Creating Vulnerability Indices Using Principal Component Analysis

Matthew Menon

2025-06-12

Setup

Load in necessary libraries.

```
library(readr)
library(MASS)
library(dplyr)
library(factoextra)
library(exactextractr)
library(tigris)
library(ggplot2)
library(plotly)
library(sf)
library(tibble)
library(GGally)
```

Set working directory and load in `final_df.csv` containing all the necessary data for all Principal Component Analyses. Create DC tracts for spatial maps of vulnerability indexes.

```
# Set working directory
setwd("C:/Users/matth/Desktop/Undergraduate-Research/GWU-Bootcamp/Final Project")

# Read your CSV
data <- read_csv("Data/final_df.csv",
                 col_types = cols(GEOID = col_character()), progress = FALSE)

dc_tracts <- tracts(state = "DC", year = 2020, class = "sf")
```

Flood Suscepbitility Index

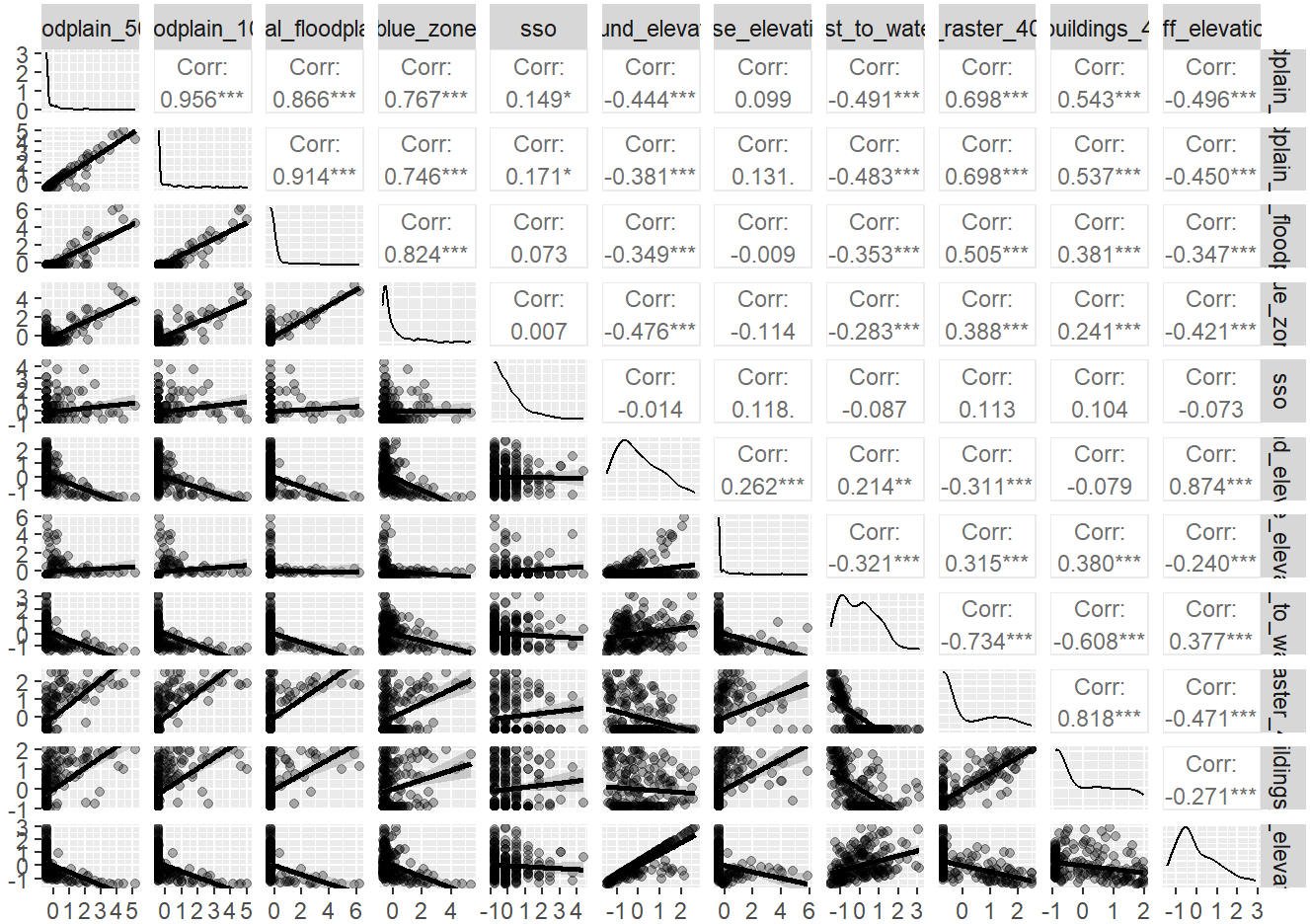
Subset the data from the original dataframe into variables focused on flood susceptibility. Then scale all variables except for `GEOID`, and convert the `flood_scaled` dataframe into a tibble for ease of use.

```
flood_data <- data %>%
  dplyr::select(GEOID, floodplain_500, floodplain_100, tidal_floodplain, blue_zone, sso, ground_
elevation, base_elevation, dist_to_water, pct_raster_407m, pct_buildings_407m) %>%
  mutate(diff_elevation = ground_elevation - base_elevation)

# Scale numeric columns except GEOID
flood_scaled <- flood_data %>%
  mutate(across(
    .cols = where(is.numeric) & !matches("GEOID"),
    .fns = ~ as.numeric(scale(.x))
  ))
flood_scaled = as.tibble(flood_scaled)
```

Construct a correlation matrix of all possible variables to include in the Principal Component Analysis. Based on the results, I choose to remove `ground_elevation`, `base_elevation`, `sso`, `dist_to_water`, and `pct_buildings_407m` in addition to `GEOID` before running the analysis.

```
ggpairs(flood_scaled %>% dplyr::select(-GEOID),
        upper = list(continuous = wrap("cor", size = 3)),
        lower = list(continuous = wrap("smooth", alpha = 0.3)),
        diag = list(continuous = wrap("densityDiag")))
```



```
# Remove non-numeric for PCA input
pca_input <- flood_scaled %>% dplyr::select(-GEOID, -ground_elevation, -base_elevation, -sso, -d
ist_to_water, -pct_buildings_407m)
```

Run a Principal Component Analysis on flood variables of interest to obtain principal components, the proportion of total variance they explain, and the loadings of each principal component for every variable.

```
# Run PCA on scaled data (already scaled, so no centering/scaling here)
pca <- prcomp(pca_input, center = FALSE, scale. = FALSE)

# Summary to see variance explained
pca_summary <- summary(pca)

# Print variance explained by each PC (proportion of variance)
print(pca_summary$importance)
```

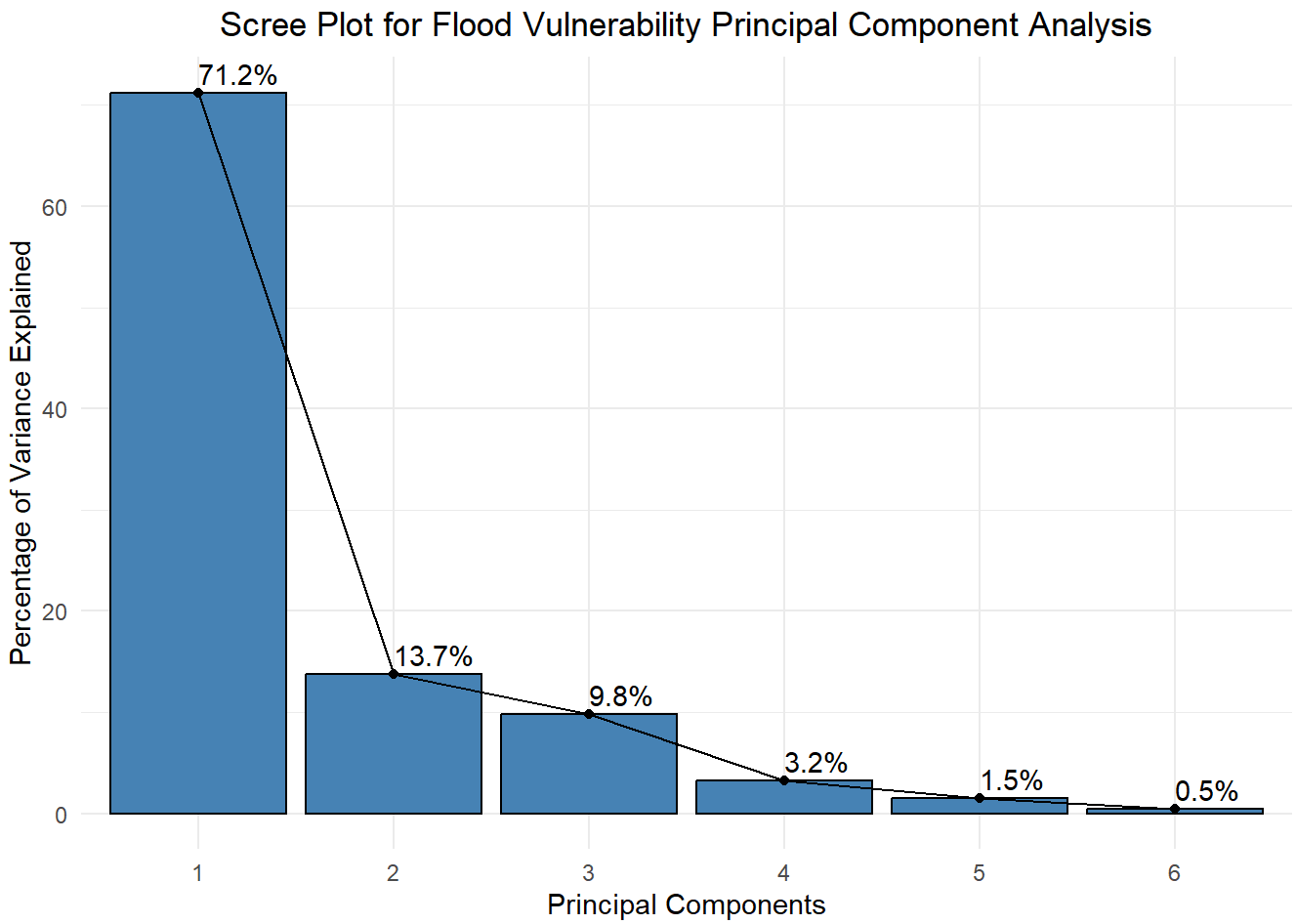
```
##                             PC1       PC2      PC3       PC4       PC5      PC6
## Standard deviation     2.067404 0.9079327 0.765893 0.4410938 0.3030907 0.168755
## Proportion of Variance 0.712360 0.1373900 0.097770 0.0324300 0.0153100 0.004750
## Cumulative Proportion  0.712360 0.8497500 0.947520 0.9799400 0.9952500 1.000000
```

```
print(pca$rotation)
```

```
##                         PC1         PC2         PC3        PC4         PC5
## floodplain_500    0.4663183 -0.05735749  0.11070710  0.2254074  0.69241130
## floodplain_100    0.4661006 -0.11608943  0.17109168  0.3615188  0.07727808
## tidal_floodplain  0.4402992 -0.36299453 -0.05176918  0.2841404 -0.67251483
## blue_zone         0.4049806 -0.31925993 -0.44008604 -0.7170344  0.09620686
## pct_raster_407m   0.3555689  0.44713906  0.65647833 -0.4411045 -0.20946550
## diff_elevation   -0.2845972 -0.74135557  0.57545808 -0.1704357  0.09584856
##                          PC6
## floodplain_500    0.486579241
## floodplain_100   -0.776745352
## tidal_floodplain  0.372397100
## blue_zone        -0.130361110
## pct_raster_407m   0.064996956
## diff_elevation   -0.003012582
```

Create a scree plot to show how much of the variance each principal component from the PCA explains in the data. I choose to only include on principal component in the flood susceptibility index since the “elbow” of the scree plot occurs after principal component 1.

```
# Generate a scree plot
fviz_eig(pca, addlabels = TRUE, barfill = "steelblue", barcolor = "black") +
  labs(title = "Scree Plot for Flood Vulnerability Principal Component Analysis", x = "Principal
Components", y = "Percentage of Variance Explained") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

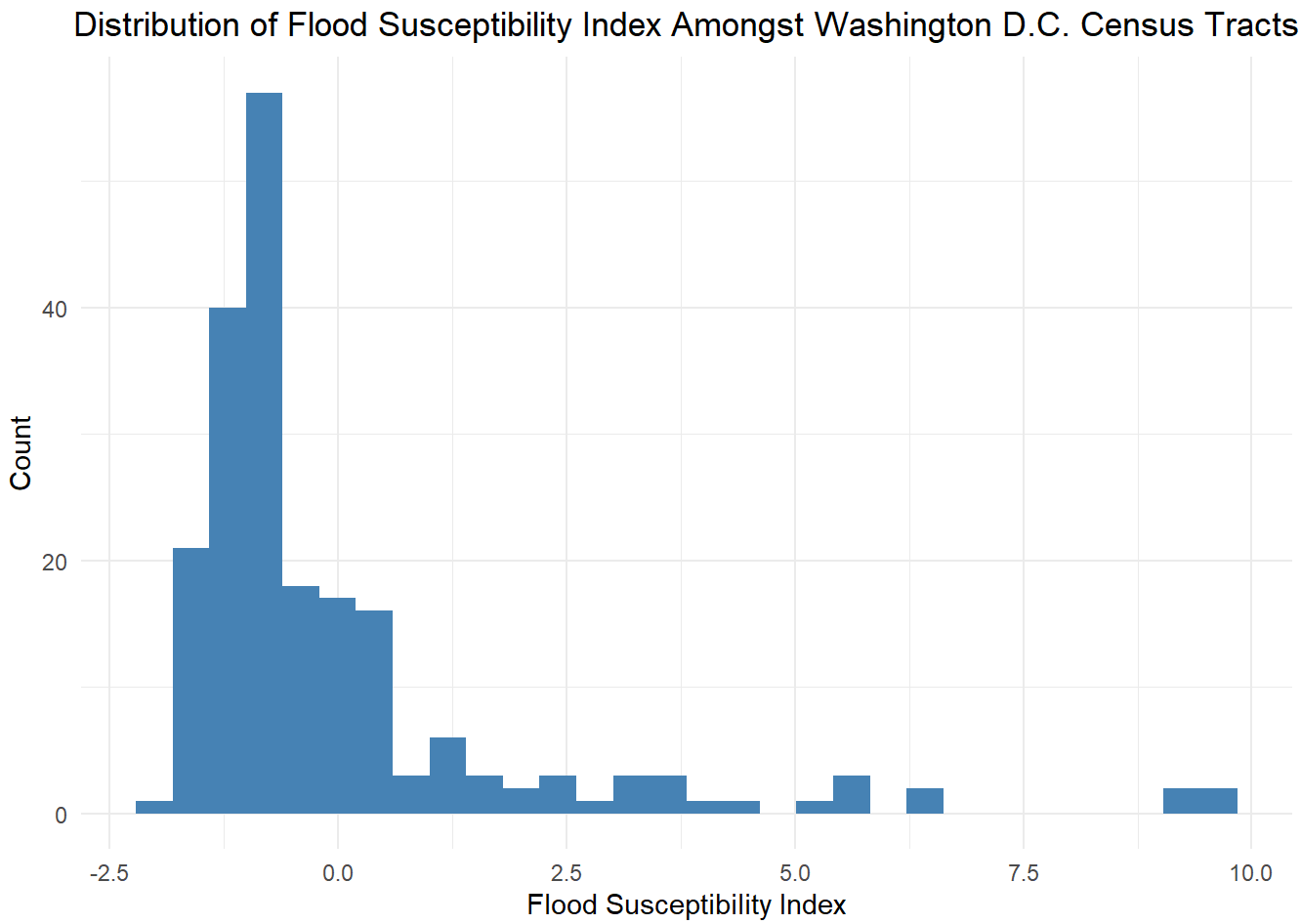


Add the flood susceptibility index back to the original datasets.

```
# Add flood susceptibility index to original data
flood_data$flood_index <- pca$x[, 1]
data$flood_index <- pca$x[, 1]
```

Create a bar chart to show the distribution of the flood susceptibility index.

```
# Plot flood susceptibility index histogram
ggplot(flood_data, aes(x = flood_index)) +
  geom_histogram(fill = "steelblue") +
  labs(title = "Distribution of Flood Susceptibility Index Amongst Washington D.C. Census Tract
s",
       x = "Flood Susceptibility Index", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

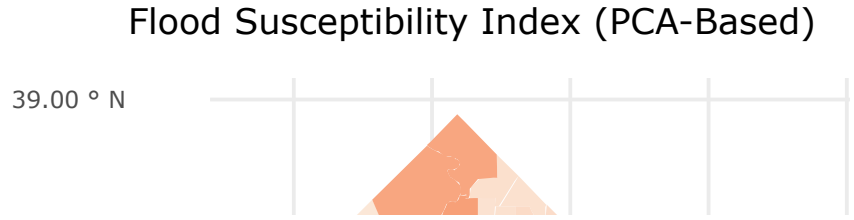


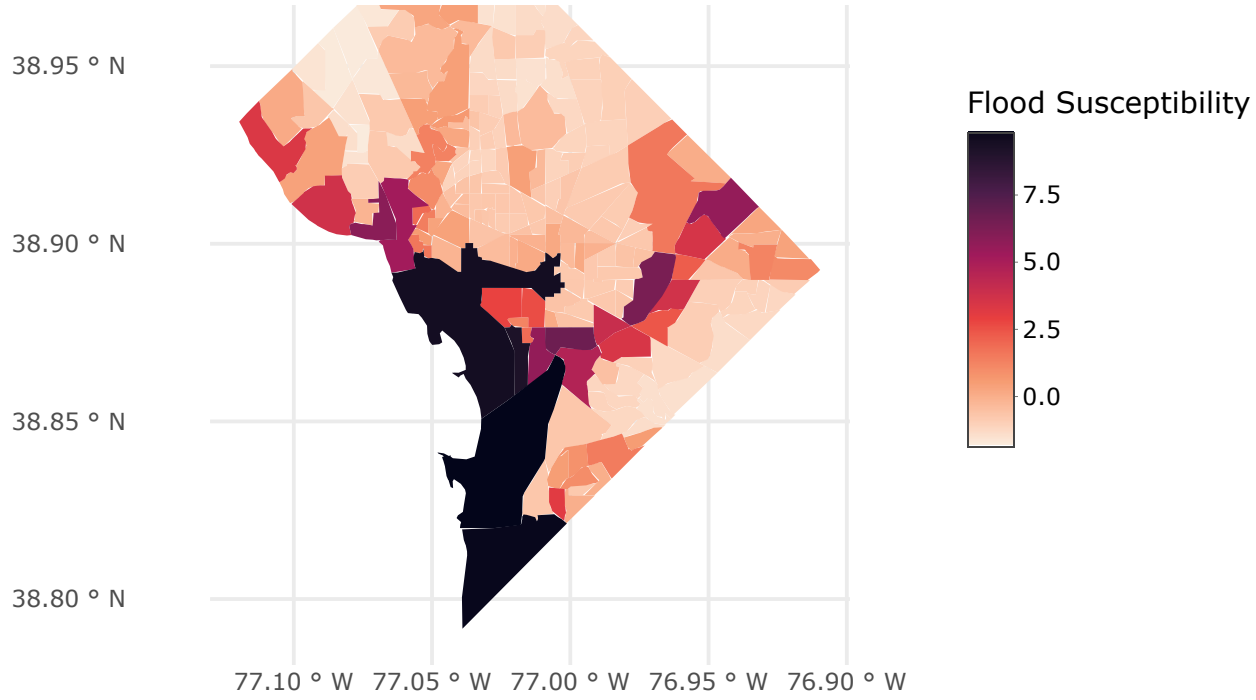
Generate a spatial representation of the flood susceptibility index described by Principal Component 1 of the above PCA to visualize most flood susceptible census tracts in Washington D.C.

```
# Join with spatial data for mapping
dc_map <- dc_tracts %>%
  left_join(flood_data, by = "GEOID")

# Spatial plot with ggplot + plotly
pca_plot <- ggplot(dc_map) +
  geom_sf(aes(fill = flood_index,
              text = paste0("Flood Susceptibility: ", round(flood_index, 2))),
          color = NA) +
  scale_fill_viridis_c(option = "rocket", name = "Flood Susceptibility", direction = -1) +
  labs(title = "Flood Susceptibility Index (PCA-Based)",
       subtitle = "Census Tracts (2020)",
       caption = "Source: Open Data DC") +
  theme_minimal()

ggplotly(pca_plot, tooltip = "text")
```





Social and Health Vulnerability Index

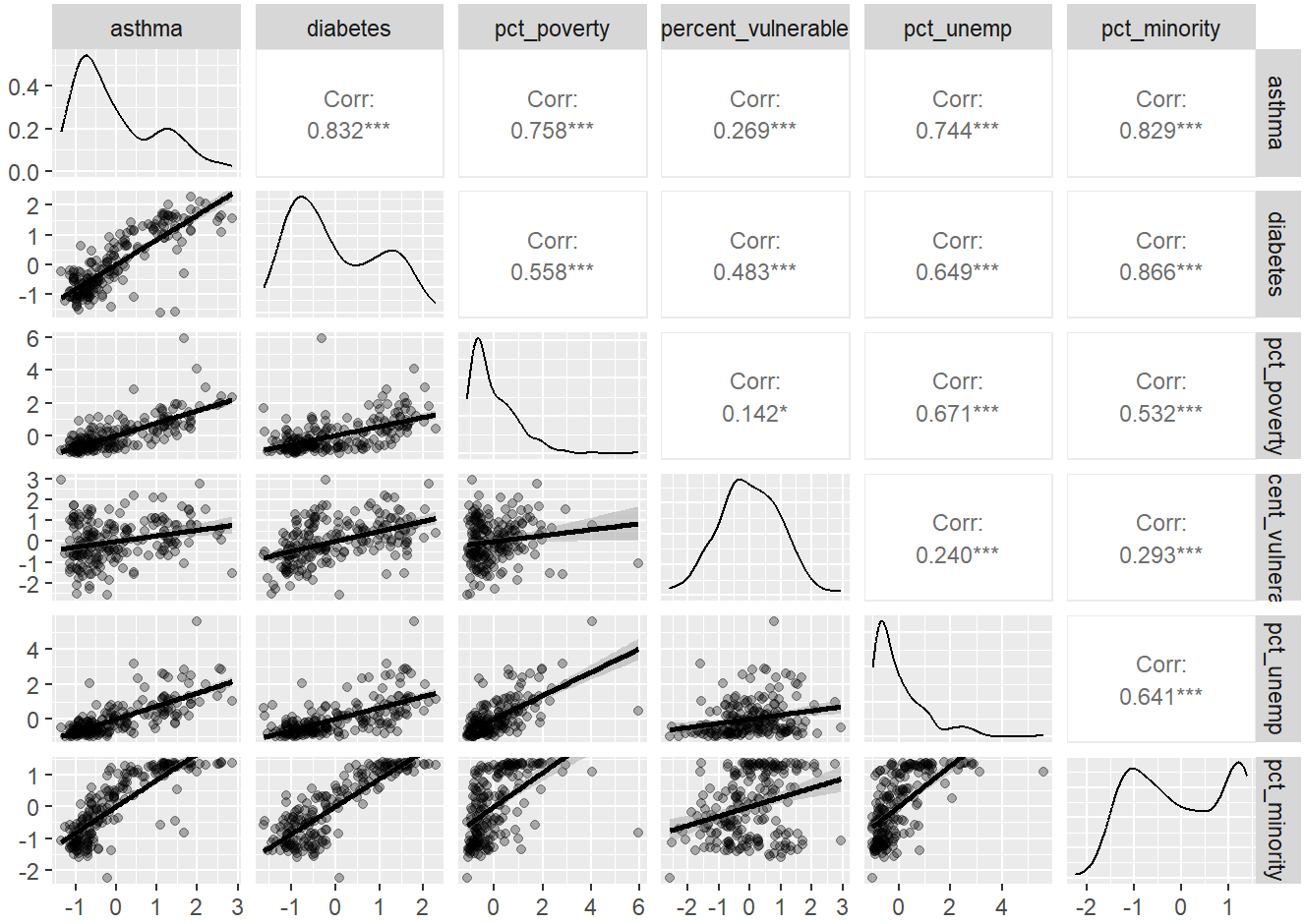
Subset the data from the original dataframe into variables focused on social and health vulnerability. Then scale all variables except for `GEOID`, and convert the `community_scaled` dataframe into a tibble for ease of use.

```
community_data <- data %>%
  dplyr::select(GEOID, asthma, diabetes, pct_poverty, percent_vulnerable, pct_unemp, pct_minorit
y)

# Scale numeric columns except GEOID
community_scaled <- community_data %>%
  mutate(across(
    .cols = where(is.numeric) & !matches("GEOID"),
    .fns = ~ as.numeric(scale(.x))
  ))
community_scaled = as.tibble(community_scaled)
```

Construct a correlation matrix of all possible variables to include in the Principal Component Analysis. Based on the results, I choose to only remove `GEOID` before running the analysis.

```
ggpairs(community_scaled %>% dplyr::select(-GEOID),
        upper = list(continuous = wrap("cor", size = 3)),
        lower = list(continuous = wrap("smooth", alpha = 0.3)),
        diag = list(continuous = wrap("densityDiag")))
```




```
# Remove non-numeric for PCA input
pca_input <- community_scaled %>% dplyr::select(-GEOID)
```

Run a Principal Component Analysis on social and health variables of interest to obtain principal components, the proportion of total variance they explain, and the loadings of each principal component for every variable.

```
# Run PCA on scaled data (already scaled, so no centering/scaling here)
pca <- prcomp(pca_input, center = FALSE, scale. = FALSE)

# Summary to see variance explained
pca_summary <- summary(pca)

# Print variance explained by each PC (proportion of variance)
print(pca_summary$importance)
```

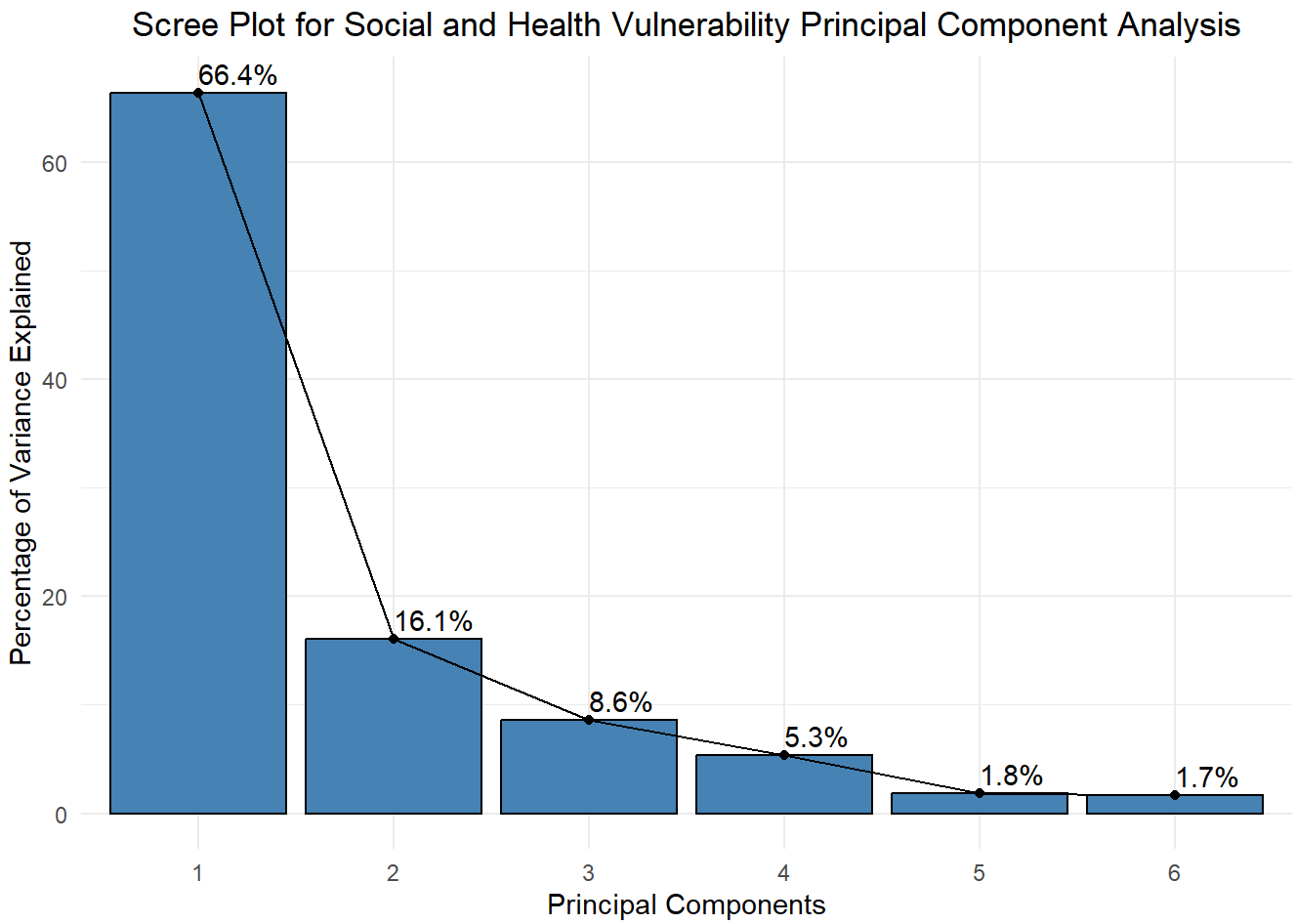
```
##                             PC1       PC2       PC3       PC4       PC5
## Standard deviation     1.996373 0.9823282 0.719483 0.5659642 0.3314568
## Proportion of Variance 0.664250 0.1608300 0.086280 0.0533900 0.0183100
## Cumulative Proportion  0.664250 0.8250800 0.911350 0.9647400 0.9830500
##                            PC6
## Standard deviation     0.3188909
## Proportion of Variance 0.0169500
## Cumulative Proportion  1.0000000
```

```
print(pca$rotation)
```

```
##                         PC1         PC2        PC3         PC4         PC5
## asthma             0.4725288  0.14341047  0.0834720 -0.18959688 -0.56462290
## diabetes           0.4580745 -0.20696189  0.3227754 -0.09343559 -0.37295702
## pct_poverty        0.3883829  0.36509356 -0.5794031 -0.51112759  0.28022643
## percent_vulnerable 0.2123532 -0.87500526 -0.3854889 -0.03834860  0.10086491
## pct_unemp          0.4171360  0.19292910 -0.3063238  0.83203004  0.02087418
## pct_minority       0.4437867 -0.02123807  0.5574088 -0.01807747  0.67302878
##                           PC6
## asthma             -0.62804169
## diabetes            0.70379215
## pct_poverty         0.20094584
## percent_vulnerable -0.17036971
## pct_unemp           0.04724374
## pct_minority       -0.19647700
```

Create a scree plot to show how much of the variance each principal component from the PCA explains in the data. I choose to only include principal component 1 in the social and health vulnerability index since the “elbow” of the scree plot occurs after principal component 1.

```
# Generate a scree plot
fviz_eig(pca, addlabels = TRUE, barfill = "steelblue", barcolor = "black") +
  labs(title = "Scree Plot for Social and Health Vulnerability Principal Component Analysis", x
= "Principal Components", y = "Percentage of Variance Explained") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

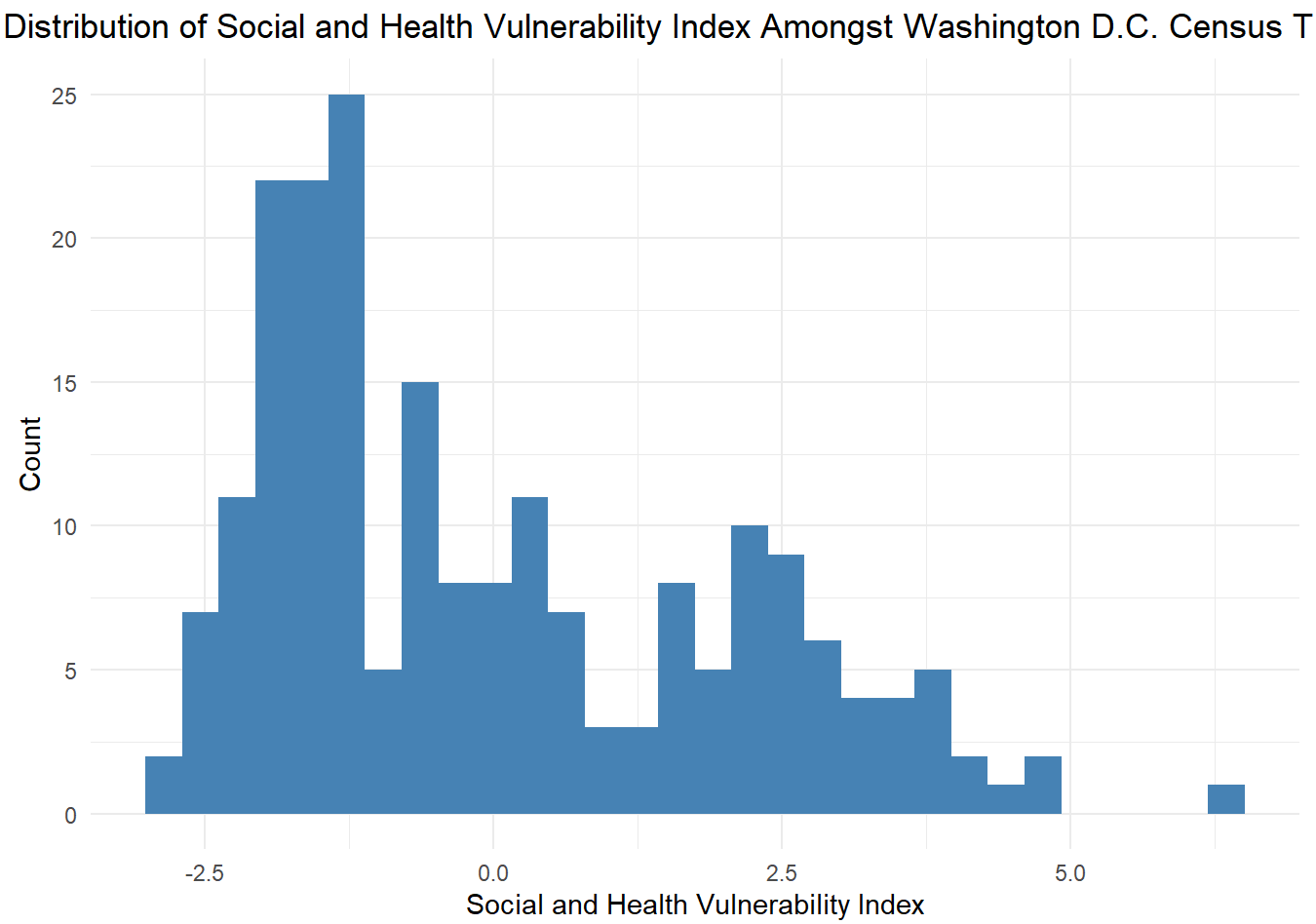


Add the social and health vulnerability index back to the original datasets.

```
# Add social + health vulnerability index to original data
community_data$social_health_index <- pca$x[,1]
data$social_health_index <- pca$x[,1]
```

Create a bar chart to show the distribution of the social and health vulnerability index.

```
# Plot social and health vulnerability histogram
ggplot(community_data, aes(x = social_health_index)) +
  geom_histogram(fill = "steelblue") +
  labs(title = "Distribution of Social and Health Vulnerability Index Amongst Washington D.C. Ce
nsus Tracts",
       x = "Social and Health Vulnerability Index", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

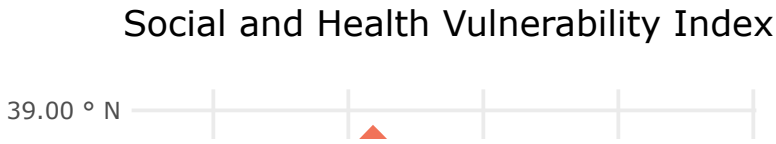


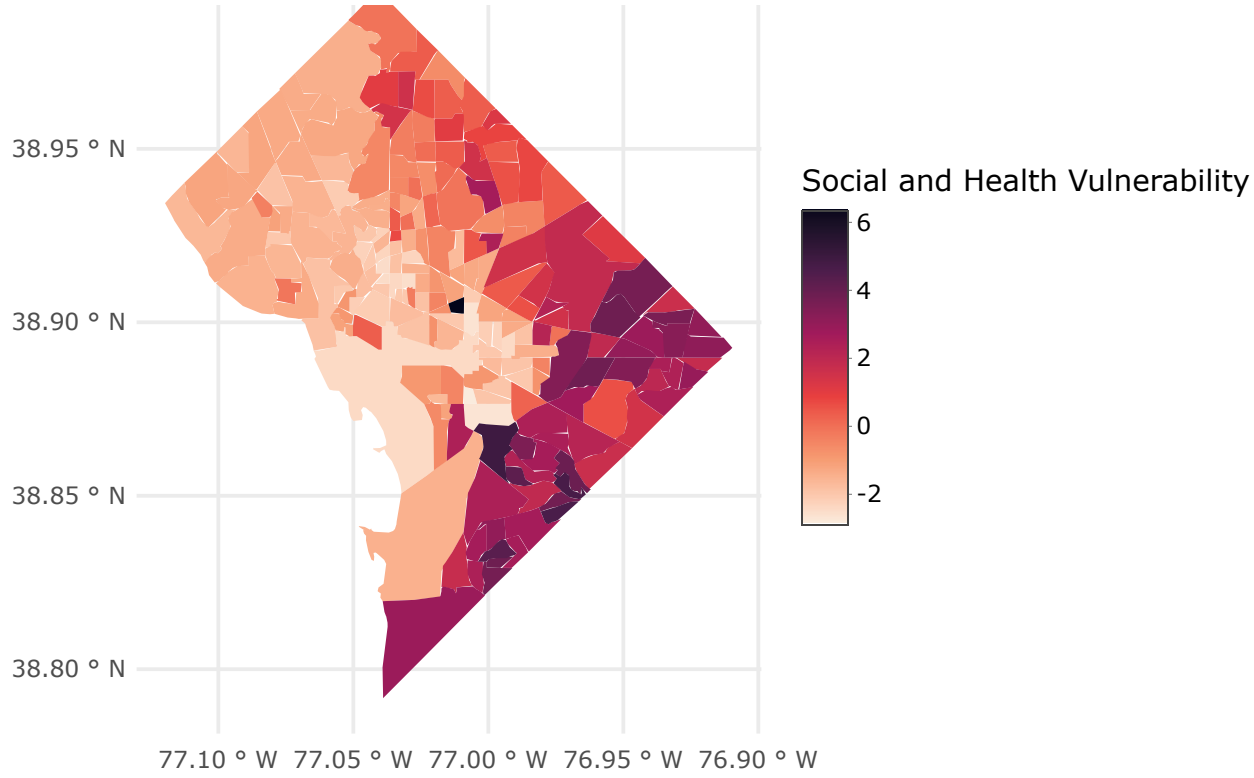
Generate a spatial representation of the social and health vulnerability index described by Principal Components 1 and 2 of the above PCA to visualize most vulnerable areas when considering health and social demographic factors by census tracts in Washington D.C.

```
# Join with spatial data for mapping
dc_map <- dc_tracts %>%
  left_join(community_data, by = "GEOID")

# Spatial plot with ggplot + plotly
pca_plot <- ggplot(dc_map) +
  geom_sf(aes(fill = social_health_index,
              text = paste0("Flood Vulnerability: ", round(social_health_index, 2))),
          color = NA) +
  scale_fill_viridis_c(option = "rocket", name = "Social and Health Vulnerability", direction =
-1) +
  labs(title = "Social and Health Vulnerability Index (PCA-Based)",
       subtitle = "Census Tracts (2020)",
       caption = "Source: Open Data DC") +
  theme_minimal()

ggplotly(pca_plot, tooltip = "text")
```





Infrastructure Index

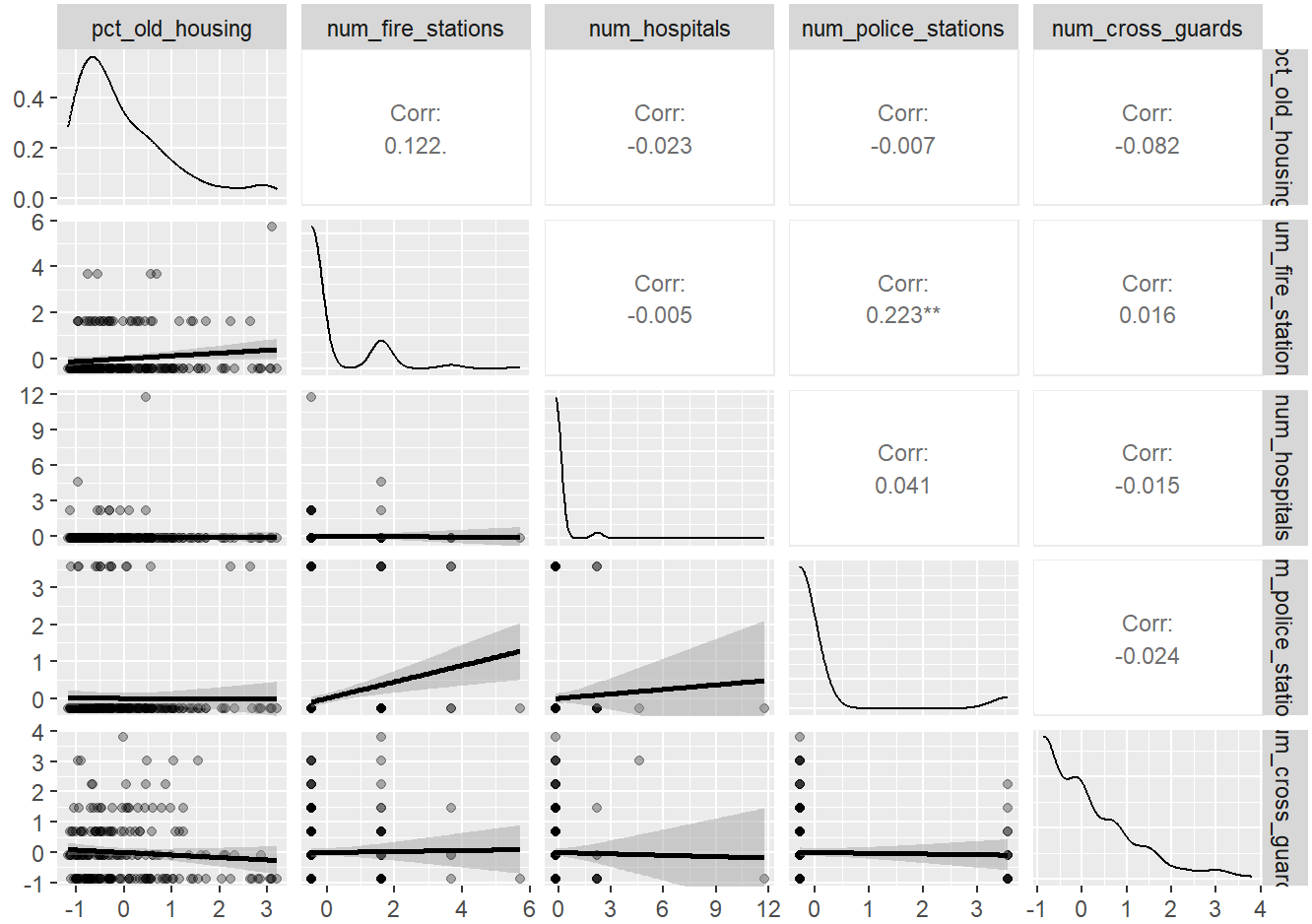
Subset the data from the original dataframe into variables focused on infrastructure. Then scale all variables except for `GEOID` , and convert the `infrastructure_scaled` dataframe into a tibble for ease of use.

```
infrastructure_data <- data %>%
  dplyr::select(GEOID, pct_old_housing, num_fire_stations, num_hospitals, num_police_stations, n
um_cross_guards)

# Scale numeric columns except GEOID
infrastructure_scaled <- infrastructure_data %>%
  mutate(across(
    .cols = where(is.numeric) & !matches("GEOID"),
    .fns = ~ as.numeric(scale(.x))
  ))
infrastructure_scaled = as.tibble(infrastructure_scaled)
```

Construct a correlation matrix of all possible variables to include in the Principal Component Analysis. Based on the results, I choose to only `GEOID` before running the analysis, as they all show very low correlation and there is no clear variable that to be removed over the others.

```
ggpairs(infrastructure_scaled %>% dplyr::select(-GEOID),
        upper = list(continuous = wrap("cor", size = 3)),
        lower = list(continuous = wrap("smooth", alpha = 0.3)),
        diag = list(continuous = wrap("densityDiag")))
```




```
# Remove non-numeric for PCA input
pca_input <- infrastructure_scaled %>% dplyr::select(-GEOID)
```

Run a Principal Component Analysis on infrastructure variables of interest to obtain principal components, the proportion of total variance they explain, and the loadings of each principal component for every variable.

```
# Run PCA on scaled data (already scaled, so no centering/scaling here)
pca <- prcomp(pca_input, center = FALSE, scale. = FALSE)

# Summary to see variance explained
pca_summary <- summary(pca)

# Print variance explained by each PC (proportion of variance)
print(pca_summary$importance)
```

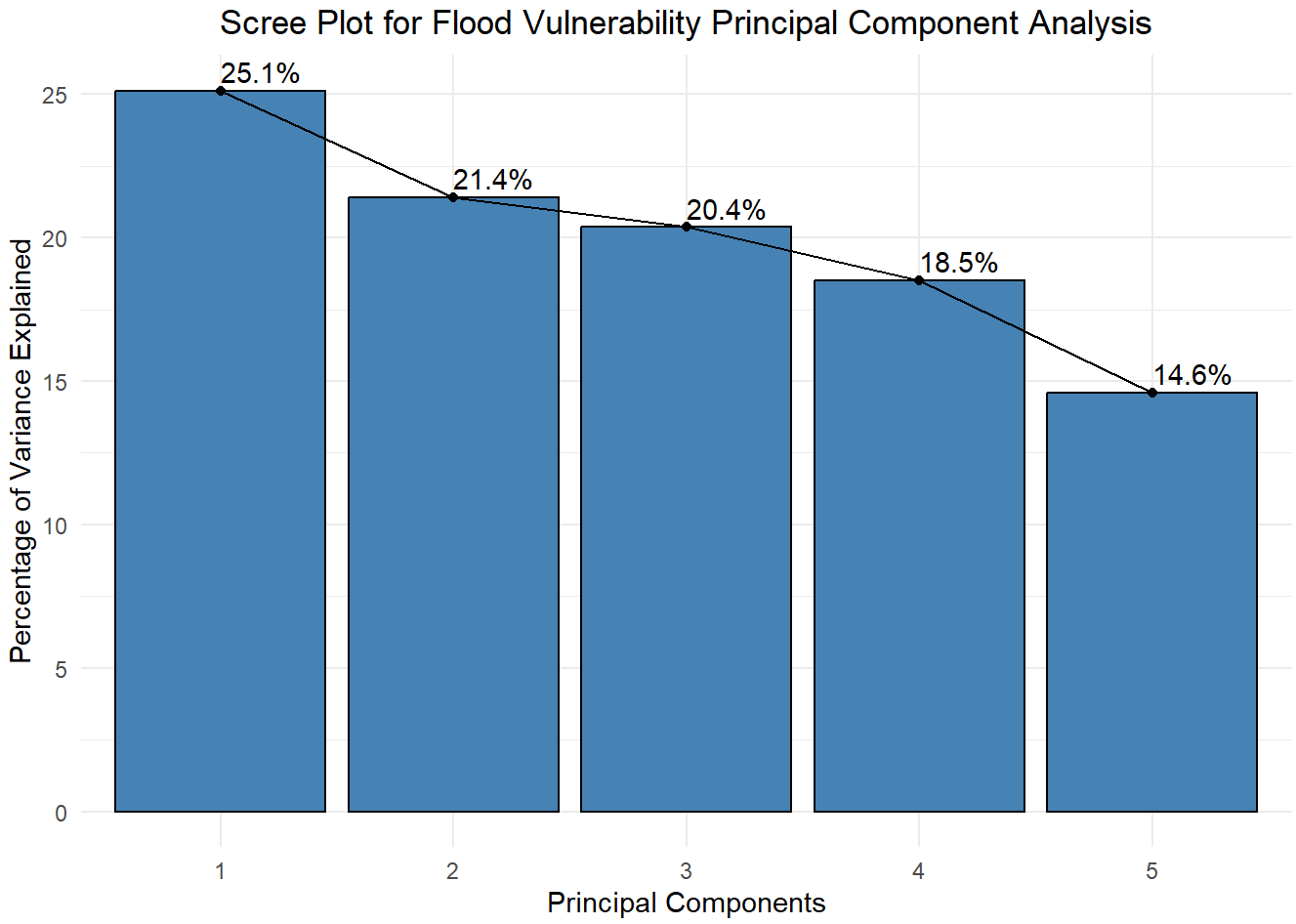
```
##                             PC1      PC2      PC3       PC4       PC5
## Standard deviation     1.120908 1.034441 1.009128 0.9620062 0.8542269
## Proportion of Variance 0.251290 0.214010 0.203670 0.1850900 0.1459400
## Cumulative Proportion  0.251290 0.465300 0.668970 0.8540600 1.0000000
```

```
print(pca$rotation)
```

```
##                             PC1        PC2         PC3        PC4         PC5
## pct_old_housing       0.35059514  0.6493901 -0.10714030 -0.5476268  0.37947517
## num_fire_stations     0.69202360 -0.1068238 -0.20121776 -0.1124812 -0.67568587
## num_hospitals         0.06096542 -0.2748697  0.82389765 -0.4884062 -0.05815452
## num_police_stations   0.61429613 -0.3520674  0.09754357  0.3685912  0.59440275
## num_cross_guards     -0.13080471 -0.6061158 -0.50961777 -0.5595112  0.20676223
```

Create a scree plot to show how much of the variance each principal component from the PCA explains in the data. I notice the linearity of the scree plot, and the little variance being explained by the first principal component. Since the first principal components don't explain significantly more of the variance compared to the later ones, I decide to take a different approach in creating an infrastructure vulnerability index.

```
# Generate a scree plot
fviz_eig(pca, addlabels = TRUE, barfill = "steelblue", barcolor = "black") +
  labs(title = "Scree Plot for Flood Vulnerability Principal Component Analysis", x = "Principal
Components", y = "Percentage of Variance Explained") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



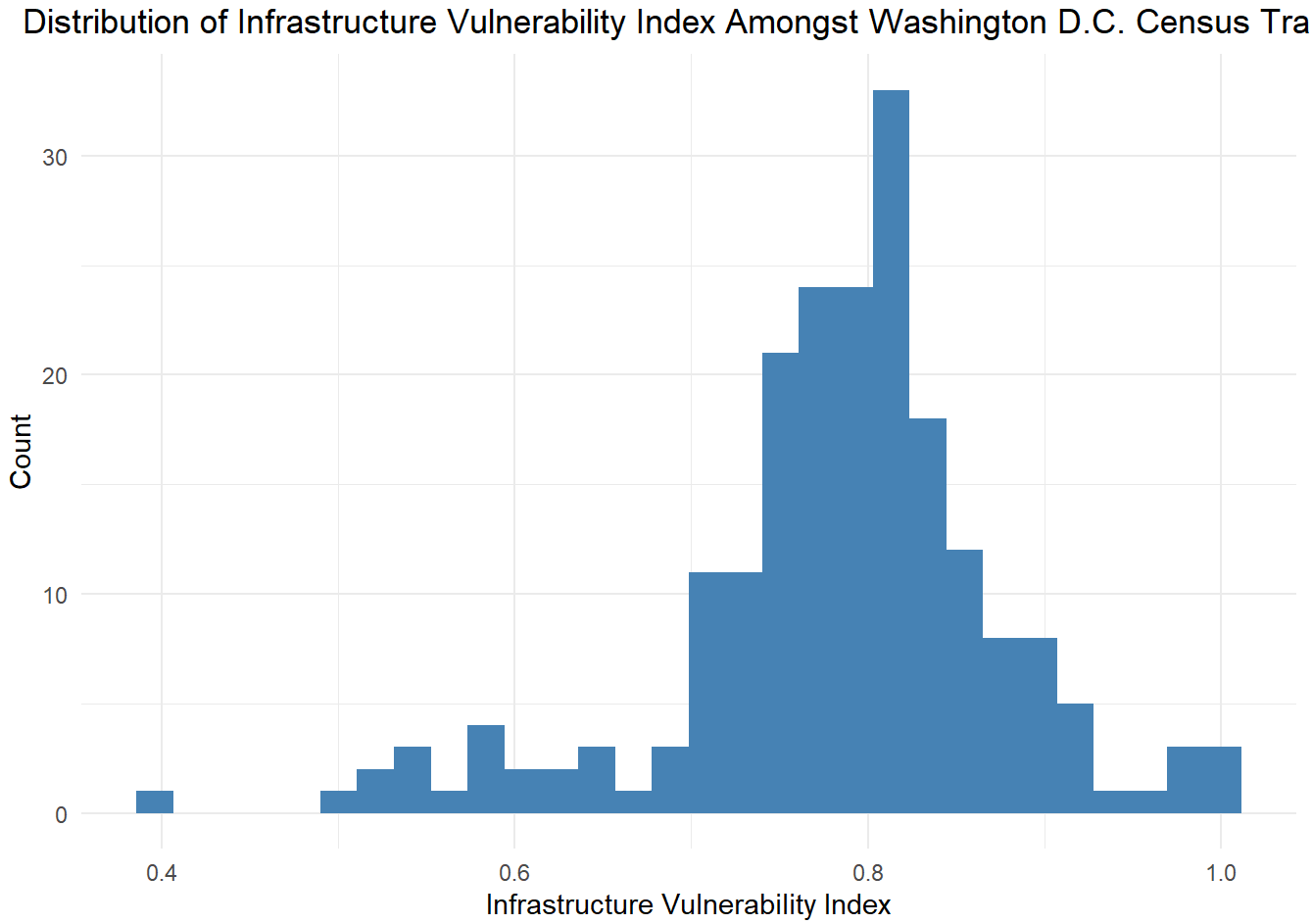
Instead of utilizing Principal Component Analysis, in this case I simply utilize equal weights of 0.2 for each of the five infrastructure variables after normalizing them to create the infrastructure index.

```
infrastructure_data <- infrastructure_data %>%
  mutate(across(c(num_fire_stations, num_hospitals, num_police_stations, num_cross_guards, pct_o
ld_housing),
                ~ (. - min(.)) / (max(.) - min(.)))) %>%
  mutate(infra_index = 0.2 * (1 - num_fire_stations)
                     + 0.2 * (1 - num_hospitals)
                     + 0.2 * (1 - num_police_stations)
                     + 0.2 * (1 - num_cross_guards)
                     + 0.2 * pct_old_housing)

data$infra_index = infrastructure_data$infra_index
```

Create a bar chart to show the distribution of the infrastructure vulnerability index.

```
# Plot infrastructure vulnerability histogram
ggplot(infrastructure_data, aes(x = infra_index)) +
  geom_histogram(fill = "steelblue") +
  labs(title = "Distribution of Infrastructure Vulnerability Index Amongst Washington D.C. Censu
s Tracts",
       x = "Infrastructure Vulnerability Index", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

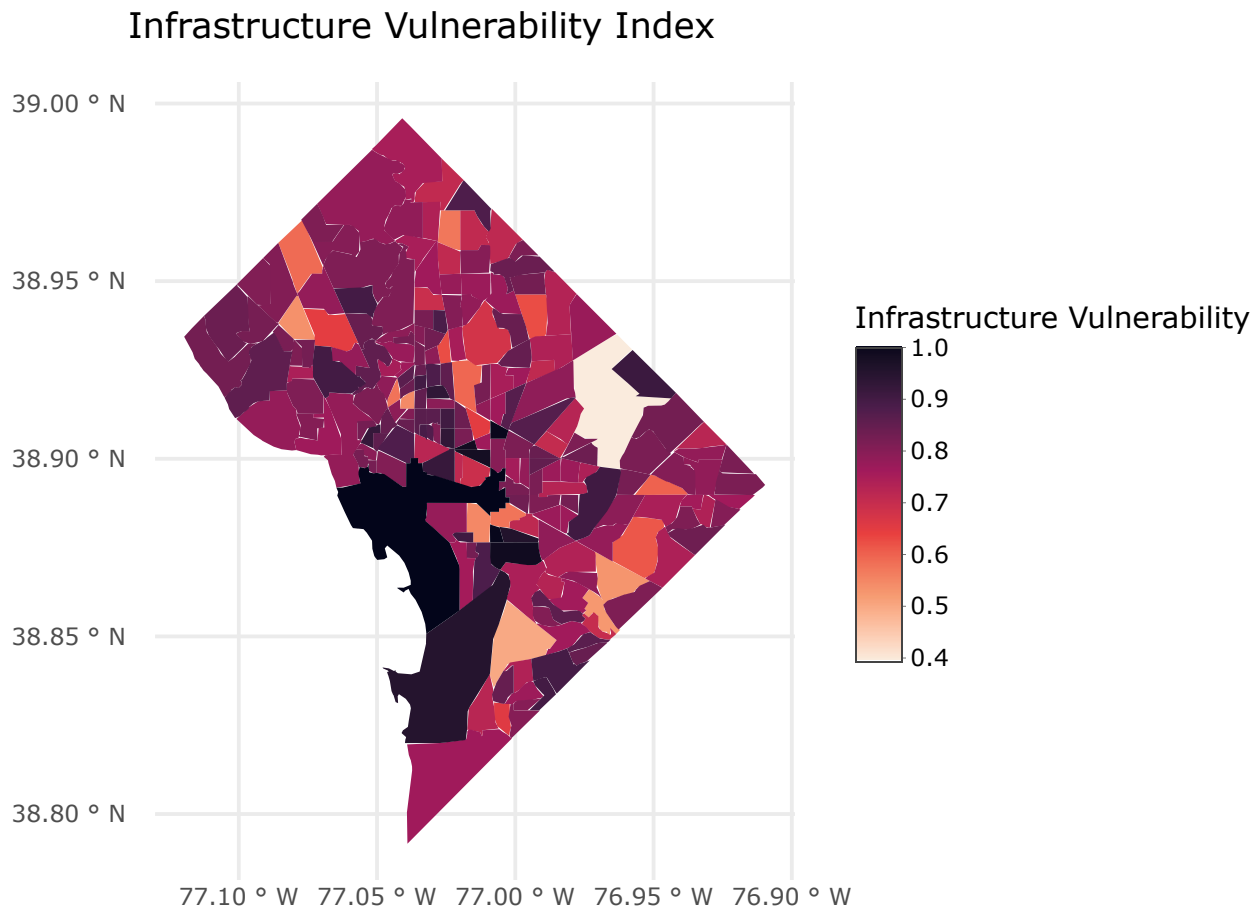


Generate a spatial representation of the infrastructure vulnerability index described by equal weights to all variables to visualize most vulnerable areas when considering presence of key infrastructure by census tracts in Washington D.C.

```
# Join with spatial data for mapping
dc_map <- dc_tracts %>%
  left_join(infrastructure_data, by = "GEOID")

# Spatial plot with ggplot + plotly
pca_plot <- ggplot(dc_map) +
  geom_sf(aes(fill = infra_index,
              text = paste0("Infrastructure Vulnerability: ", round(infra_index, 2))),
          color = NA) +
  scale_fill_viridis_c(option = "rocket", name = "Infrastructure Vulnerability", direction = -1)
+
  labs(title = "Infrastructure Vulnerability Index",
       subtitle = "Census Tracts (2020)",
       caption = "Source: Open Data DC") +
  theme_minimal()

ggplotly(pca_plot, tooltip = "text")
```



Composite Vulnerability Index

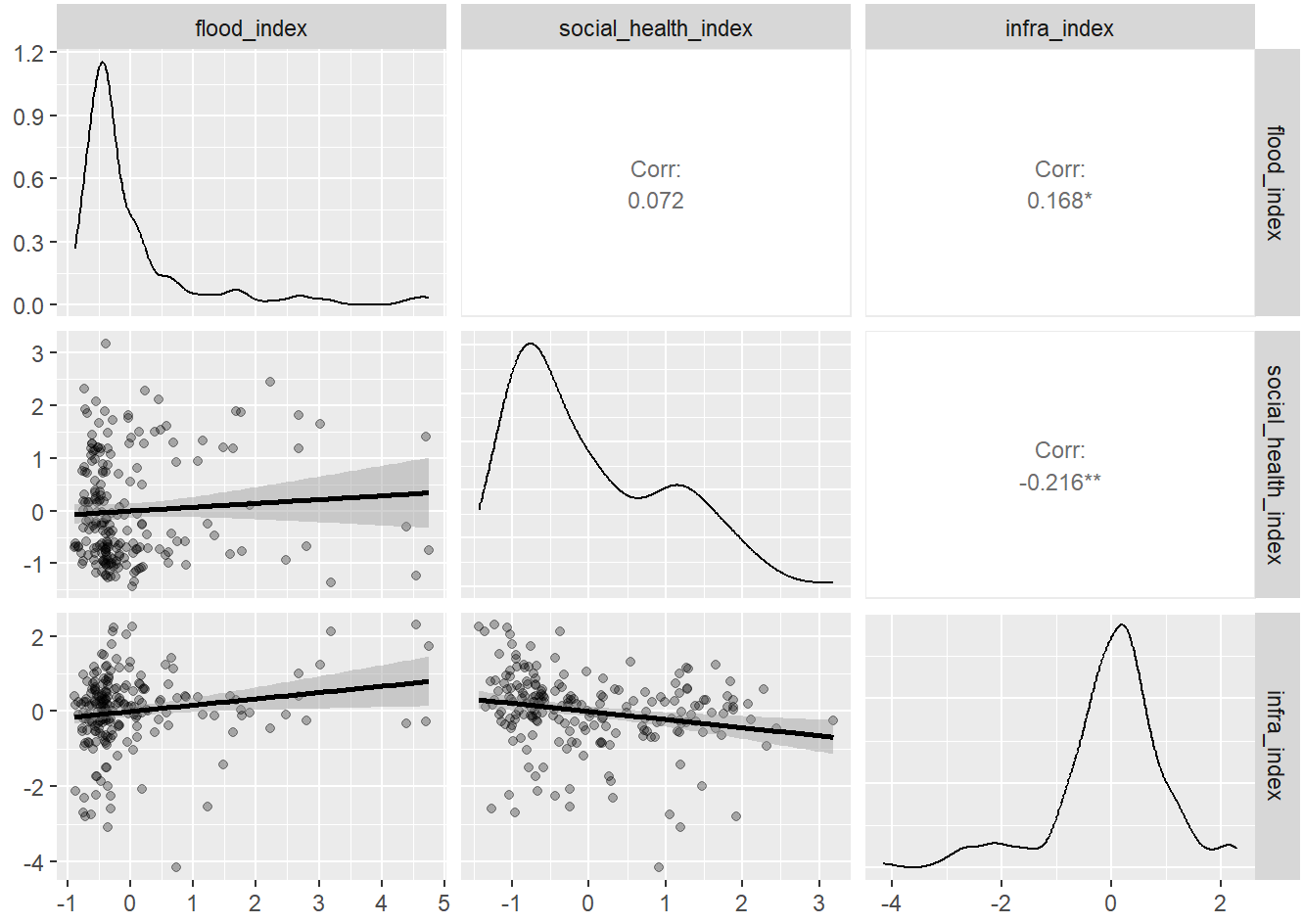
Subset the data from the original dataframe into the three indices derived from the above analyses. Then scale all variables except for `GEOID`, and convert the `community_scaled` dataframe into a tibble for ease of use.

```
indices_data <- data %>%
  dplyr::select(GEOID, flood_index, social_health_index, infra_index)

# Scale numeric columns except GEOID
indices_scaled <- indices_data %>%
  mutate(across(
    .cols = where(is.numeric) & !matches("GEOID"),
    .fns = ~ as.numeric(scale(.x))
  ))
indices_scaled = as.tibble(indices_scaled)
```

Construct a correlation matrix of all possible variables to include in the Principal Component Analysis. Based on the results, since all the indices are extremely uncorrelated with one another, I choose to only remove `GEOID` before running the analysis.

```
ggpairs(indices_scaled %>% dplyr::select(-GEOID),
        upper = list(continuous = wrap("cor", size = 3)),
        lower = list(continuous = wrap("smooth", alpha = 0.3)),
        diag = list(continuous = wrap("densityDiag")))
```



```
# Remove non-numeric for PCA input
pca_input <- indices_scaled %>% dplyr::select(-GEOID)
```

Run a Principal Component Analysis on all the indices found above to obtain principal components, the proportion of total variance they explain, and the loadings of each principal component for every variable.

```
# Run PCA on scaled data (already scaled, so no centering/scaling here)
pca <- prcomp(pca_input, center = FALSE, scale. = FALSE)

# Summary to see variance explained
pca_summary <- summary(pca)

# Print variance explained by each PC (proportion of variance)
print(pca_summary$importance)
```

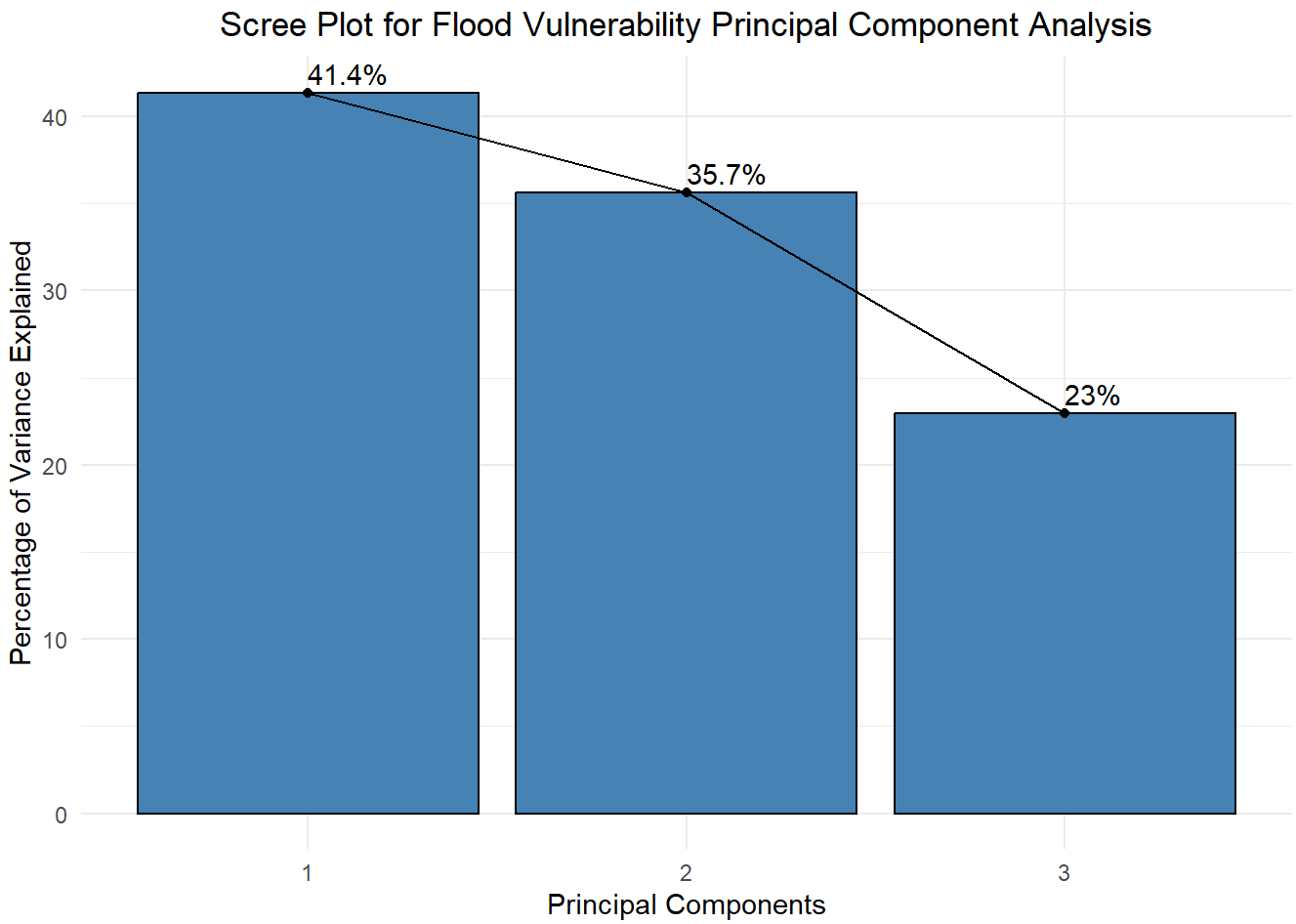
```
##                             PC1      PC2       PC3
## Standard deviation     1.114107 1.034187 0.8301942
## Proportion of Variance 0.413740 0.356510 0.2297400
## Cumulative Proportion  0.413740 0.770260 1.0000000
```

```
print(pca$rotation)
```

```
##                            PC1        PC2        PC3
## flood_index         -0.3509729 0.79897131 -0.4883266
## social_health_index  0.5636296 0.59670403  0.5711970
## infra_index         -0.7477565 0.07476059  0.6597508
```

Create a scree plot to show how much of the variance each principal component from the PCA explains in the data. As was the case with the infrastructure index, each principal component explains a similar amount of variation compared to the previous, meaning there is no clear number of principal components to use. It is smarter to utilize weights independent of this analysis in this case.

```
# Generate a scree plot
fviz_eig(pca, addlabels = TRUE, barfill = "steelblue", barcolor = "black") +
  labs(title = "Scree Plot for Flood Vulnerability Principal Component Analysis", x = "Principal
Components", y = "Percentage of Variance Explained") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



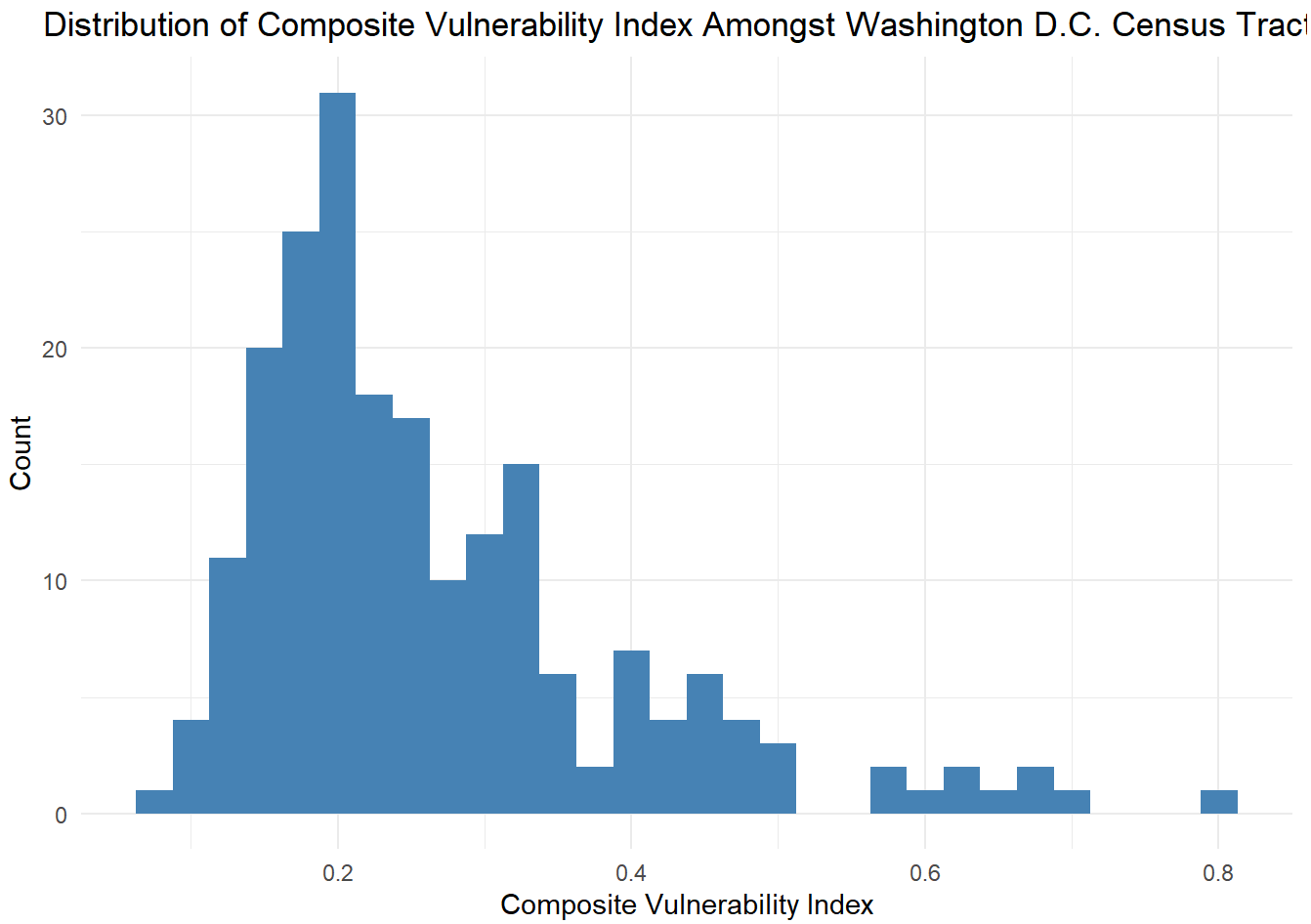
Instead of utilizing Principal Component Analysis, I normalizing each of the vulnerability indices before assigning weights to them based on their importance to the study. The primary focus is to understand how flood susceptibility and social plus health vulnerability are related, while infrastructure is of less concern but still important nonetheless. This is why I chose to apply equal weights of 0.4 to the flood index and to the social and health vulnerability index, and a smaller weight of 0.2 to the infrastructure index.

```
indices_data <- indices_data %>%
  mutate(across(c(flood_index, social_health_index, infra_index),
                ~ (. - min(.)) / (max(.) - min(.)))) %>%
  mutate(composite_index = 0.5 * flood_index
                    + 0.4 * social_health_index
                    + 0.1 * infra_index)

data$composite_index = indices_data$composite_index
```

Create a bar chart to show the distribution of the composite vulnerability index.

```
# Plot composite vulnerability histogram
ggplot(indices_data, aes(x = composite_index)) +
  geom_histogram(fill = "steelblue") +
  labs(title = "Distribution of Composite Vulnerability Index Amongst Washington D.C. Census Tra
cts",
       x = "Composite Vulnerability Index", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

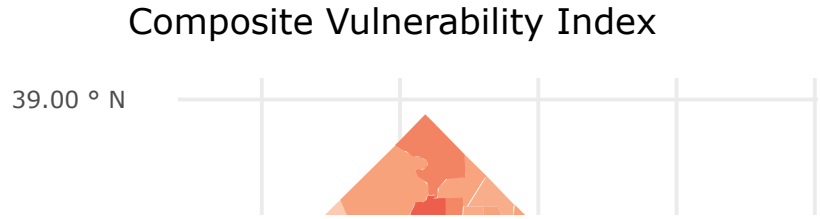


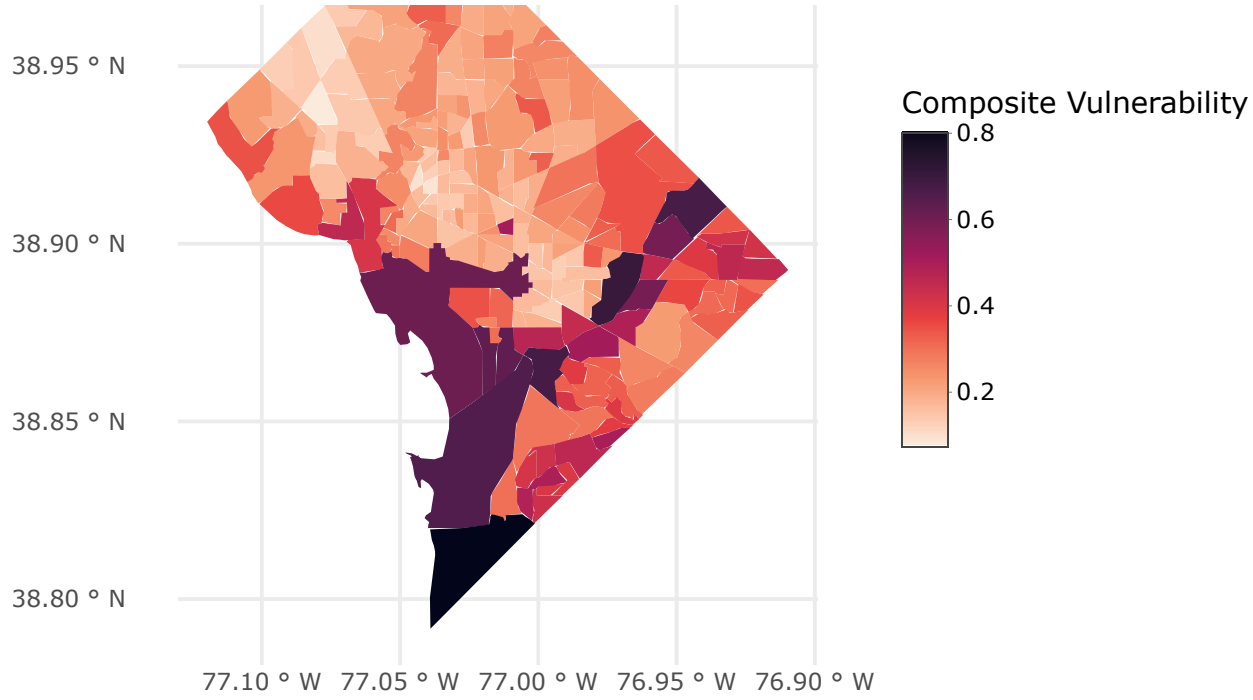
Generate a spatial representation of the composite vulnerability index described by custom weights to the three indices created throughout this analyses to visualize most vulnerable census tracts overall in Washington D.C.

```
# Join with spatial data for mapping
dc_map <- dc_tracts %>%
  left_join(indices_data, by = "GEOID")

# Spatial plot with ggplot + plotly
pca_plot <- ggplot(dc_map) +
  geom_sf(aes(fill = composite_index,
              text = paste0("Composite Vulnerability: ", round(composite_index, 2))),
          color = NA) +
  scale_fill_viridis_c(option = "rocket", name = "Composite Vulnerability", direction = -1) +
  labs(title = "Composite Vulnerability Index",
       subtitle = "Census Tracts (2020)",
       caption = "Source: Open Data DC") +
  theme_minimal()

ggplotly(pca_plot, tooltip = "text")
```





Once all the analyses are complete and every index is fully derived, save them to the original data set.

```
# Save results
write.csv(data, "Data/flood_vulnerability_scores.csv", row.names = FALSE)
```