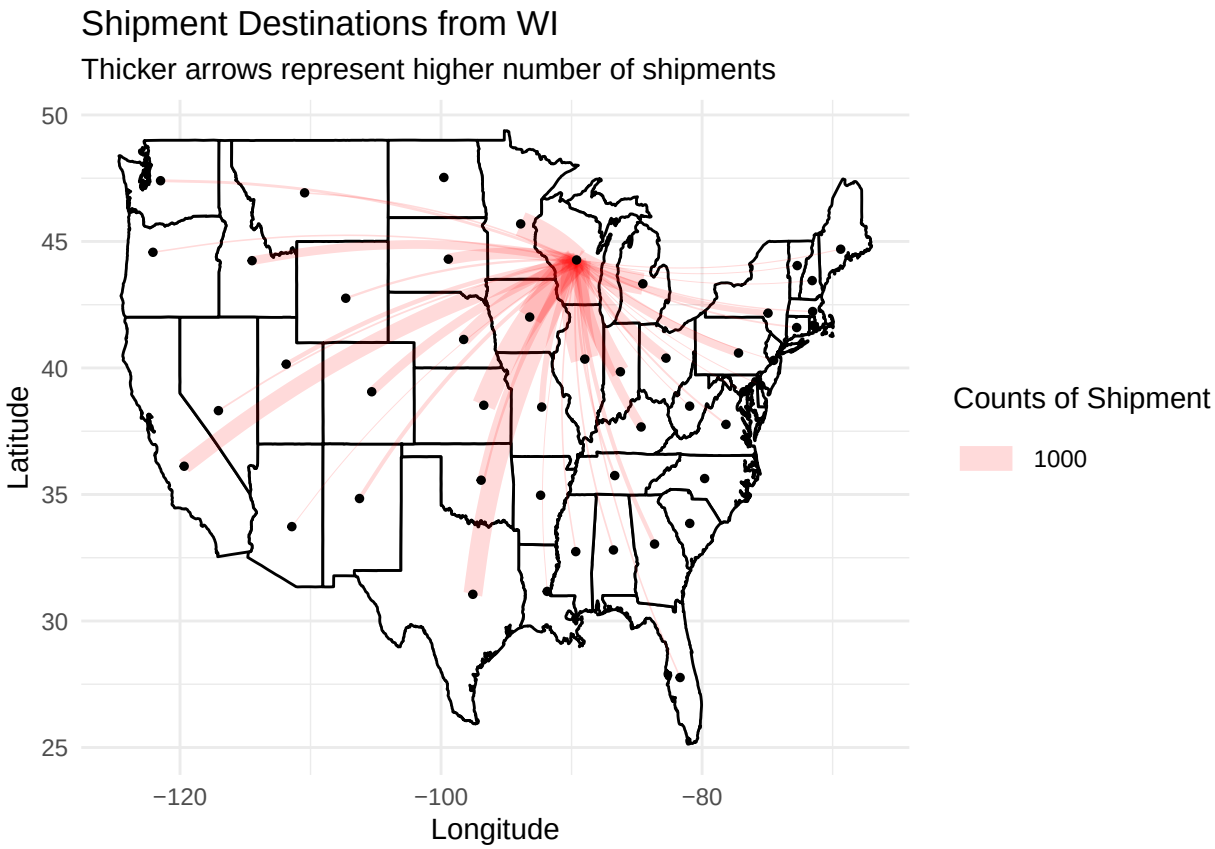
major importers:

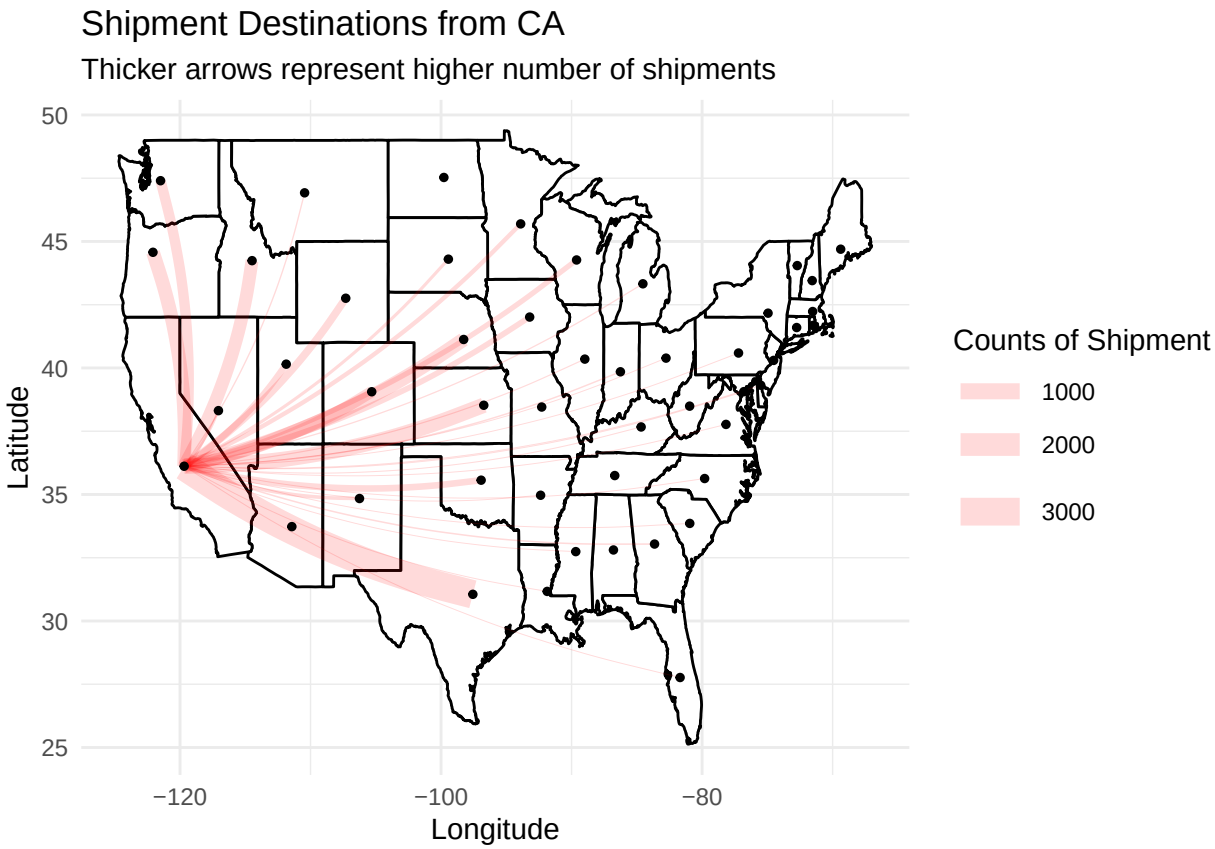
```
# Aggregate the data to find the total number of shipments received by each state
importer_states <- movements_data %>%
  group_by(Destination) %>%
  summarize(Total_Shipments = sum(`Counts of Shipment`, na.rm = TRUE)) %>%
  arrange(desc(Total_Shipments))

# Display the top 5 major importer states
top_importers <- head(importer_states, 5)

# Print the result
print(top_importers)
```

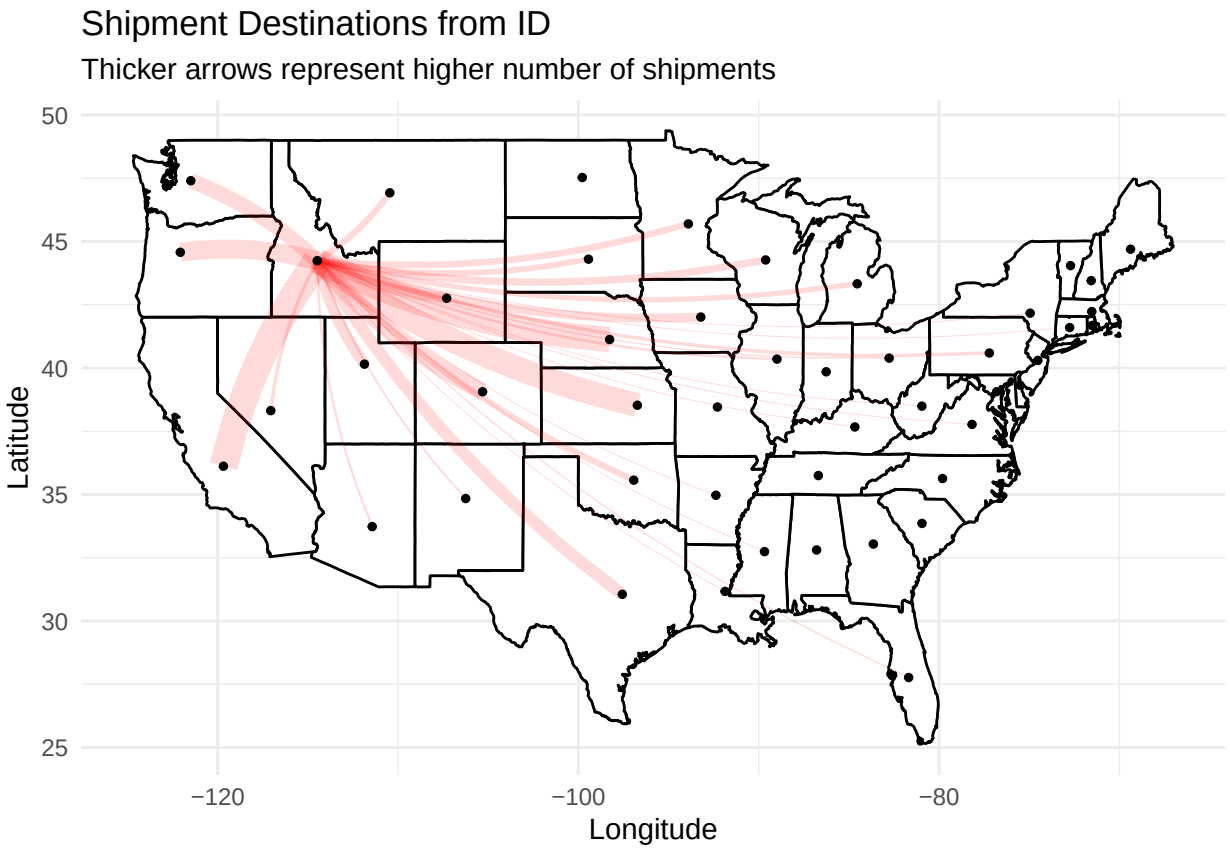
```
## # A tibble: 5 x 2
##   Destination Total_Shipments
##   <chr>                 <dbl>
## 1 TX                     9799
## 2 KS                     6831
## 3 NE                     6174
## 4 CA                     5950
## 5 IA                     4193
```

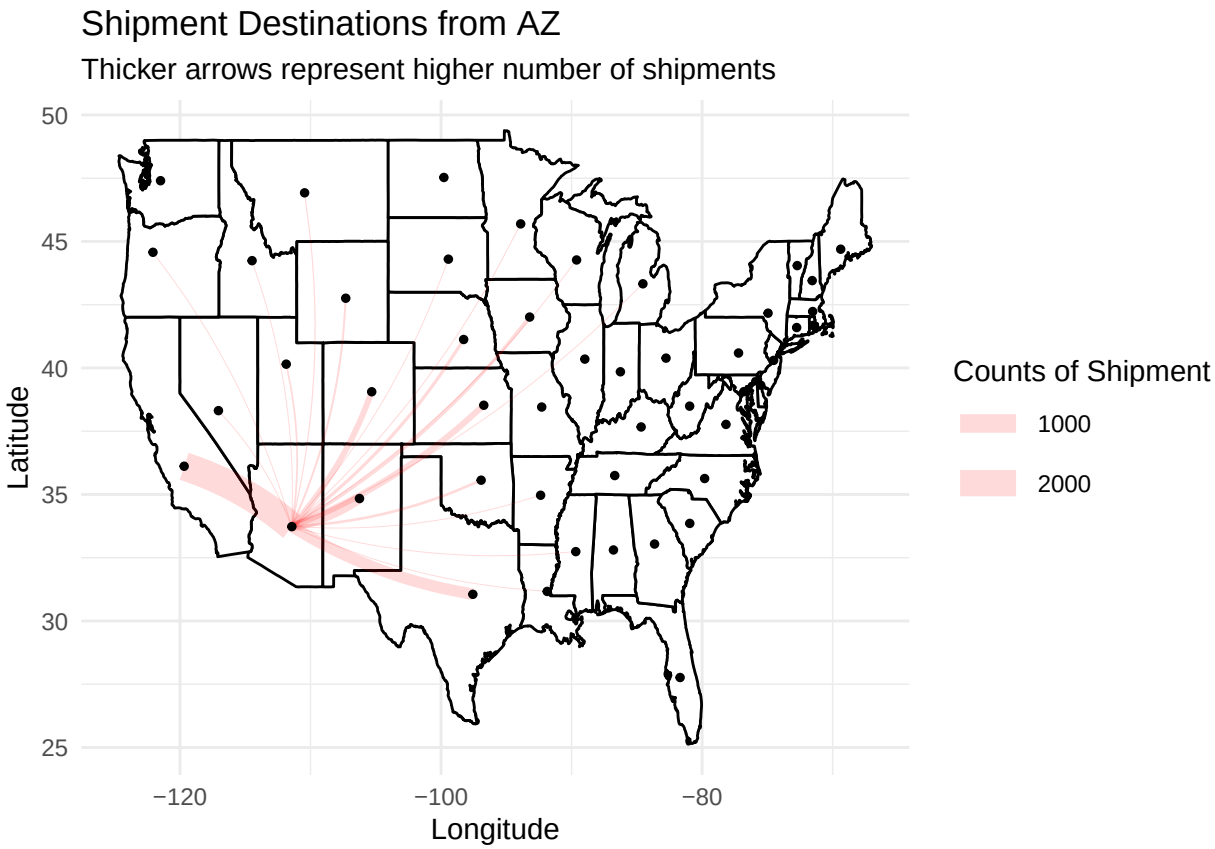
```
# Filter out Hawaii and Alaska from state_coords
state_coords_filtered <- state_coords %>%
  filter(!state %in% c("HAWAII", "ALASKA"))

# Filter out Hawaii and Alaska from state_coords based on abbreviations
state_coords_filtered <- state_coords %>%
  filter(!abbr %in% c("HI", "AK"))

library(ggplot2)
library(dplyr)
library(maps)

# Update the plotting function to exclude Hawaii and Alaska
plot_exporter_map <- function(exporter_state, movements_data, state_coords) {
  # Filter the data for the given exporter state
  filtered_data <- movements_data %>%
    filter(Origin == exporter_state) %>%
    # Join with state_coords to get coordinates
    left_join(state_coords, by = c("Origin" = "abbr")) %>%
    rename(Origin_lon = lon, Origin_lat = lat) %>%
    left_join(state_coords, by = c("Destination" = "abbr")) %>%
    rename(Destination_lon = lon, Destination_lat = lat) %>%
    # Remove rows with missing coordinates
    filter(!is.na(Origin_lon) & !is.na(Origin_lat) & !is.na(Destination_lon) & !is.na(Destination_lat))

  # Check if there is data to plot
  if (nrow(filtered_data) == 0) {
    cat("No valid data to plot for", exporter_state, "\n")
    return(NULL)
  }
```

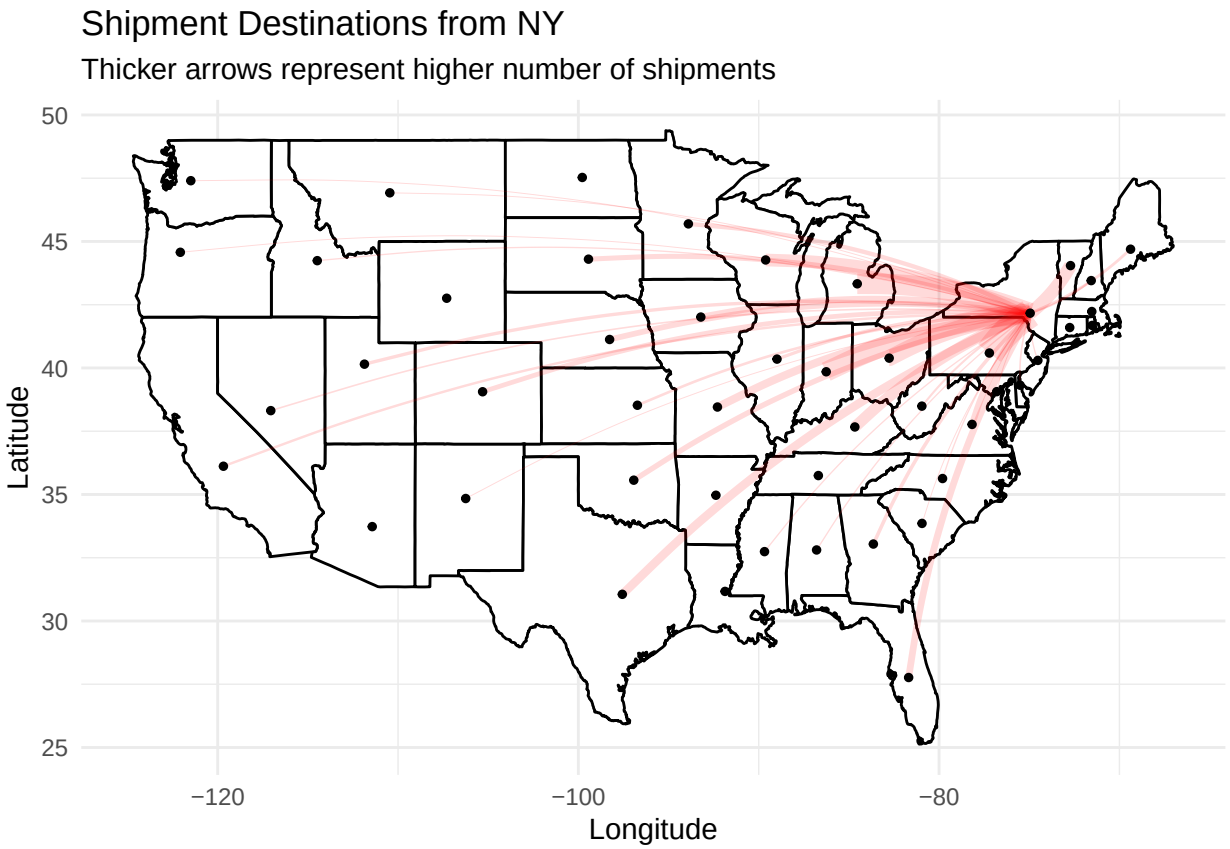
```r
  # Get US map data excluding Hawaii and Alaska
  us_map <- map_data("state") %>%
    filter(region != "hawaii" & region != "alaska")

  # Create the map plot
  p <- ggplot() +
    geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "white", color = "black") +
    geom_curve(data = filtered_data,
               aes(x = Origin_lon, y = Origin_lat,
                   xend = Destination_lon, yend = Destination_lat,
                   size = `Counts of Shipment`),
               curvature = 0.1,  # Adjust curvature as needed
               color = "red", alpha = 0.15) +  # Adjust alpha for transparency
    geom_point(data = state_coords, aes(x = lon, y = lat), color = "black", size = 1) +
    scale_size_continuous(range = c(0.01, 5),  # Adjust the range for more noticeable differences
                          breaks = c(1000, 2000, 3000),
                          labels = c("1000", "2000", "3000")) +  # Set custom breaks and labels
    labs(title = paste("Shipment Destinations from", exporter_state),
         subtitle = "Thicker arrows represent higher number of shipments",
         x = "Longitude",
         y = "Latitude") +
    theme_minimal() +
    theme(legend.position = "right")  # Move legend to the right

  # Print the plot
  print(p)
}

# List of major exporter states
exporter_states <- c("WI", "CA", "ID", "AZ", "NY")

# Plot maps for each major exporter state
for (state in exporter_states) {
  plot_exporter_map(state, movements_data, state_coords_filtered)
}
```


Shipment Destinations from WI
Thicker arrows represent higher number of shipments
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude
Counts of Shipment
1000


Shipment Destinations from CA
Thicker arrows represent higher number of shipments
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Counts of Shipment
1000
2000
3000


Shipment Destinations from ID
Thicker arrows represent higher number of shipments
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude

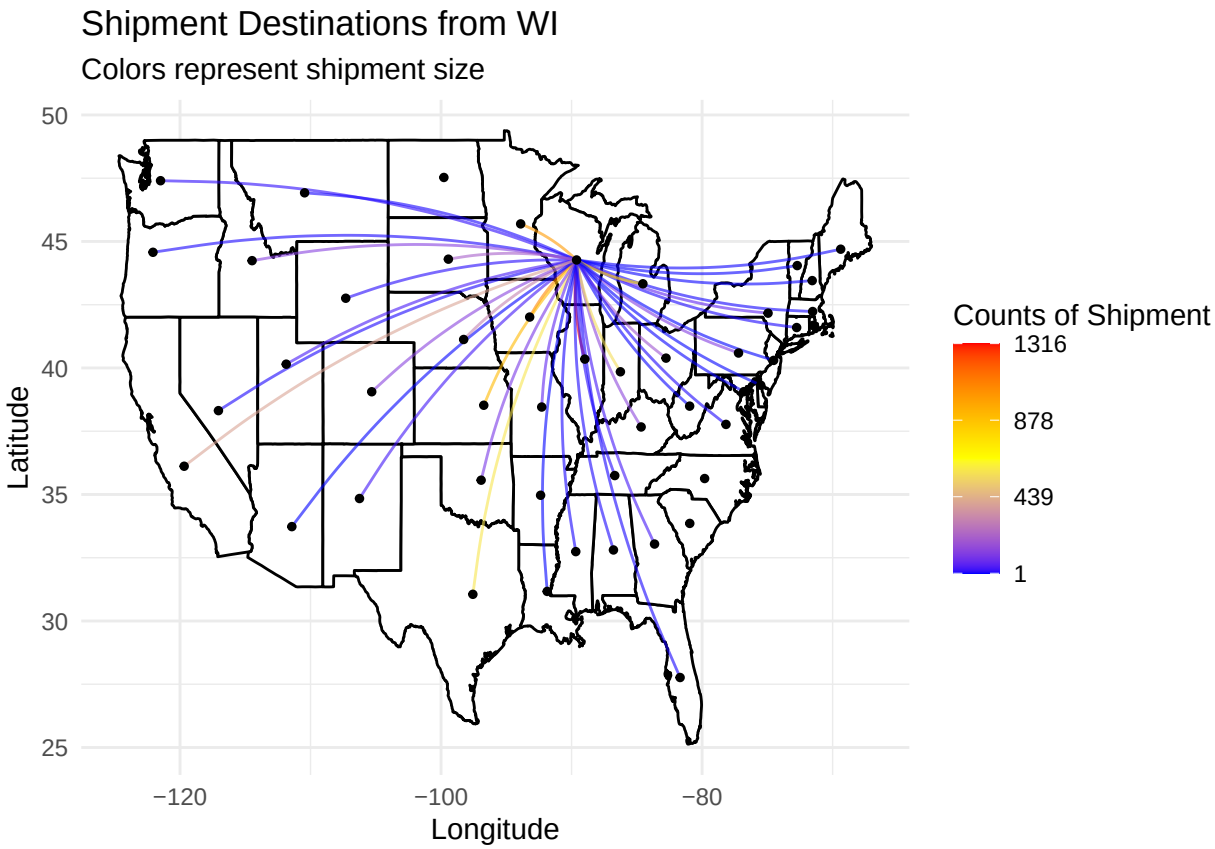

Shipment Destinations from AZ
Thicker arrows represent higher number of shipments
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude
Counts of Shipment
1000
2000

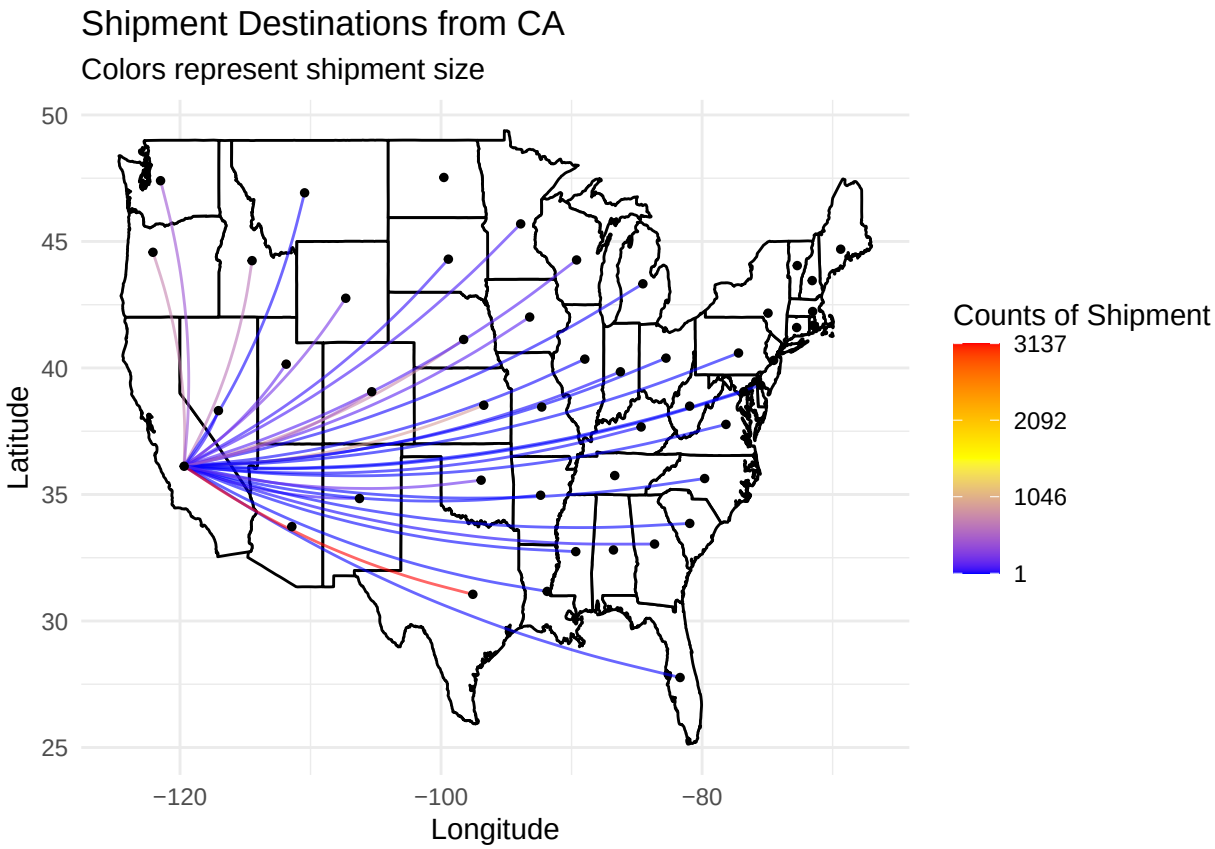


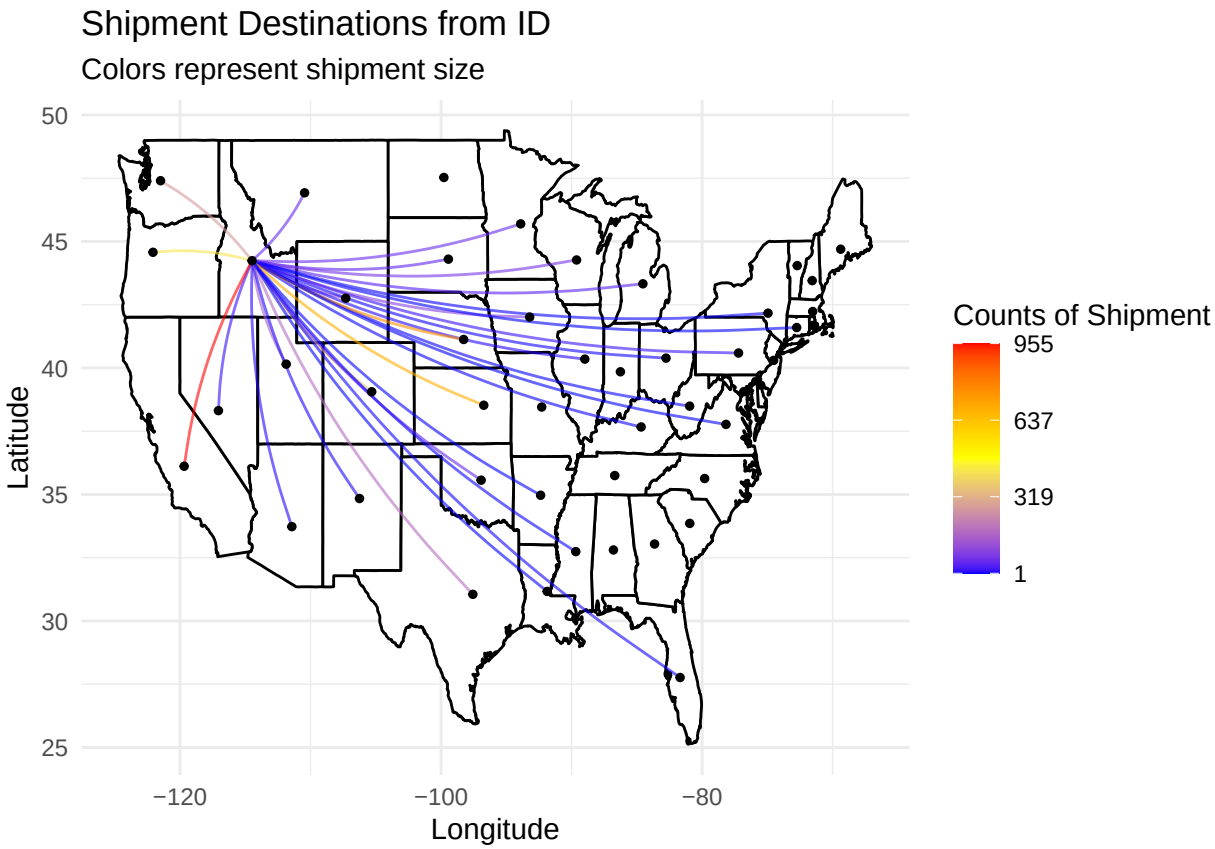
now instead of thickness, lets do 3 different colors:

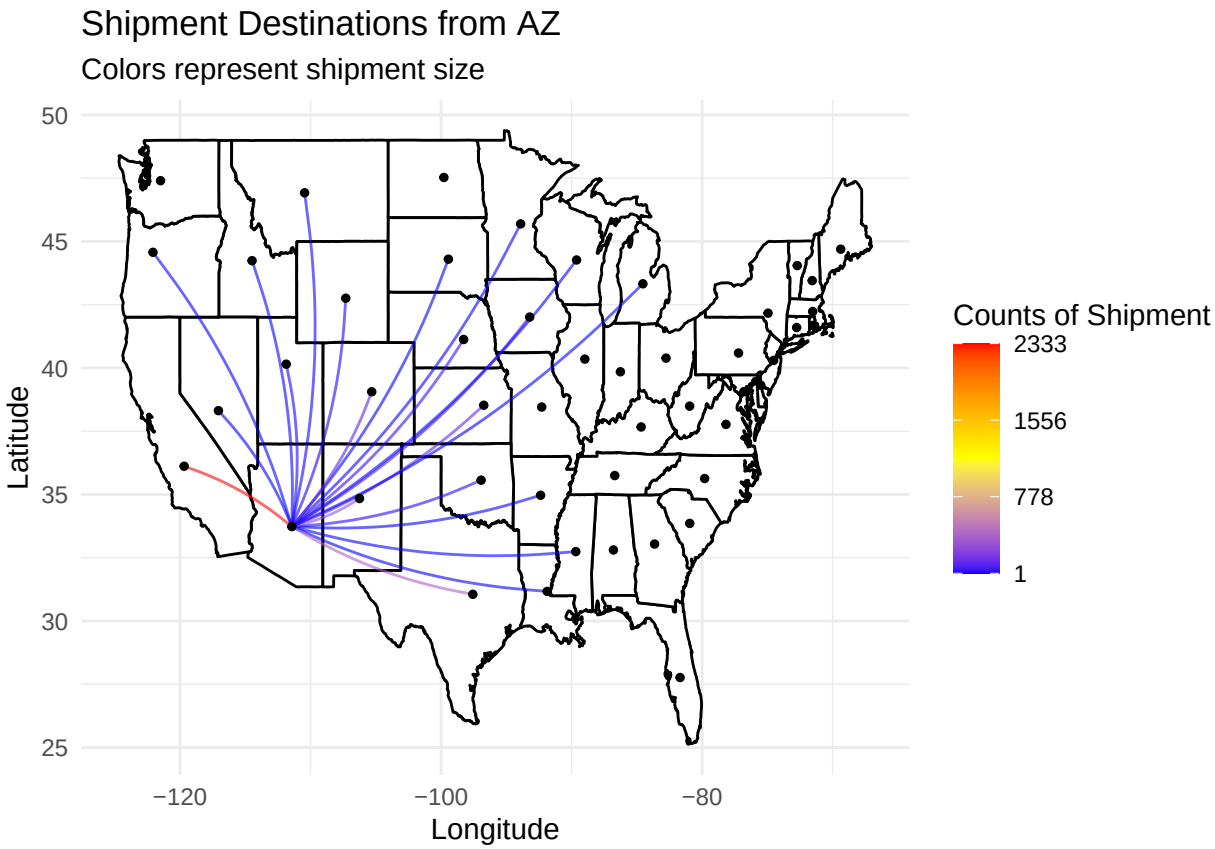
```
library(ggplot2)
library(dplyr)
library(maps)

# Function to plot shipment destinations for a given exporter state
plot_exporter_map <- function(exporter_state, movements_data, state_coords) {
  # Filter the data for the given exporter state
  filtered_data <- movements_data %>%
    filter(Origin == exporter_state) %>%
    # Join with state_coords to get coordinates
    left_join(state_coords, by = c("Origin" = "abbr")) %>%
    rename(Origin_lon = lon, Origin_lat = lat) %>%
    left_join(state_coords, by = c("Destination" = "abbr")) %>%
    rename(Destination_lon = lon, Destination_lat = lat) %>%
    # Remove rows with missing coordinates
    filter(!is.na(Origin_lon) & !is.na(Origin_lat) & !is.na(Destination_lon) & !is.na(Destination_lat))

  # Check if there is data to plot
  if (nrow(filtered_data) == 0) {
    cat("No valid data to plot for", exporter_state, "\n")
    return(NULL)
  }

  # Get US map data excluding Hawaii and Alaska
  us_map <- map_data("state") %>%
```

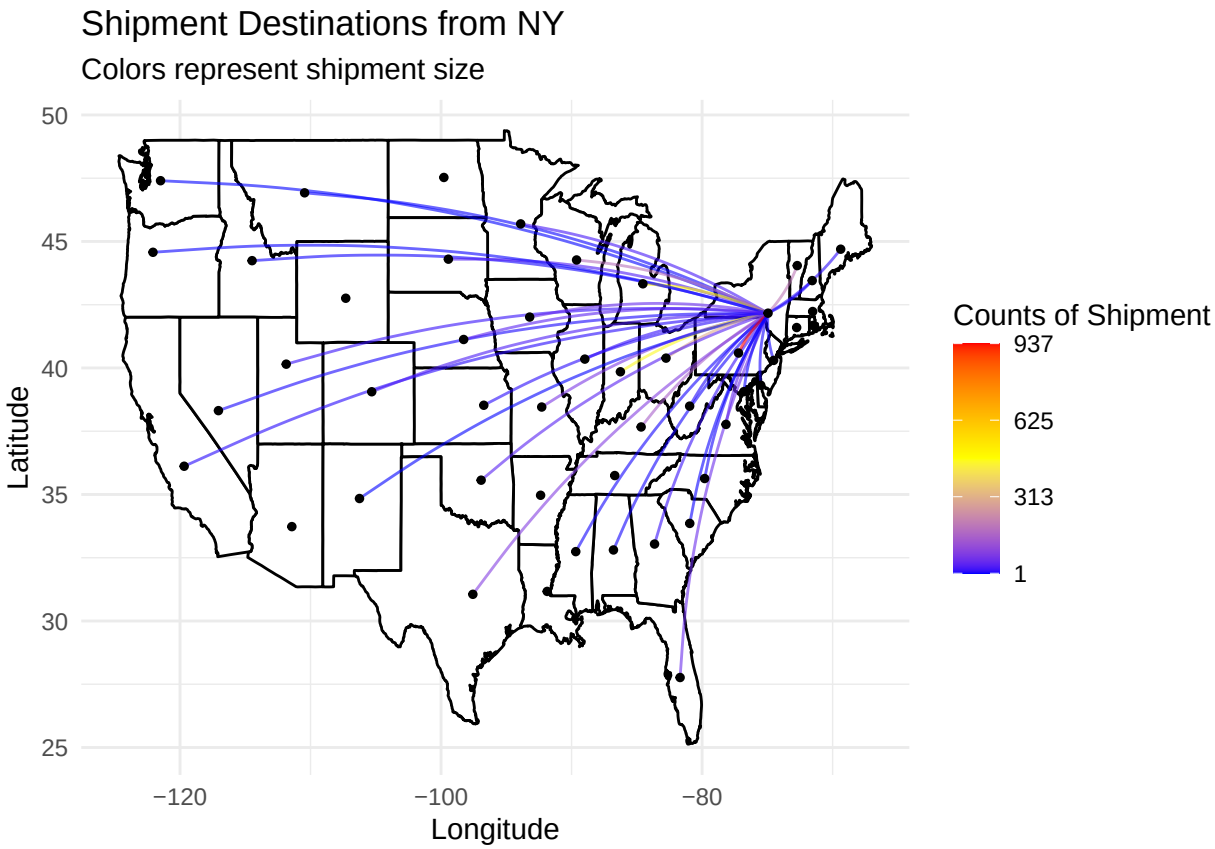
```
    filter(region != "hawaii" & region != "alaska")

  # Determine the range for shipment sizes
  shipment_range <- range(filtered_data$`Counts of Shipment`, na.rm = TRUE)

  # Define color breaks and labels
  color_breaks <- seq(shipment_range[1], shipment_range[2], length.out = 4)
  color_labels <- paste0(round(color_breaks, 0))

  # Create the map plot with color representing shipment size
  p <- ggplot() +
    geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "white", color = "black")
    geom_curve(data = filtered_data,
               aes(x = Origin_lon, y = Origin_lat,
                   xend = Destination_lon, yend = Destination_lat,
                   color = `Counts of Shipment`),
               curvature = 0.1,  # Adjust curvature as needed
               alpha = 0.6) +  # Adjust alpha for transparency
    scale_color_gradientn(colors = c("blue", "yellow", "red"),
                          breaks = color_breaks,
                          labels = color_labels) +
    geom_point(data = state_coords, aes(x = lon, y = lat), color = "black", size = 1) +
    labs(title = paste("Shipment Destinations from", exporter_state),
         subtitle = "Colors represent shipment size",
         x = "Longitude",
         y = "Latitude",
         color = "Counts of Shipment") +
    theme_minimal() +
    theme(legend.position = "right")  # Move legend to the right

  # Print the plot
  print(p)
}

# List of major exporter states
exporter_states <- c("WI", "CA", "ID", "AZ", "NY")

# Plot maps for each major exporter state
for (state in exporter_states) {
  plot_exporter_map(state, movements_data, state_coords_filtered)
}
```


Shipment Destinations from WI
Colors represent shipment size
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Counts of Shipment
1316
878
439
1


Shipment Destinations from CA
Colors represent shipment size
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Counts of Shipment
3137
2092
1046
1


Shipment Destinations from ID
Colors represent shipment size
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Counts of Shipment
955
637
319
1

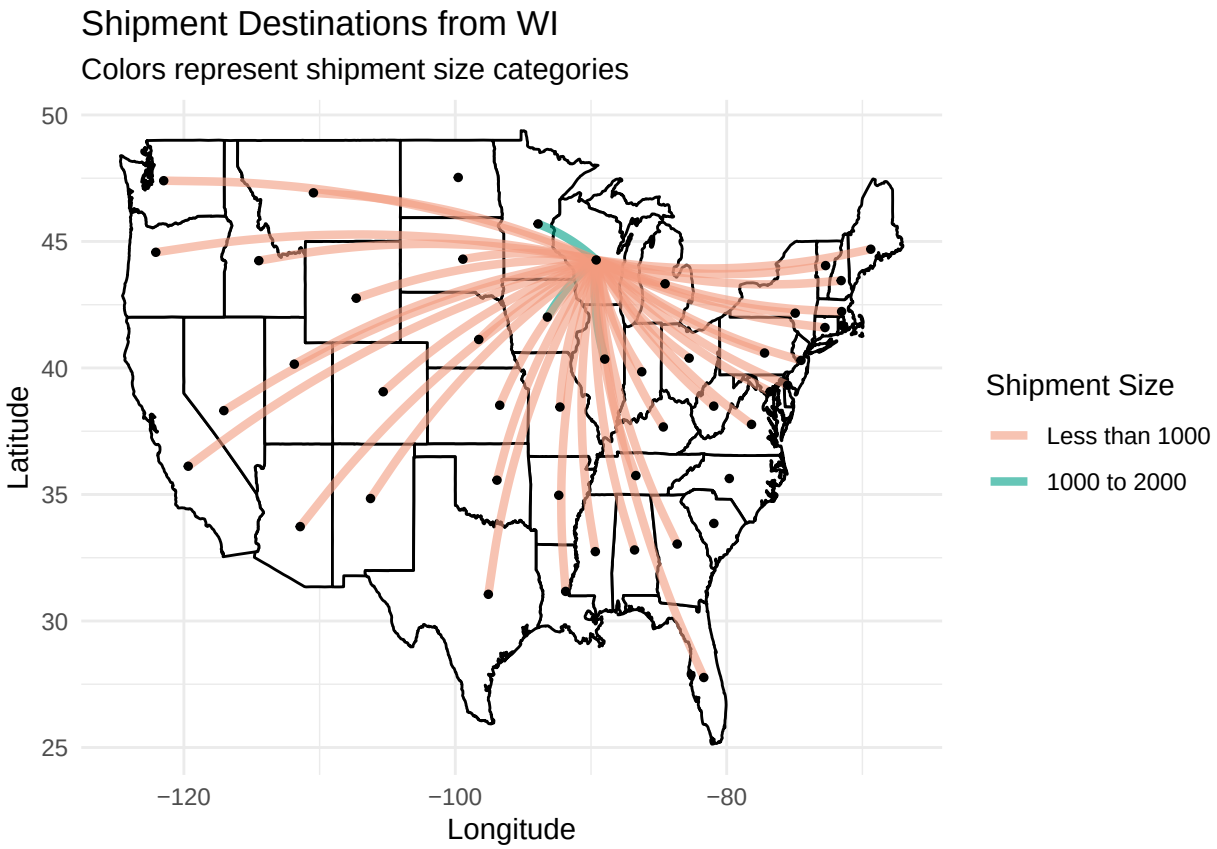

Shipment Destinations from AZ
Colors represent shipment size
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Counts of Shipment
2333
1556
778
1

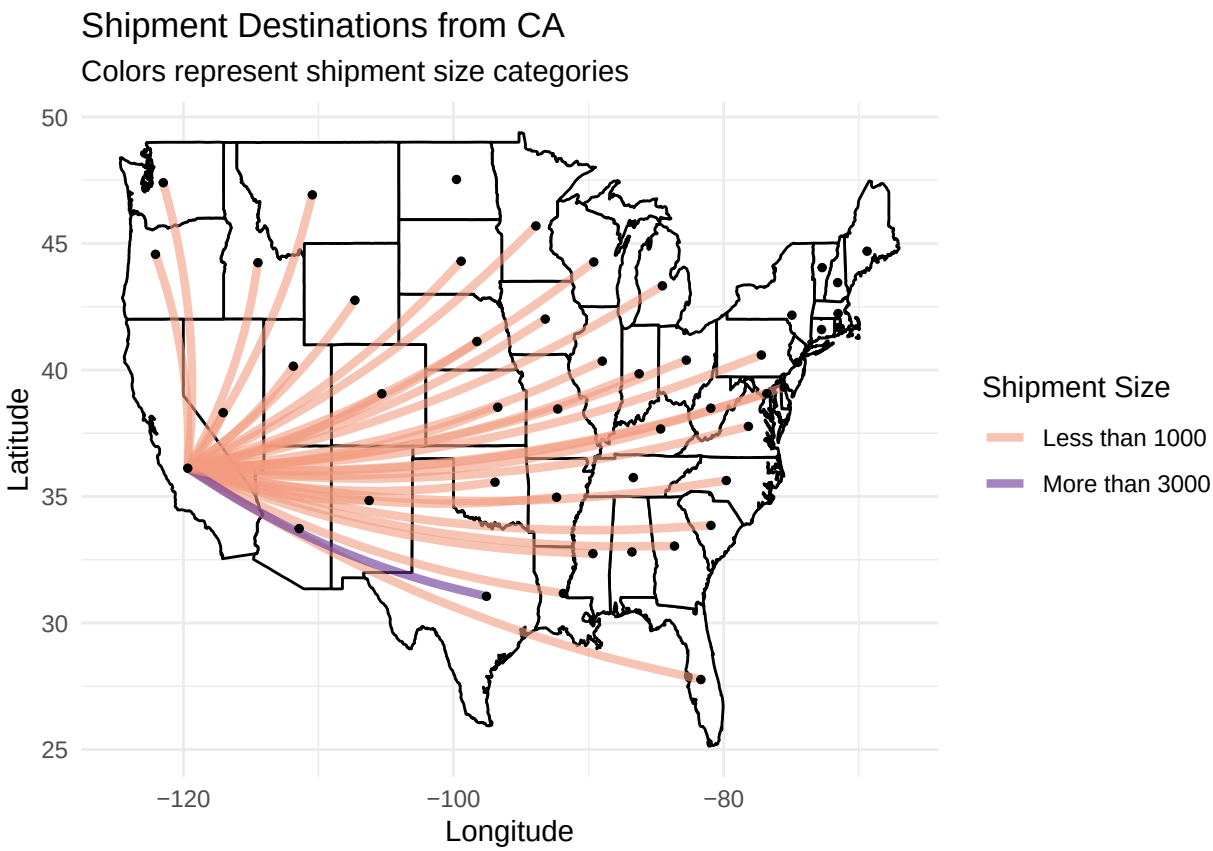


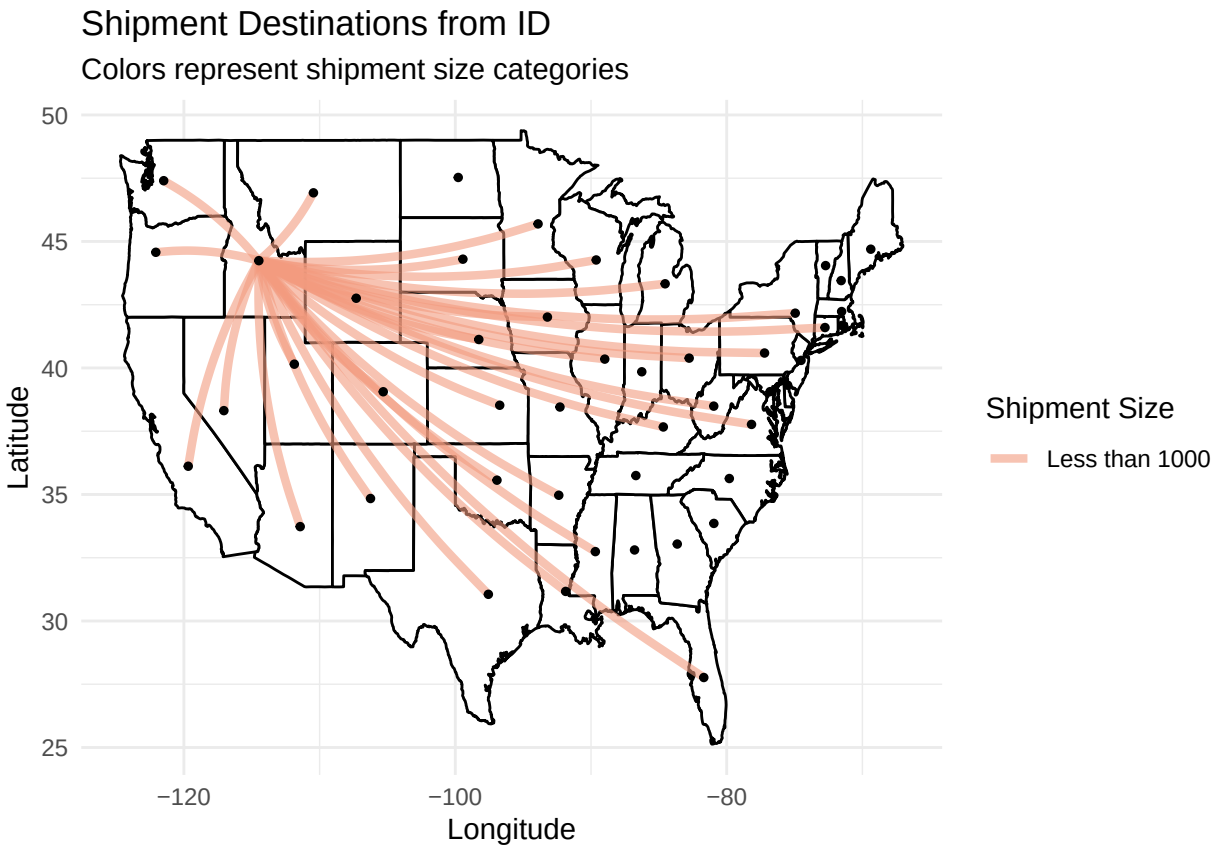
this is rainbow. lets do 3 distinct colors

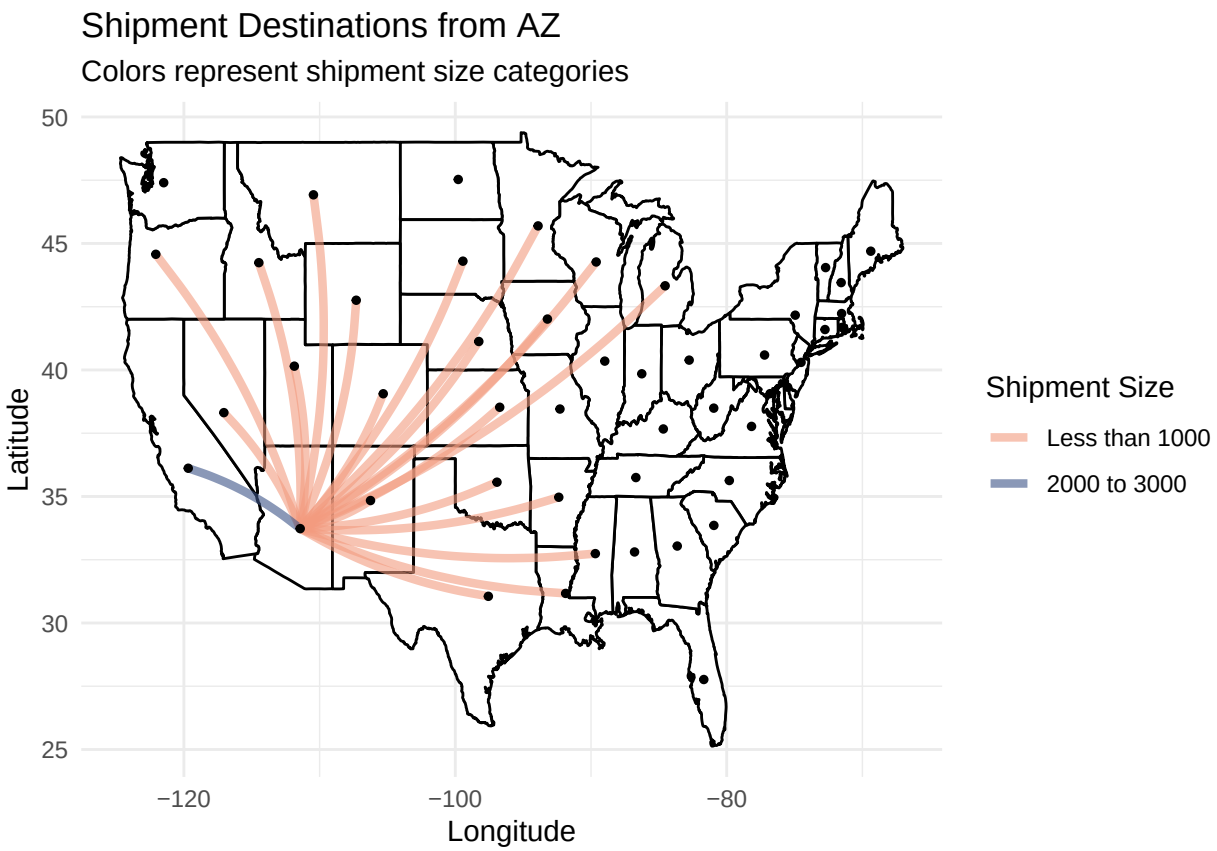
```
# Function to plot shipment destinations for a given exporter state
plot_exporter_map <- function(exporter_state, movements_data, state_coords) {
  # Filter the data for the given exporter state
  filtered_data <- movements_data %>%
    filter(Origin == exporter_state) %>%
    # Join with state_coords to get coordinates
    left_join(state_coords, by = c("Origin" = "abbr")) %>%
    rename(Origin_lon = lon, Origin_lat = lat) %>%
    left_join(state_coords, by = c("Destination" = "abbr")) %>%
    rename(Destination_lon = lon, Destination_lat = lat) %>%
    # Remove rows with missing coordinates
    filter(!is.na(Origin_lon) & !is.na(Origin_lat) & !is.na(Destination_lon) & !is.na(Destination_lat))

  # Check if there is data to plot
  if (nrow(filtered_data) == 0) {
    cat("No valid data to plot for", exporter_state, "\n")
    return(NULL)
  }

  # Get US map data excluding Hawaii and Alaska
  us_map <- map_data("state") %>%
    filter(region != "hawaii" & region != "alaska")

  # Define shipment size categories
  filtered_data <- filtered_data %>%
```

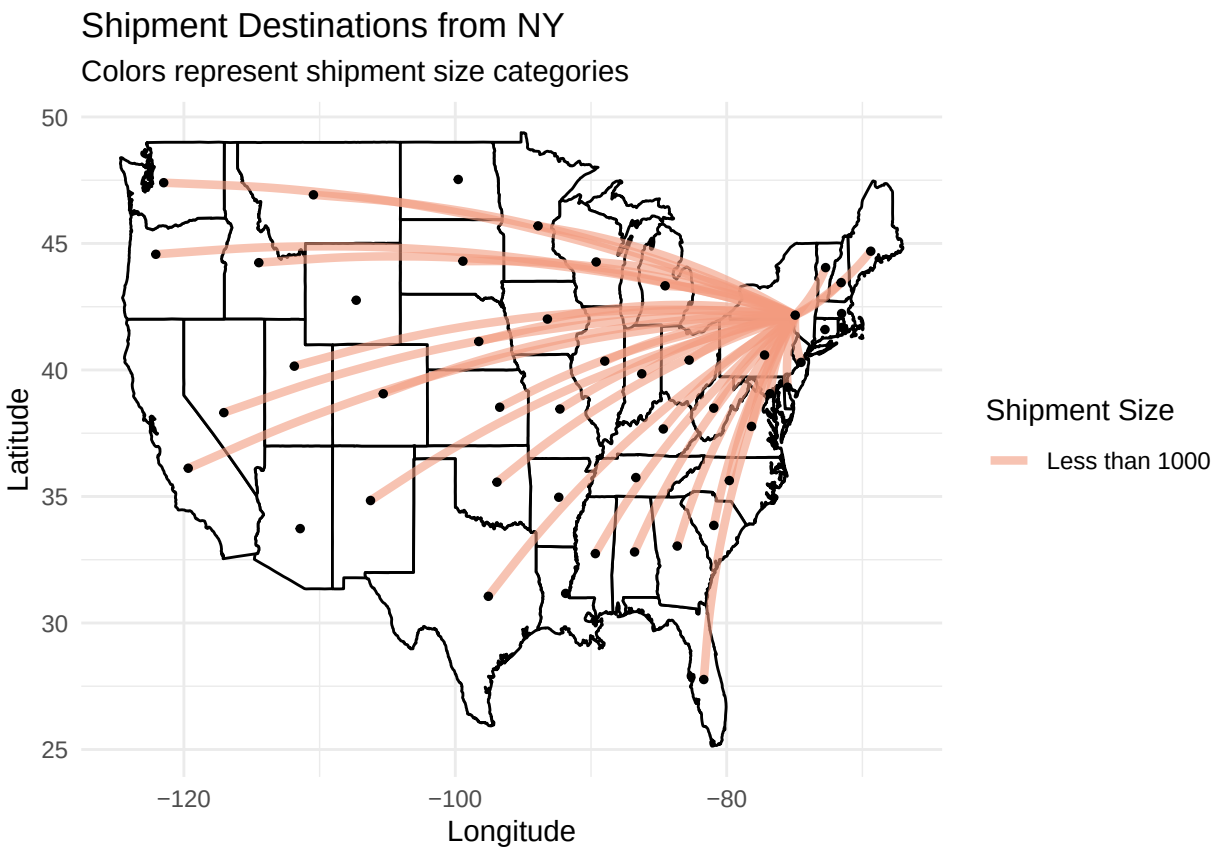
```
    mutate(Shipment_Category = cut(`Counts of Shipment`,
                                   breaks = c(-Inf, 1000, 2000, 3000, Inf),
                                   labels = c("Less than 1000", "1000 to 2000", "2000 to 3000", "More t

  # Define colors for each category
  color_palette <- c("Less than 1000" = "#F39B7FFF",
                     "1000 to 2000" = "#00A087FF",
                     "2000 to 3000" = "#3C5488FF",
                     "More than 3000" = "#663399")

  # Create the map plot with color representing shipment size categories
  p <- ggplot() +
    geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "white", color = "black")
    geom_curve(data = filtered_data,
               aes(x = Origin_lon, y = Origin_lat,
                   xend = Destination_lon, yend = Destination_lat,
                   color = Shipment_Category),
               curvature = 0.1,  # Adjust curvature as needed
               size = 1.5,  # Fixed line thickness
               alpha = 0.6) +  # Adjust alpha for transparency
    scale_color_manual(values = color_palette) +
    geom_point(data = state_coords, aes(x = lon, y = lat), color = "black", size = 1) +
    labs(title = paste("Shipment Destinations from", exporter_state),
         subtitle = "Colors represent shipment size categories",
         x = "Longitude",
         y = "Latitude",
         color = "Shipment Size") +
    theme_minimal() +
    theme(legend.position = "right")  # Move legend to the right

  # Print the plot
  print(p)
}

# List of major exporter states
exporter_states <- c("WI", "CA", "ID", "AZ", "NY")

# Plot maps for each major exporter state
for (state in exporter_states) {
  plot_exporter_map(state, movements_data, state_coords_filtered)
}
```


Shipment Destinations from WI
Colors represent shipment size categories
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude
Shipment Size
Less than 1000
1000 to 2000


Shipment Destinations from CA
Colors represent shipment size categories
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude
Shipment Size
Less than 1000
More than 3000

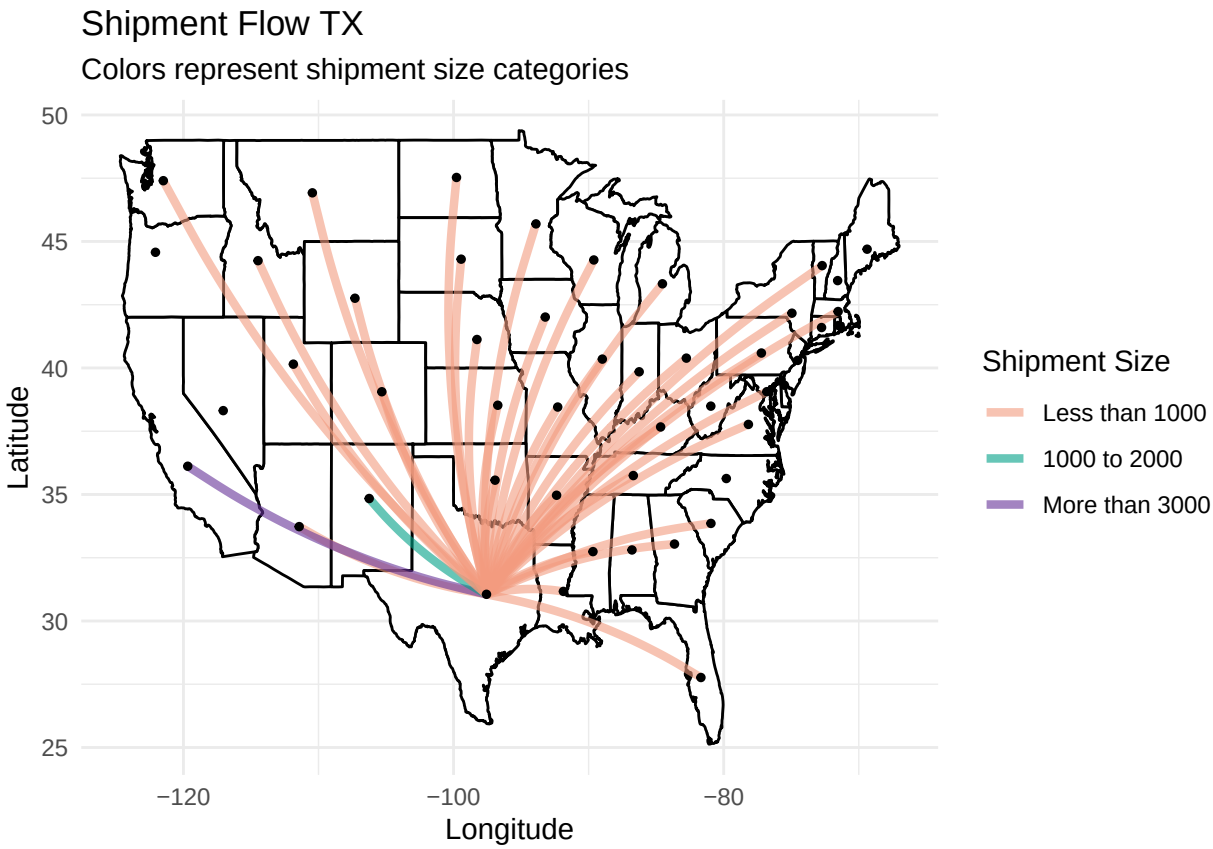

Shipment Destinations from ID
Colors represent shipment size categories
50
45
40
35
30
25
Latitude
−120
−100
−80
Longitude
Shipment Size
Less than 1000

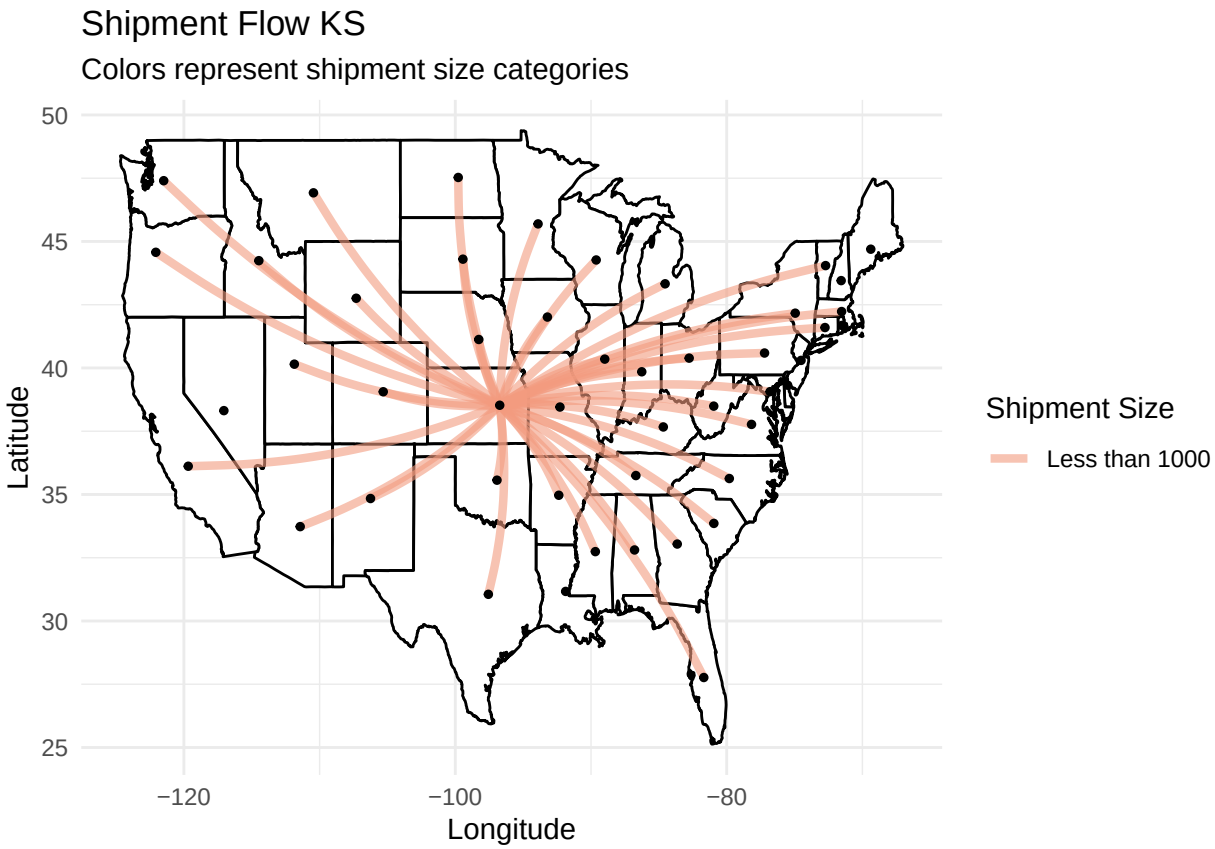

Shipment Destinations from AZ
Colors represent shipment size categories
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Shipment Size
Less than 1000
2000 to 3000

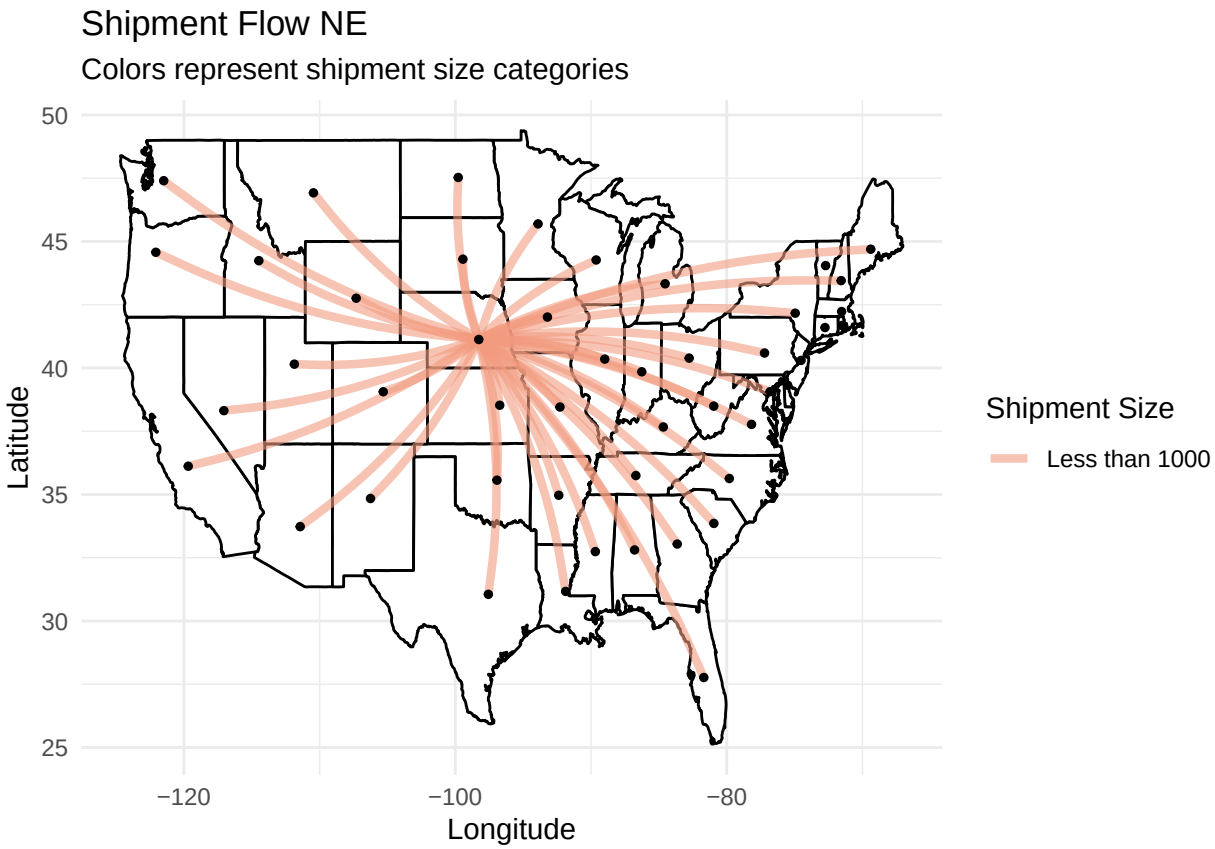


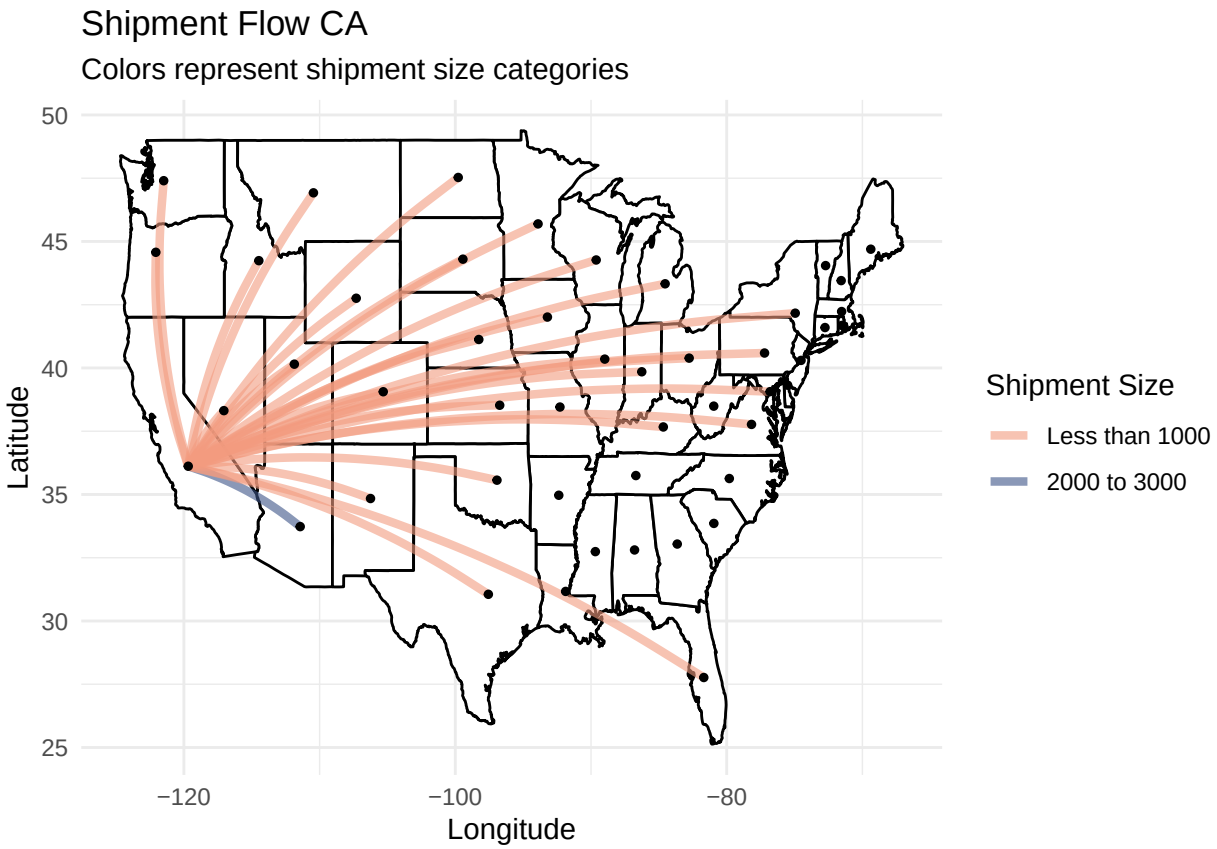

```
# List of major importer states based on your data
importer_states <- c("TX", "KS", "NE", "CA", "IA")

# Function to plot shipment origins for a given importer state
plot_importer_map <- function(importer_state, movements_data, state_coords) {
  # Filter the data for the given importer state
  filtered_data <- movements_data %>%
    filter(Destination == importer_state) %>%
    # Join with state_coords to get coordinates
    left_join(state_coords, by = c("Origin" = "abbr")) %>%
    rename(Origin_lon = lon, Origin_lat = lat) %>%
    left_join(state_coords, by = c("Destination" = "abbr")) %>%
    rename(Destination_lon = lon, Destination_lat = lat) %>%
    # Remove rows with missing coordinates
    filter(!is.na(Origin_lon) & !is.na(Origin_lat) & !is.na(Destination_lon) & !is.na(Destination_lat))

  # Check if there is data to plot
  if (nrow(filtered_data) == 0) {
    cat("No valid data to plot for", importer_state, "\n")
    return(NULL)
  }

  # Get US map data excluding Hawaii and Alaska
  us_map <- map_data("state") %>%
    filter(region != "hawaii" & region != "alaska")
```

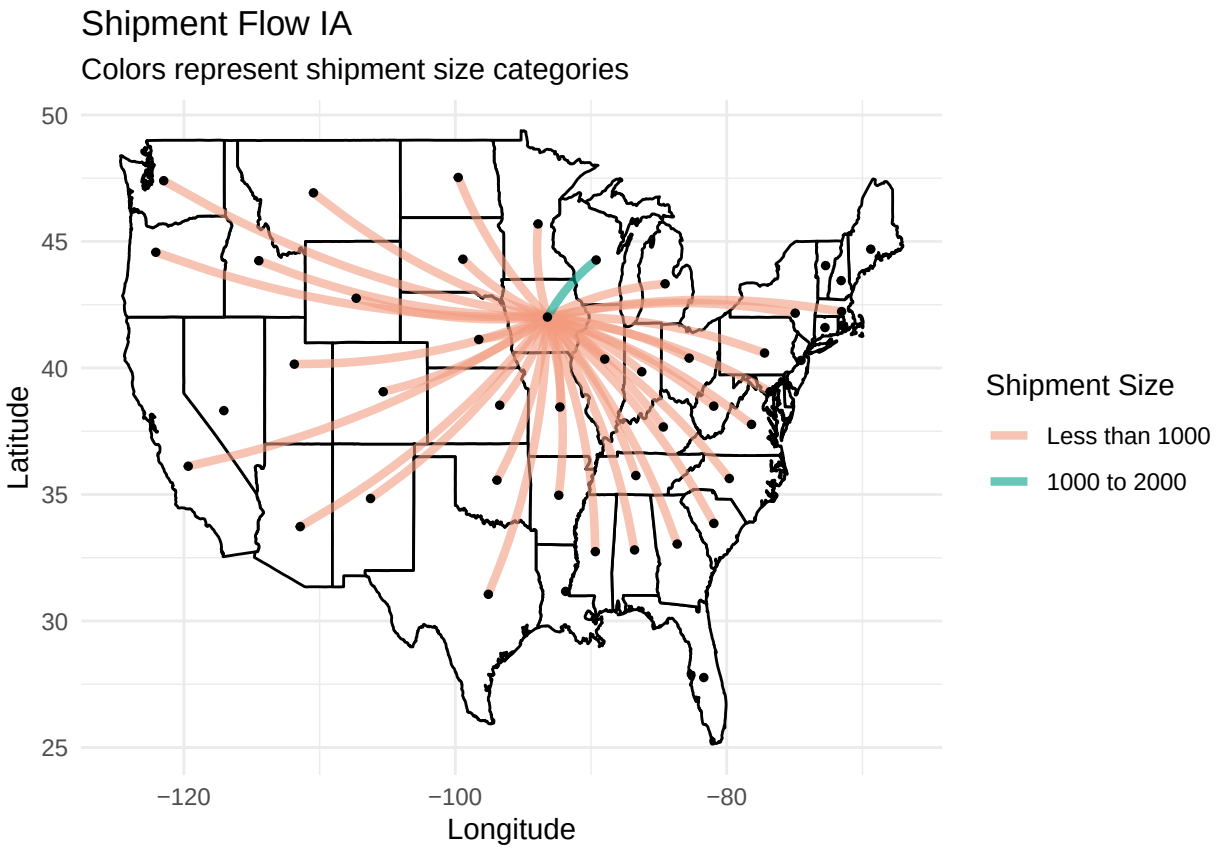
```r
  # Define shipment size categories
  filtered_data <- filtered_data %>%
    mutate(Shipment_Category = cut(`Counts of Shipment`,
                                   breaks = c(-Inf, 1000, 2000, 3000, Inf),
                                   labels = c("Less than 1000", "1000 to 2000", "2000 to 3000", "More t

  # Define colors for each category
  color_palette <- c("Less than 1000" = "#F39B7FFF",
                     "1000 to 2000" = "#00A087FF",
                     "2000 to 3000" = "#3C5488FF",
                     "More than 3000" = "#663399")

  # Create the map plot with color representing shipment size categories
  p <- ggplot() +
    geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "white", color = "black")
    geom_curve(data = filtered_data,
               aes(x = Origin_lon, y = Origin_lat,
                   xend = Destination_lon, yend = Destination_lat,
                   color = Shipment_Category),
               curvature = 0.1,  # Adjust curvature as needed
               size = 1.5,  # Fixed line thickness
               alpha = 0.6) +  # Adjust alpha for transparency
    scale_color_manual(values = color_palette) +
    geom_point(data = state_coords, aes(x = lon, y = lat), color = "black", size = 1) +
    labs(title = paste("Shipment Flow", importer_state),
         subtitle = "Colors represent shipment size categories",
         x = "Longitude",
         y = "Latitude",
         color = "Shipment Size") +
    theme_minimal() +
    theme(legend.position = "right")  # Move legend to the right

  # Print the plot
  print(p)
}

# Plot maps for each major importer state
for (state in importer_states) {
  plot_importer_map(state, movements_data, state_coords_filtered)
}
```


Shipment Flow TX
Colors represent shipment size categories
Latitude
50
45
40
35
30
25
−120
−100
−80
Longitude
Shipment Size
Less than 1000
1000 to 2000
More than 3000


Shipment Flow KS
Colors represent shipment size categories
50
45
40
35
30
25
Latitude
−120
−100
−80
Longitude
Shipment Size
Less than 1000


Shipment Flow NE
Colors represent shipment size categories
50
45
40
35
30
25
Latitude
−120
−100
−80
Longitude
Shipment Size
Less than 1000


Shipment Flow CA
Colors represent shipment size categories
Latitude
Longitude
50
45
40
35
30
25
−120
−100
−80
Shipment Size
Less than 1000
2000 to 3000


Shipment Flow IA
Colors represent shipment size categories
50
45
40
35
30
25
Latitude
−120
−100
−80
Longitude
Shipment Size
Less than 1000
1000 to 2000