STA 141A Project: Food Insecurity

Brooke Kerstein, Gabriel Jones

2023-09-13

Introduction

Data Wrangling

Libraries

```
list.of.packages <- c("dplyr", "tidyr", "ggplot2", "hrbrthemes", "gganimate",
                      "png", "gifski", "ggridges", "tidyverse", "tibble",
                      "mapview", "sp", "janitor", "GGally", "RColorBrewer",
                      "MASS", "knitr","matlib", "lubridate", "pdftools",
                      "stringr", "ggmap", "ggsci", "patchwork", "ddpcr",
                      "caret", "car", "paletteer")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

lapply(list.of.packages, require, character.only=TRUE)
```

Import All Data Sets

```
rawdata2010 <- readr::read_csv("FeedAmerica/FeedAmerica2010.csv")
rawdata2011 <- readr::read_csv("FeedAmerica/FeedAmerica2011.csv")
rawdata2012 <- readr::read_csv("FeedAmerica/FeedAmerica2012.csv")
rawdata2013 <- readr::read_csv("FeedAmerica/FeedAmerica2013.csv")
rawdata2014 <- readr::read_csv("FeedAmerica/FeedAmerica2014.csv")
rawdata2015 <- readr::read_csv("FeedAmerica/FeedAmerica2015.csv")
rawdata2016 <- readr::read_csv("FeedAmerica/FeedAmerica2016.csv")
rawdata2017 <- readr::read_csv("FeedAmerica/FeedAmerica2017.csv")
rawdata2018 <- readr::read_csv("FeedAmerica/FeedAmerica2018.csv")
rawdata2019_2021 <- readr::read_csv("FeedAmerica/FeedAmerica2019_2021.csv")

AgeData2019 <- readr::read_csv("DisabilityData/DisabilityData2019.csv")

unemploymentdataraw <- readr::read_csv("Local_Area_Unemployment_Statistics__LAUS_.csv")
unemploymentdataraw <- janitor::clean_names(unemploymentdataraw)

incomedataraw <- readr::read_csv("incomedata.csv")
incomedataraw <- janitor::clean_names(incomedataraw)
```

```
DisData2021 <- readr::read_csv("DisabilityData/DisabilityData2021.csv")
DisData2020 <- readr::read_csv("DisabilityData/DisabilityData2020.csv")
DisData2019 <- readr::read_csv("DisabilityData/DisabilityData2019.csv")
DisData2018 <- readr::read_csv("DisabilityData/DisabilityData2018.csv")
DisData2017 <- readr::read_csv("DisabilityData/DisabilityData2017.csv")
DisData2016 <- readr::read_csv("DisabilityData/DisabilityData2016.csv")
DisData2015 <- readr::read_csv("DisabilityData/DisabilityData2015.csv")
DisData2014 <- readr::read_csv("DisabilityData/DisabilityData2014.csv")
DisData2013 <- readr::read_csv("DisabilityData/DisabilityData2013.csv")
DisData2012 <- readr::read_csv("DisabilityData/DisabilityData2012.csv")
DisData2011 <- readr::read_csv("DisabilityData/DisabilityData2011.csv")
DisData2010 <- readr::read_csv("DisabilityData/DisabilityData2010.csv")
```

Clean Up Feed America 2019-2020

```
FeedData2019_2021 <- rawdata2019_2021%>%
  filter(State=="CA")

FeedData2019_2021 <- FeedData2019_2021[,c(3:5)]

FeedData2019_2021 <- janitor::clean_names(FeedData2019_2021)

FeedData2019_2021[,3] <- sapply(FeedData2019_2021[,3],function(x) as.numeric(gsub("%","",x)))

FeedData2019_2021 <- FeedData2019_2021[c("year","county_state","overall_food_insecurity_rate")]
```

Clean Up Feed America 2010-2018

```
cleanFeed <- function(data){

  dataCA <- data%>%
    filter(State=="CA")

  dataCA <- dataCA[,c(3,4)]

  dataCA[,2] <- sapply(dataCA[,2],function(x) as.numeric(gsub("%","",x)))

  colnames(dataCA) <- c("county_state","overall_food_insecurity_rate")

  return (dataCA)
}

FeedData2018 <- cbind("year"=rep(2018,nrow(cleanFeed(rawdata2018))),cleanFeed(rawdata2018))
FeedData2017 <- cbind("year"=rep(2017,nrow(cleanFeed(rawdata2017))),cleanFeed(rawdata2017))
FeedData2016 <- cbind("year"=rep(2016,nrow(cleanFeed(rawdata2016))),cleanFeed(rawdata2016))
FeedData2015 <- cbind("year"=rep(2015,nrow(cleanFeed(rawdata2015))),cleanFeed(rawdata2015))
FeedData2014 <- cbind("year"=rep(2014,nrow(cleanFeed(rawdata2014))),cleanFeed(rawdata2014))
FeedData2013 <- cbind("year"=rep(2013,nrow(cleanFeed(rawdata2013))),cleanFeed(rawdata2013))
FeedData2012 <- cbind("year"=rep(2012,nrow(cleanFeed(rawdata2012))),cleanFeed(rawdata2012))
```

```
FeedData2011 <- cbind("year"=rep(2011,nrow(cleanFeed(rawdata2011))),cleanFeed(rawdata2011))
FeedData2010 <- cbind("year"=rep(2010,nrow(cleanFeed(rawdata2010))),cleanFeed(rawdata2010))
```

Food Insecurity 2010-2021

```
FeedData <- rbind(FeedData2010,FeedData2011,FeedData2012,FeedData2013,FeedData2014,FeedData2015,FeedData

countyNameDisable <- FeedData$county_state

FeedData <- FeedData%>%
  mutate(county_state = gsub(", California", "", county_state))%>%
  rename(county = county_state)%>%
  arrange(county)

countyName <- FeedData$county
```

Cleaning Age Data for 2019 Correlation

```
AgeData2019CA <- AgeData2019 %>%
  filter(NAME %in% c(countyNameDisable, 'Geographic Area Name')) %>%
  row_to_names(row_number = 1)

colnames(AgeData2019CA)[3] <- "Total Population"

Age2019 <- AgeData2019CA %>%
  .[,!grepl("Margin of Error", colnames(.))] %>%
  .[,!grepl("Annotation", colnames(.))] %>%
  .[,!grepl("Percent", colnames(.))] %>%
  .[, grepl("Geographic Area Name|Total Population|Population under 18 years|Population 65 years and ov
  .[,!grepl("years!!|over!!", colnames(.))]

Age2019$'Geographic Area Name' <-gsub(", California", "", Age2019$'Geographic Area Name')

colnames(Age2019)[1] <- "county"

colnames(Age2019) <- gsub(".*\\Estimate!!", "", colnames(Age2019))
colnames(Age2019) <- gsub("Total civilian noninstitutionalized population!!",
                                    "",colnames(Age2019))
colnames(Age2019) <-gsub("DISABILITY TYPE BY DETAILED AGE!!", "", colnames(Age2019))
colnames(Age2019) <- gsub("!!", ": ", colnames(Age2019))
colnames(Age2019) <- gsub(": :", ":", colnames(Age2019))

Age2019 <- Age2019 %>% mutate_at(-1, as.numeric)

TailAge2019 <- Age2019[,c(1:4)]
for (i in c(3:4)) {
  TailAge2019[, i] <- Age2019[, i] / Age2019[, 2]
}

colnames(TailAge2019) <- c("county","total_population","population_under_18","population_over_65")
```

Clean Up Disability 2013-2021

```
cleanDis1 <- function(data){

  dataCA <- data %>%
    filter(NAME %in% c(countyNameDisable, 'Geographic Area Name')) %>%
    row_to_names(row_number = 1)

  colnames(dataCA)[3] <- "Total Population"

  dataCA1 <- dataCA %>%
    .[,!grepl("Margin of Error", colnames(.))] %>%
    .[,!grepl("Annotation", colnames(.))] %>%
    .[, grepl("Geographic Area Name|Percent with a disability",colnames(.))] %>%
    .[,!grepl("population!",colnames(.))]

  dataCA1$'Geographic Area Name' <-gsub(", California", "", dataCA1$'Geographic Area Name')
  colnames(dataCA1)[1] <- "county"
  colnames(dataCA1)[2] <- "Percent with a disability"

  dataCA1 <- dataCA1 %>% mutate_at(-1, as.numeric)

  return (dataCA1)

}

Dis2021 <- cbind("year"=rep(2021,nrow(cleanDis1(DisData2021))),cleanDis1(DisData2021))
Dis2020 <- cbind("year"=rep(2020,nrow(cleanDis1(DisData2020))),cleanDis1(DisData2020))
Dis2019 <- cbind("year"=rep(2019,nrow(cleanDis1(DisData2019))),cleanDis1(DisData2019))
Dis2018 <- cbind("year"=rep(2018,nrow(cleanDis1(DisData2018)[,c(1,2)])),cleanDis1(DisData2018)[,c(1,2)])
Dis2017 <- cbind("year"=rep(2017,nrow(cleanDis1(DisData2017)[,c(1,2)])),cleanDis1(DisData2017)[,c(1,2)])
Dis2016 <- cbind("year"=rep(2016,nrow(cleanDis1(DisData2016)[,c(1,2)])),cleanDis1(DisData2016)[,c(1,2)])
Dis2015 <- cbind("year"=rep(2015,nrow(cleanDis1(DisData2015)[,c(1,2)])),cleanDis1(DisData2015)[,c(1,2)])
Dis2014 <- cbind("year"=rep(2014,nrow(cleanDis1(DisData2014)[,c(1,2)])),cleanDis1(DisData2014)[,c(1,2)])
Dis2013 <- cbind("year"=rep(2013,nrow(cleanDis1(DisData2013)[,c(1,2)])),cleanDis1(DisData2013)[,c(1,2)])
```

Clean Up Disability 2010-2012

```
cleanDis2 <- function(data){

  dataCA <- data %>%
    filter(NAME %in% c(countyNameDisable, 'Geographic Area Name')) %>%
    row_to_names(row_number = 1)

  colnames(dataCA)[3] <- "Total Population"

  dataCA1 <- dataCA %>%
    .[,!grepl("Margin of Error", colnames(.))] %>%
    .[,!grepl("Annotation", colnames(.))] %>%
    .[, grepl("Geographic Area Name|Percent with a disability",colnames(.))] %>%
    .[,!grepl("population!",colnames(.))]
```

```
  dataCA1$'Geographic Area Name' <-gsub(", California", "", dataCA1$'Geographic Area Name')
  colnames(dataCA1)[1] <- "county"
  colnames(dataCA1)[2] <- "Percent with a disability"

  dataCA1 <- (dataCA1[,c(1,2)] %>% mutate_at(-1, as.numeric))

  return (dataCA1)
}

Dis2012 <- cbind("year"=rep(2012,nrow(cleanDis2(DisData2012))),cleanDis2(DisData2012))
Dis2011 <- cbind("year"=rep(2011,nrow(cleanDis2(DisData2011))),cleanDis2(DisData2011))
Dis2010 <- cbind("year"=rep(2010,nrow(cleanDis2(DisData2010))),cleanDis2(DisData2010))
```

Total Disability 2010-2021

```
totalDisability <- rbind(Dis2010,Dis2011,Dis2012,Dis2013,Dis2014,Dis2015,Dis2016,Dis2017,Dis2018,Dis201
totalDisability <- totalDisability %>% arrange(county)
colnames(totalDisability) <- c("year","county","percent_disabled")
```

Clean Up Unemployment 2010-2021

```
UnemploymentData <- unemploymentdataraw%>%
  filter(area_type=="County", status_preliminary_final=="Final")%>%
  filter(year>=2010 & year<2022)%>%
  filter(!area_name %in% c("Non Residential County","Resident Out of State County","Unallocated County"
  group_by(year, area_name)%>%
  summarise("unemployment_rate_avg"=mean(unemployment_rate))%>%
  distinct(.)%>%
  ungroup() %>%
  rename("county"="area_name")
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

Clean Up Income 2010-2020

```
IncomeData <- incomedataraw%>%
  filter(taxable_year >= 2010 & taxable_year <= 2021)%>%
  filter(!county %in% c("Nonresident","Resident Out of State County","Unallocated","Resident Out of Sta
  rename("year"="taxable_year")%>%
  mutate("county"=paste(.$county, "County"))%>%
  arrange(year,county)

IncomeData <- IncomeData[,c(1,2,6)]
```

Final CSV

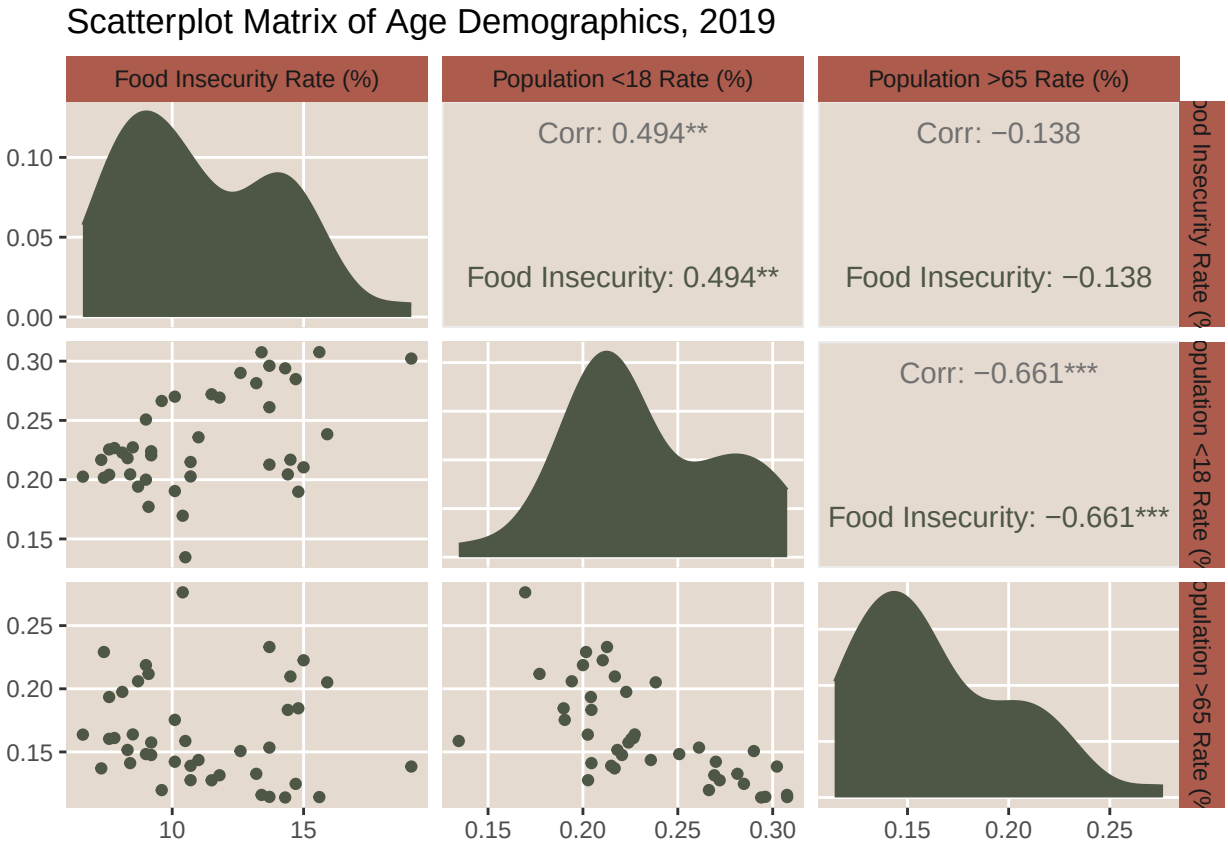
```
majorDF <- FeedData %>% full_join(totalDisability)
majorDF <- majorDF %>% full_join(UnemploymentData)
majorDF <- majorDF %>% full_join(IncomeData)

majorDF <- majorDF %>% filter(!county %in% c("Resident Out of State County","Nonresident20 County","Resi

majorDF2019 <- majorDF %>% filter(year==2019)
majorDF2019 <- TailAge2019 %>% inner_join(majorDF2019)
```

Analysis

Data Visualization

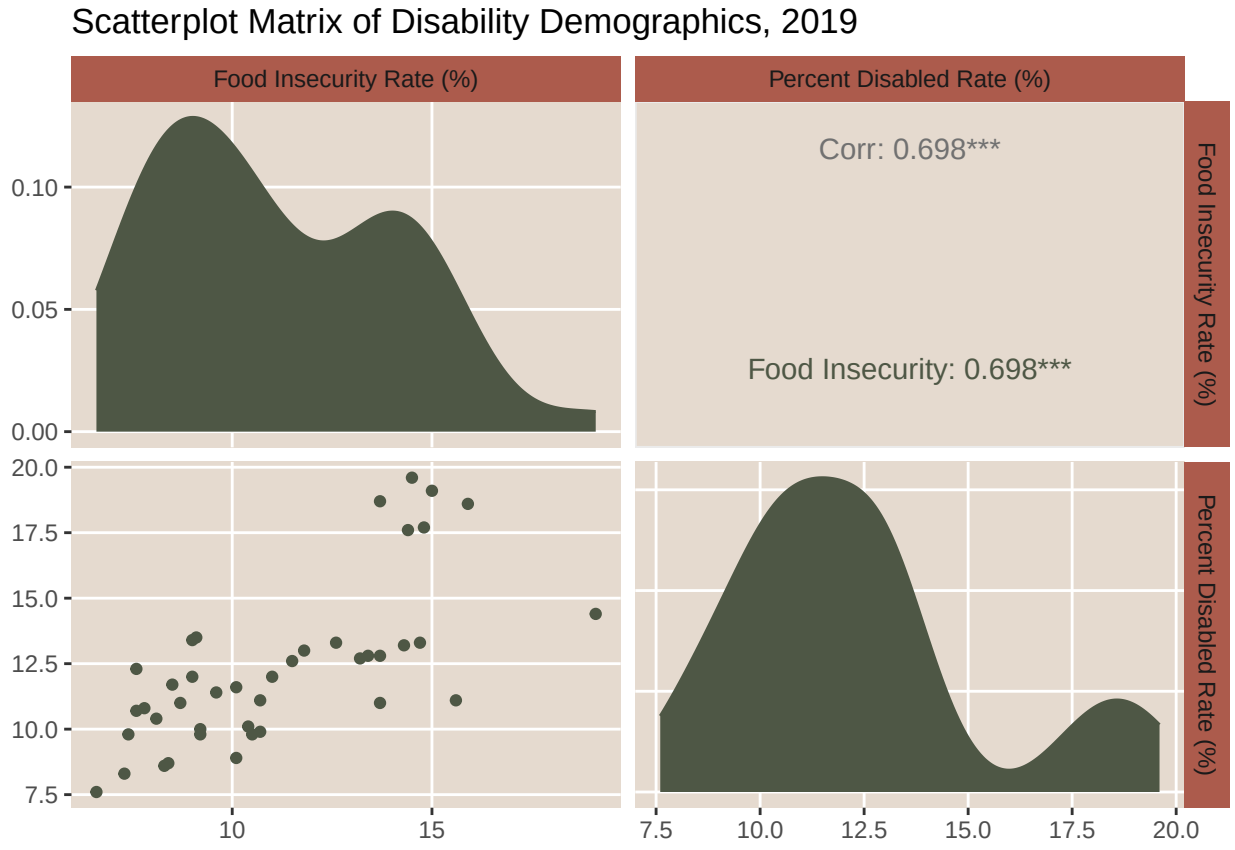
Age Demographics 2019 GGpair

```
ggpairs(majorDF2019[,c(6,3,4)],
        columnLabels=c("Food Insecurity Rate (%)","Population <18 Rate (%)","Population >65 Rate (%)"),
        aes(fill="Food Insecurity",col="Food Insecurity"),
        title = "Scatterplot Matrix of Age Demographics, 2019") +
  scale_fill_manual(values=c("#4E5745"))+
  scale_color_manual(values=c("#4E5745"))+
  theme(strip.background = element_rect(fill = "#AC5B4C"), panel.background=element_rect(fill="#E5DACF"
        panel.grid.minor = element_blank())
```



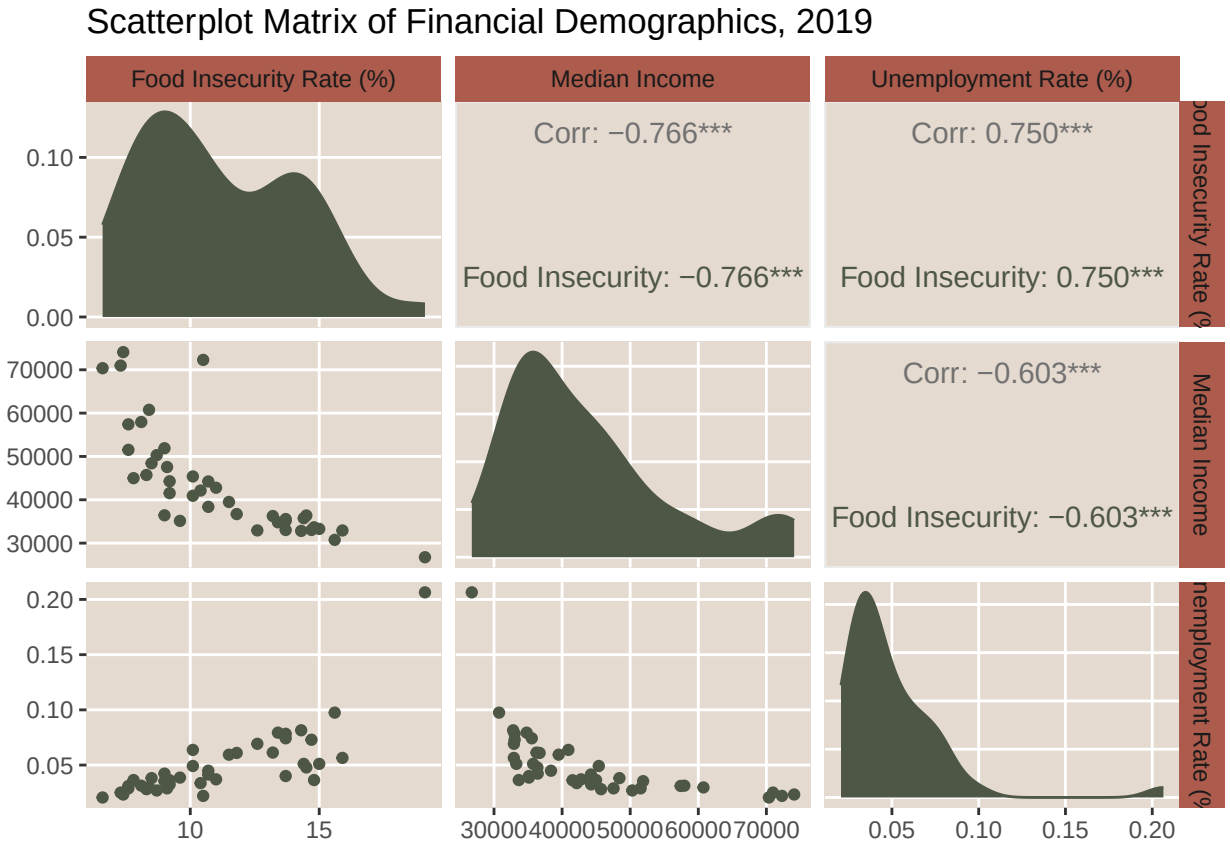
Disability Demographics 2019 GGpair

```
ggpairs(majorDF2019[,c(6,7)],
        columnLabels=c("Food Insecurity Rate (%)","Percent Disabled Rate (%)"),
        aes(fill="Food Insecurity",col="Food Insecurity"),
        title = "Scatterplot Matrix of Disability Demographics, 2019") +
  scale_fill_manual(values=c("#4E5745"))+
  scale_color_manual(values=c("#4E5745"))+
  theme(strip.background = element_rect(fill = "#AC5B4C"), panel.background=element_rect(fill="#E5DACF"
        panel.grid.minor = element_blank())
```



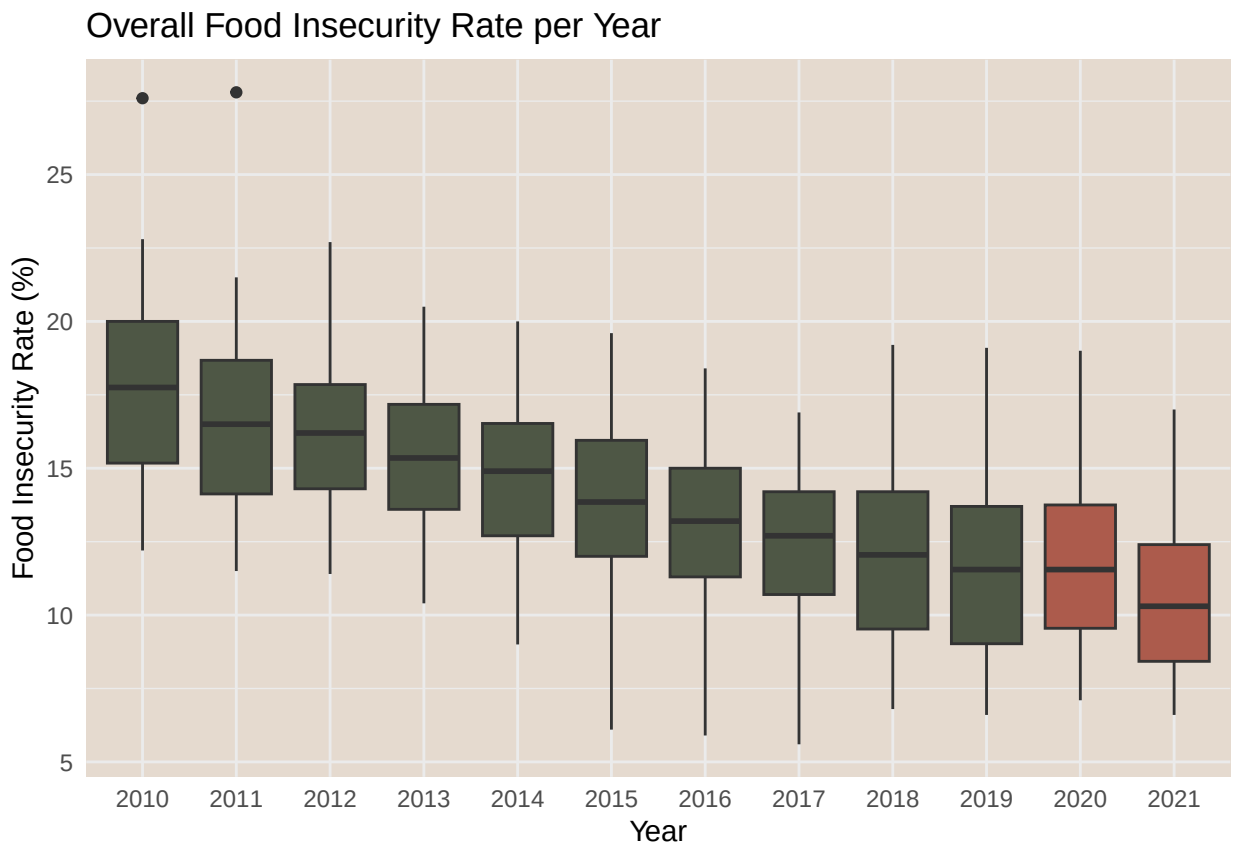
Income Demographics 2019 GGpair

```
ggpairs(majorDF2019[,c(6,9,8)],
        columnLabels=c("Food Insecurity Rate (%)","Median Income","Unemployment Rate (%)"),
        aes(fill="Food Insecurity",col="Food Insecurity"),
        title = "Scatterplot Matrix of Financial Demographics, 2019") +
  scale_fill_manual(values=c("#4E5745"))+
  scale_color_manual(values=c("#4E5745"))+
  theme(strip.background = element_rect(fill = "#AC5B4C"), panel.background=element_rect(fill="#E5DACF"
        panel.grid.minor = element_blank())
```



Food Insecurity Boxplot

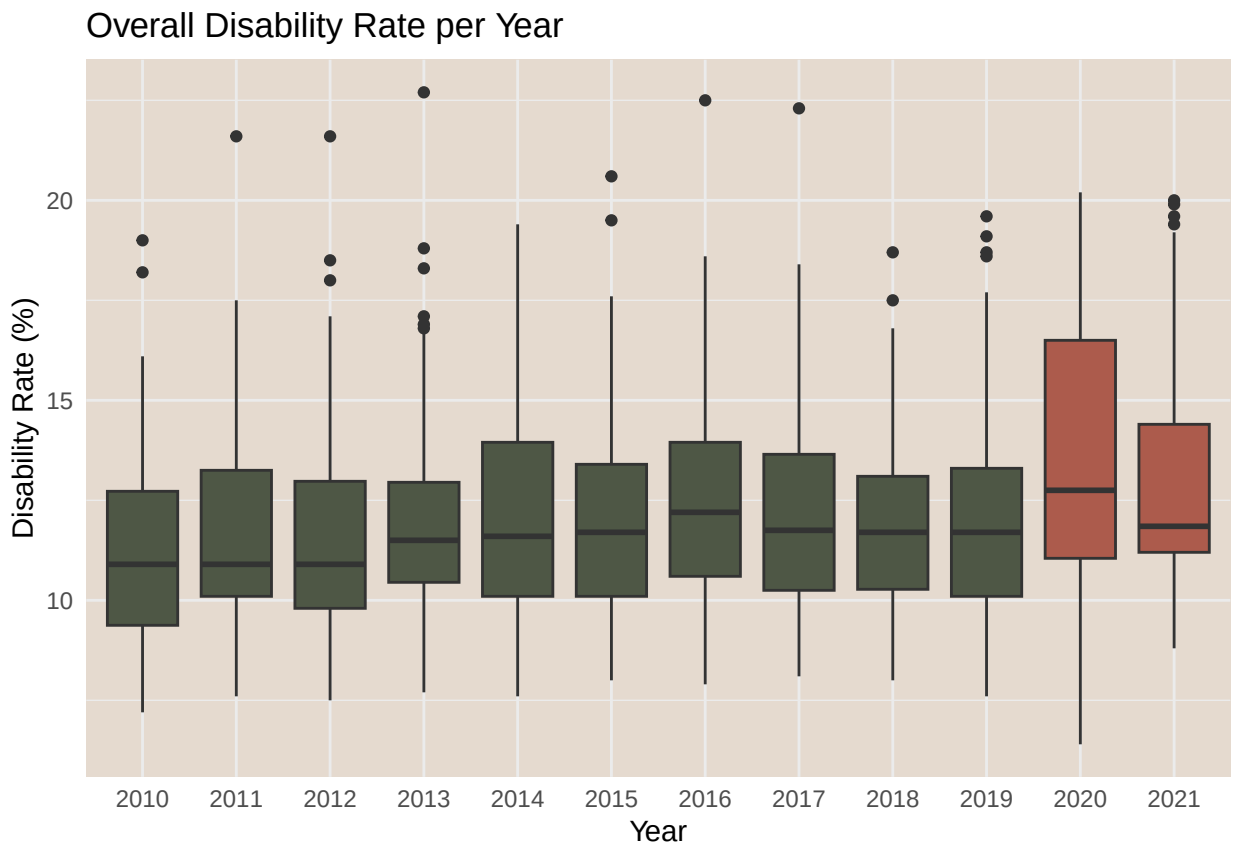
```
majorDF[,c(1,3)] %>%
  mutate(year=as.factor(year))%>%
  ggplot(aes(x=year, y=overall_food_insecurity_rate, fill=(year==c(2020,2021)))) +
  geom_boxplot(show.legend = F)+
  scale_fill_manual(values=c("#4E5745","#AC5B4C"))+
  theme_minimal()+
  theme(panel.background=element_rect(fill="#E5DACF",color="#E5DACF",size=0.5,linetype="solid"))+
  labs(title="Overall Food Insecurity Rate per Year",x="Year",y="Food Insecurity Rate (%)")
```



Disability Boxplot

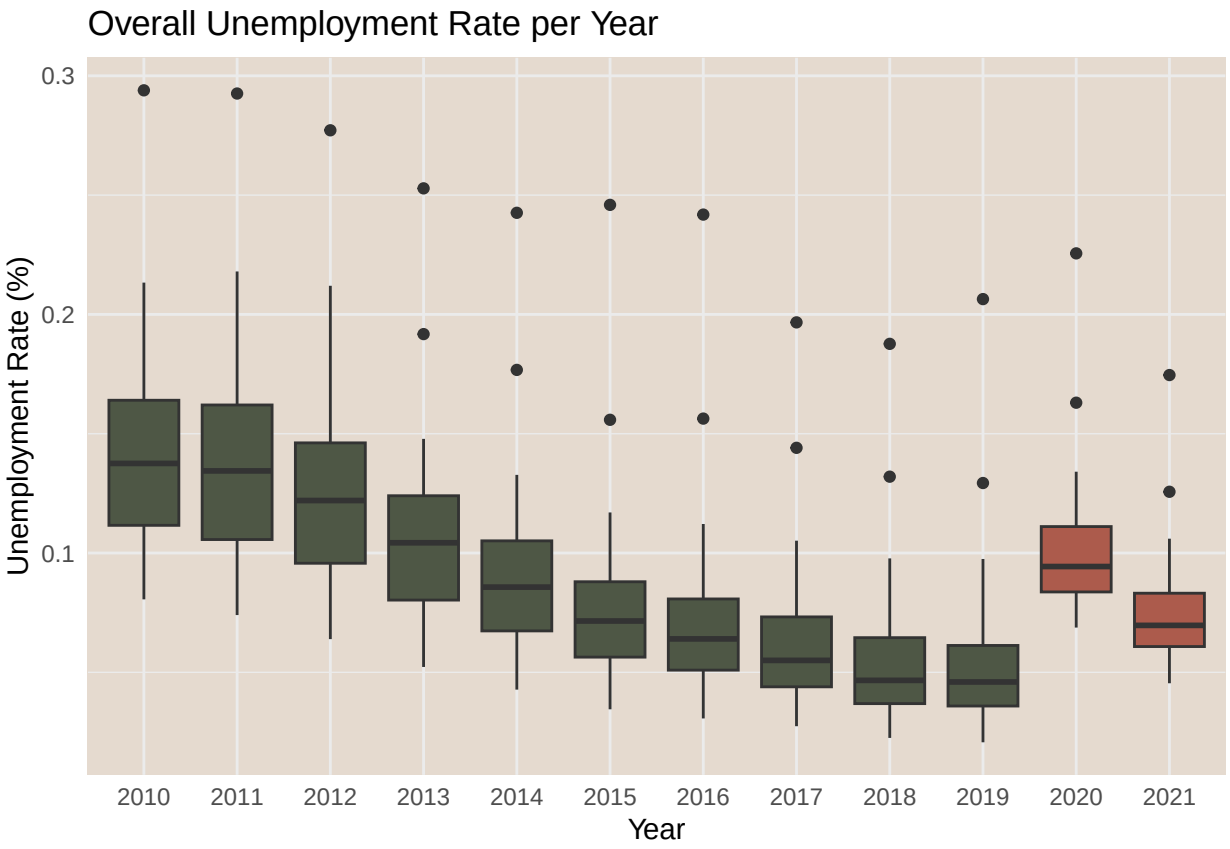
```
majorDF[,c(1,4)] %>%
  mutate(year=as.factor(year))%>%
  ggplot(aes(x=year, y=percent_disabled, fill=(year==2020|year==2021))) +
  geom_boxplot(show.legend = F)+
  scale_fill_manual(values=c("#4E5745","#AC5B4C"))+
  theme_minimal()+
  theme(panel.background=element_rect(fill="#E5DACF",color="#E5DACF",size=0.5,linetype="solid"))+
  labs(title="Overall Disability Rate per Year",x="Year",y="Disability Rate (%)")
```

```
## Warning: Removed 195 rows containing non-finite values (`stat_boxplot()`).
```



Unemployment Boxplot

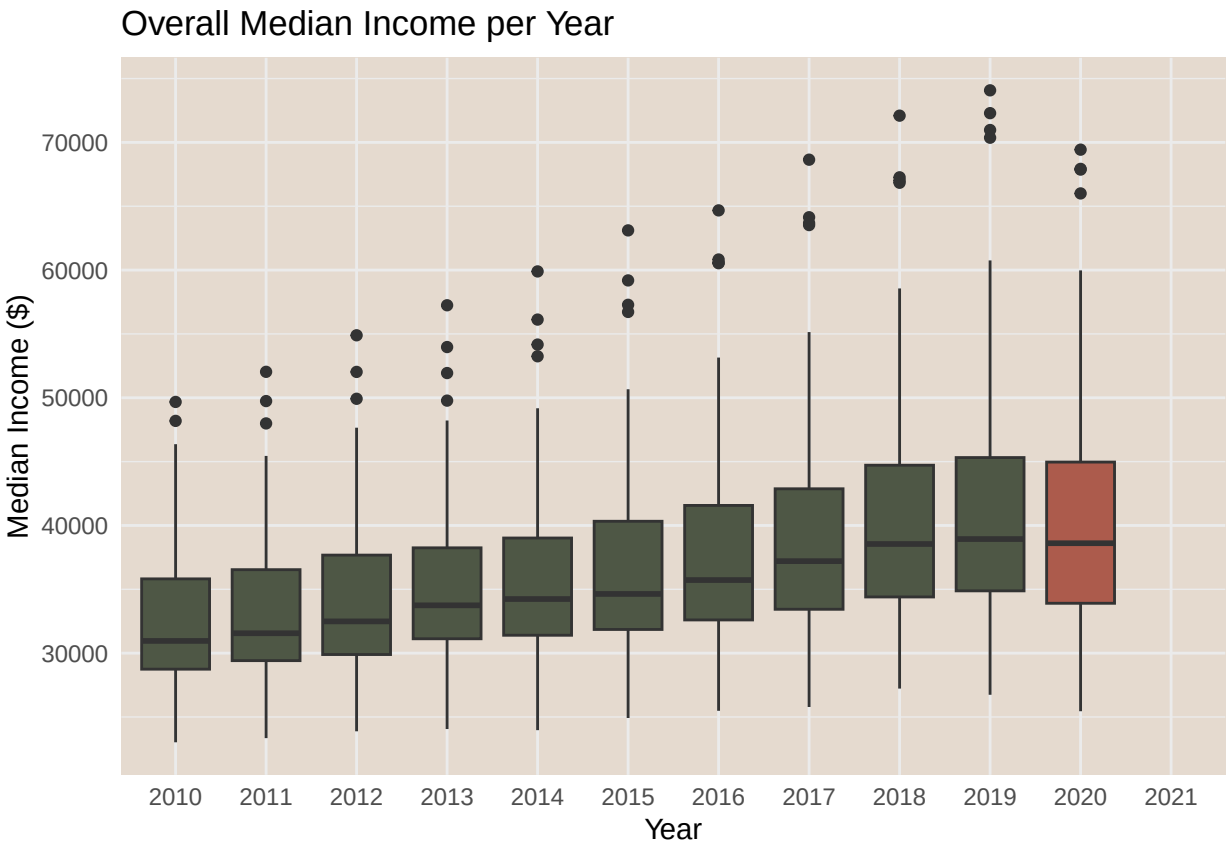
```
majorDF[,c(1,5)] %>%
  mutate(year=as.factor(year))%>%
  ggplot(aes(x=year, y=unemployment_rate_avg, fill=(year==2020 | year==2021))) +
  geom_boxplot(show.legend = F)+
  scale_fill_manual(values=c("#4E5745","#AC5B4C"))+
  theme_minimal()+
  theme(panel.background=element_rect(fill="#E5DACF",color="#E5DACF",size=0.5,linetype="solid"))+
  labs(title="Overall Unemployment Rate per Year",x="Year",y="Unemployment Rate (%)")
```



Income Boxplot

```
majorDF[,c(1,6)]%>%
  mutate(year=as.factor(year))%>%
  ggplot(aes(x=year, y=median_income, fill=(year==2020))) +
  geom_boxplot(show.legend = F)+
  scale_fill_manual(values=c("#4E5745","#AC5B4C"))+
  theme_minimal()+
  theme(panel.background=element_rect(fill="#E5DACF",color="#E5DACF",size=0.5,linetype="solid"))+
  labs(title="Overall Median Income per Year",x="Year",y="Median Income ($)")
```

```
## Warning: Removed 58 rows containing non-finite values (`stat_boxplot()`).
```



2019 Map Set-up

```
shape <- sf::read_sf(dsn = "CA_Counties_ShapeFile", layer = "CA_Counties_TIGER2016")
counties <- shape['NAME'] %>% arrange(NAME)
colnames(counties)[1] <- "county"

majorDF2019$county <- gsub(" County","",majorDF2019$county)
counties <- counties %>% full_join(majorDF2019)
```

2019 Map Execution

```
map <- mapview(
  counties,
  zcol = "overall_food_insecurity_rate",
  layer.name = "Food Insecurity Rate (%)",
  map.types = "CartoDB.Positron",
  na.color = "#AC5B4C",
  col.regions = brewer.pal(100, "Blues"),
  alpha.regions = 1
) +
  mapview(
    counties,
```

```
    zcol = "unemployment_rate_avg",
    layer.name = "Unemployment Rate (%)",
    map.types = "CartoDB.Positron",
    na.color = "#AC5B4C",
    col.regions = brewer.pal(100, "Blues"),
    alpha.regions = 1
  ) +
  mapview(
    counties,
    zcol = "median_income",
    layer.name = "Median Income",
    map.types = "CartoDB.Positron",
    na.color = "#AC5B4C",
    col.regions = brewer.pal(100, "Blues"),
    alpha.regions = 1
  ) +
  mapview(
    counties,
    zcol = "percent_disabled",
    layer.name = "Disability Rate (%)",
    map.types = "CartoDB.Positron",
    na.color = "#AC5B4C",
    col.regions = brewer.pal(100, "Blues"),
    alpha.regions = 1
  )
```

Modeling

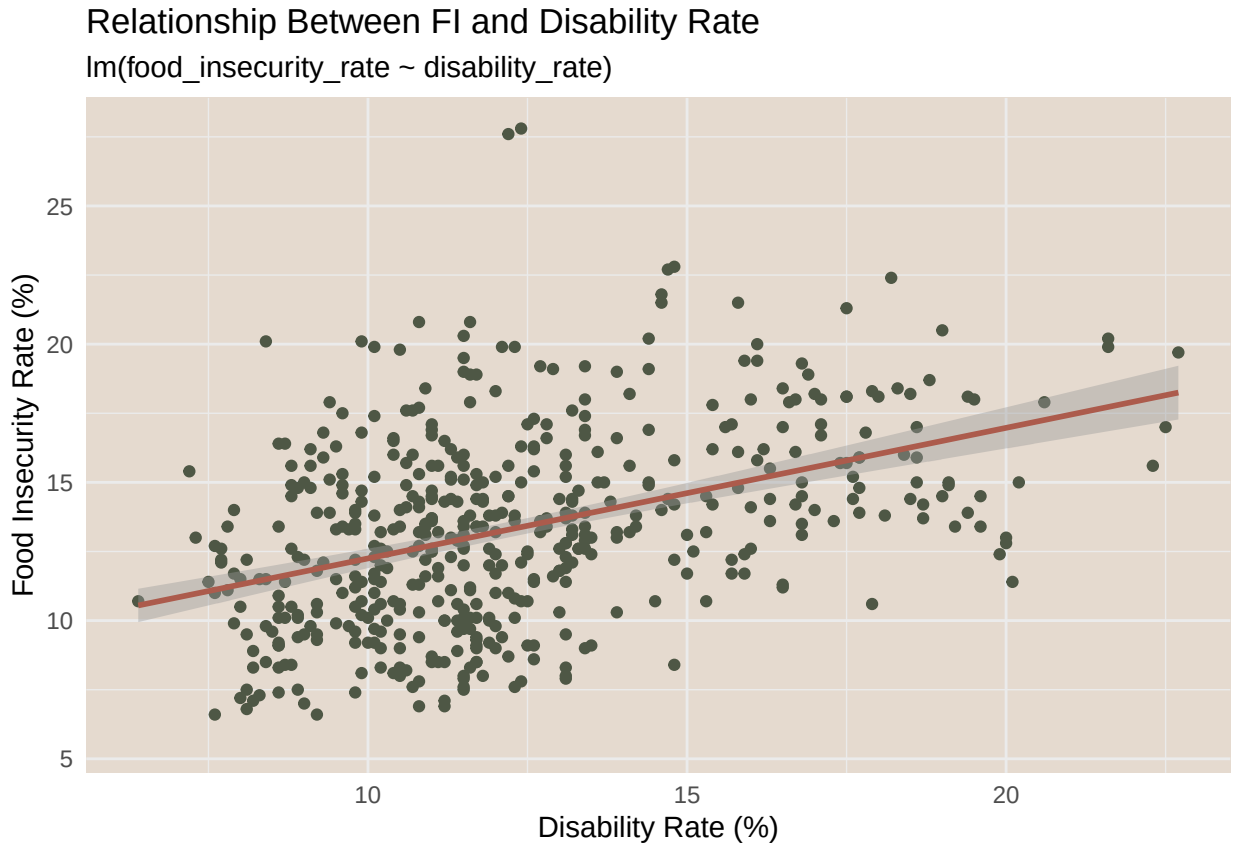
Food Insecurity vs. Disability Linear Model

```
majorDF%>%
  ggplot(aes(x=percent_disabled, y=overall_food_insecurity_rate))+
  geom_point(color="#4E5745")+
  geom_smooth(method="lm", color="#AC5B4C", show.legend = F)+
  theme_minimal()+
  theme(panel.background = element_rect(fill="#E5DACF", color = "#E5DACF", size = 0.5, linetype = "soli
  labs(title="Relationship Between FI and Disability Rate",
       subtitle="lm(food_insecurity_rate ~ disability_rate)",
       x="Disability Rate (%)",
       y="Food Insecurity Rate (%)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 195 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 195 rows containing missing values (`geom_point()`).
```



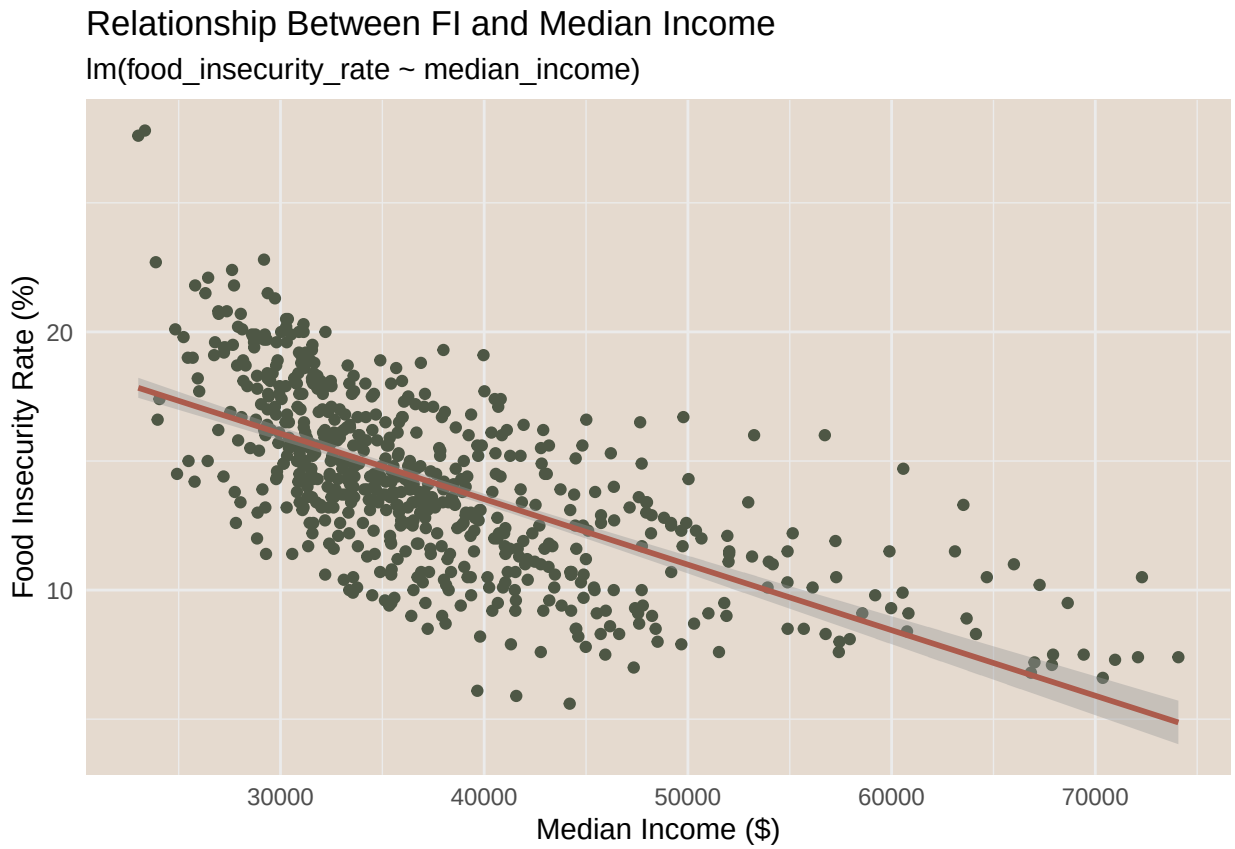
Food Insecurity vs. Median Income Linear Model

```
majorDF%>%
  ggplot(aes(x=median_income, y=overall_food_insecurity_rate))+
  geom_point(color="#4E5745")+
  geom_smooth(method="lm", color="#AC5B4C", show.legend = F)+
  theme_minimal()+
  theme(panel.background = element_rect(fill="#E5DACF", color = "#E5DACF", size = 0.5, linetype = "soli
  labs(title="Relationship Between FI and Median Income",
       subtitle="lm(food_insecurity_rate ~ median_income)",
       x="Median Income ($)",
       y="Food Insecurity Rate (%)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 58 rows containing non-finite values (`stat_smooth()`).
```

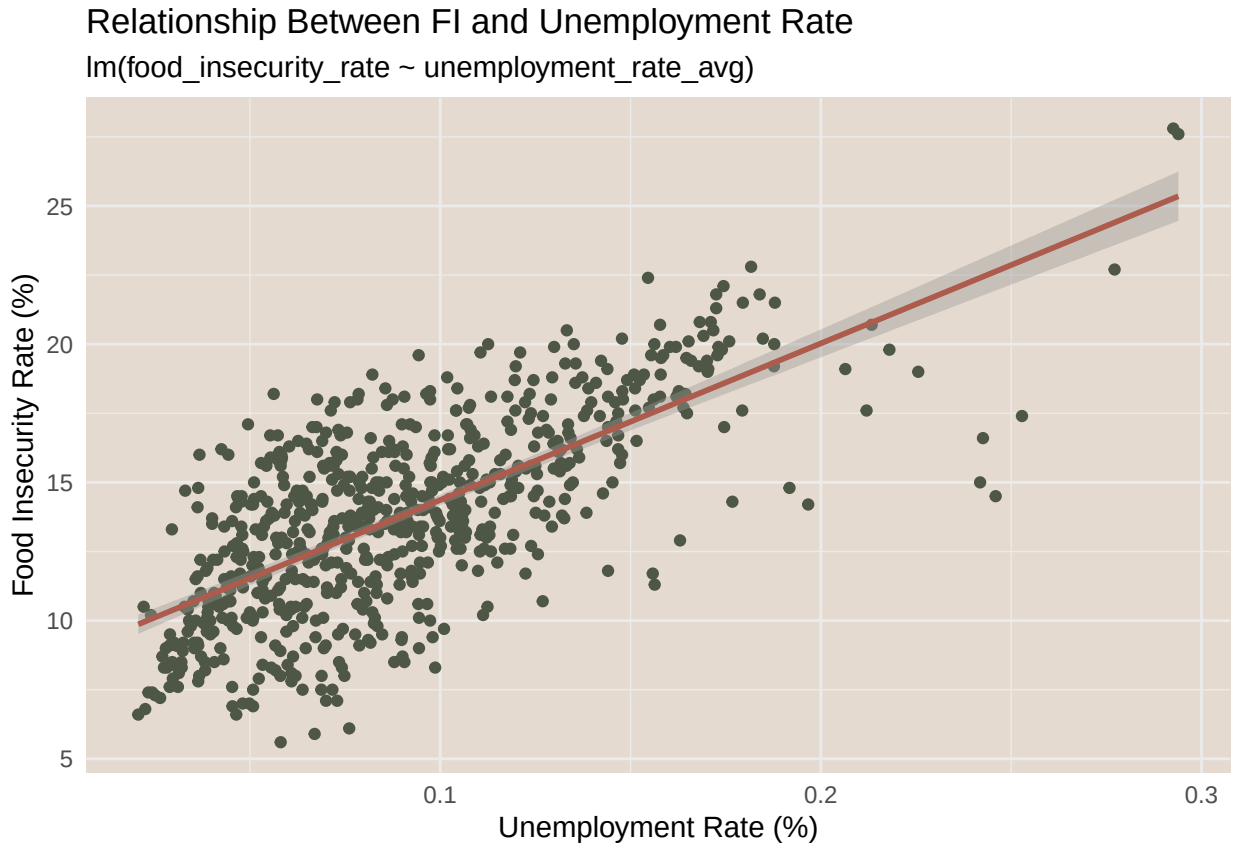
```
## Warning: Removed 58 rows containing missing values (`geom_point()`).
```



Food Insecurity vs. Unemployment Linear Model

```
majorDF%>%
  ggplot(aes(x=unemployment_rate_avg, y=overall_food_insecurity_rate))+
  geom_point(color="#4E5745")+
  geom_smooth(method="lm", color="#AC5B4C", show.legend = F)+
  theme_minimal()+
  theme(panel.background = element_rect(fill="#E5DACF", color = "#E5DACF", size = 0.5, linetype = "soli
  labs(title="Relationship Between FI and Unemployment Rate",
       subtitle="lm(food_insecurity_rate ~ unemployment_rate_avg)",
       x="Unemployment Rate (%)",
       y="Food Insecurity Rate (%)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



AIC

```
none_mod <- lm(overall_food_insecurity_rate~1,data=na.omit(majorDF)) ##model with only intercept
full_mod <- lm(overall_food_insecurity_rate~unemployment_rate_avg+median_income+percent_disabled,data=na

stepAIC(none_mod, scope=list(upper=full_mod, lower = ~1), direction="both", k=2)
```

```
## Start:  AIC=1137.21
## overall_food_insecurity_rate ~ 1
##
##                          Df Sum of Sq    RSS     AIC
## + unemployment_rate_avg   1    2881.1 2562.9  793.40
## + median_income           1    2393.8 3050.2  873.31
## + percent_disabled        1    1047.3 4396.7 1041.13
## <none>                                5444.0 1137.21
##
## Step:  AIC=793.4
## overall_food_insecurity_rate ~ unemployment_rate_avg
##
##                          Df Sum of Sq    RSS     AIC
## + percent_disabled        1    511.02 2051.8  693.32
## + median_income           1    355.04 2207.8  726.96
## <none>                                2562.9  793.40
## - unemployment_rate_avg   1   2881.14 5444.0 1137.21
```

```
##
## Step:  AIC=693.32
## overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled
##
##                          Df Sum of Sq    RSS     AIC
## + median_income           1     96.57 1955.3  673.20
## <none>                                2051.8  693.32
## - percent_disabled        1    511.02 2562.9  793.40
## - unemployment_rate_avg   1   2344.83 4396.7 1041.13
##
## Step:  AIC=673.2
## overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled +
##     median_income
##
##                          Df Sum of Sq    RSS    AIC
## <none>                                1955.3 673.20
## - median_income           1     96.57 2051.8 693.32
## - percent_disabled        1    252.55 2207.8 726.96
## - unemployment_rate_avg   1    967.15 2922.4 855.66

##
## Call:
## lm(formula = overall_food_insecurity_rate ~ unemployment_rate_avg +
##     percent_disabled + median_income, data = na.omit(majorDF))
##
## Coefficients:
##           (Intercept)  unemployment_rate_avg       percent_disabled
##             9.159e+00              4.261e+01              2.716e-01
##         median_income
##            -6.762e-05
```

[Overall against 2020] Linear Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```r
# Define Training Data (2010-2019) and Test Data (2020)
majorDFtrain <- na.omit(majorDF)%>%
  filter(year<2020)

majorDFtest <- na.omit(majorDF)%>%
  filter(year==2020)

model <- lm(overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled + median_income, da

lm_results <- data.frame(
  "county" = majorDFtest$county,
  "predicted_food_insecurity_rate" = as.numeric(model$coefficients[1]) + as.numeric(model$coefficients[
    majorDFtest$unemployment_rate_avg + as.numeric(model$coefficients[3]) *
    majorDFtest$percent_disabled + as.numeric(model$coefficients[4]) * majorDFtest$median_income,
  "actual_food_insecurity_rate" = majorDFtest$overall_food_insecurity_rate
)

lm_overall_acc20 <- 1 - mean((abs(lm_results$predicted_food_insecurity_rate-lm_results$actual_food_inse
```

[Overall against 2019] Linear Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```
# Define Training Data (2010-2019) and Test Data (2019)
majorDFtrain <- na.omit(majorDF)%>%
  filter(year<2019)

majorDFtest <- na.omit(majorDF)%>%
  filter(year==2019)

model <- lm(overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled + median_income, da

lm_results <- data.frame(
  "county" = majorDFtest$county,
  "predicted_food_insecurity_rate" = as.numeric(model$coefficients[1]) + as.numeric(model$coefficients[
    majorDFtest$unemployment_rate_avg + as.numeric(model$coefficients[3]) *
    majorDFtest$percent_disabled + as.numeric(model$coefficients[4]) * majorDFtest$median_income,
  "actual_food_insecurity_rate" = majorDFtest$overall_food_insecurity_rate
)

lm_overall_acc19 <- 1 - mean((lm_results$predicted_food_insecurity_rate-lm_results$actual_food_insecuri
```

[Overall against 2020] Random Forest Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```
# Define Training Data (2010-2019) and Test Data (2020)
majorDFtrain <- na.omit(majorDF)%>%
  filter(year<2020)

majorDFtest <- na.omit(majorDF)%>%
  filter(year==2020)

train_rf <- train(overall_food_insecurity_rate ~ unemployment_rate_avg + median_income + percent_disabl
                  tuneGrid=data.frame(mtry=1:2),
                  trControl=trainControl(method="cv", number=5))

pred <- as.numeric(predict(train_rf, newdata=majorDFtest))

rf_results <- data.frame("county"=majorDFtest$county,
                         "predicted_food_insecurity_rate"= pred,
                         "actual_food_insecurity_rate"= majorDFtest$overall_food_insecurity_rate)

rf_overall_acc20 <- 1 - mean((rf_results$predicted_food_insecurity_rate-rf_results$actual_food_insecuri
```

[Overall against 2019] Random Forest Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```
# Define Training Data (2010-2019) and Test Data (2020)
majorDFtrain <- na.omit(majorDF)%>%
```

```
  filter(year<2019)

majorDFtest <- na.omit(majorDF)%>%
  filter(year==2019)

train_rf <- train(overall_food_insecurity_rate ~ unemployment_rate_avg + median_income + percent_disabl
                  tuneGrid=data.frame(mtry=1:2),
                  trControl=trainControl(method="cv", number=5))

pred <- as.numeric(predict(train_rf, newdata=majorDFtest))

rf_results <- data.frame("county"=majorDFtest$county,
                         "predicted_food_insecurity_rate"= pred,
                         "actual_food_insecurity_rate"= majorDFtest$overall_food_insecurity_rate)

rf_overall_acc19 <- 1 - mean((rf_results$predicted_food_insecurity_rate-rf_results$actual_food_insecuri
```

[By county against 2019,2020] Linear Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```
lm_trainr <- function(c,y) {
  # Define Training Data (2010-2019) and Test Data (2020)
  majorDFtrain <- na.omit(majorDF) %>%
    filter(year < y)%>%
    filter(county==c)

  majorDFtest <- na.omit(majorDF) %>%
    filter(year < y)%>%
    filter(county==c)

  if (nrow(majorDFtrain) == 0 | nrow(majorDFtest) == 0){

    rf_results <- data.frame("county"=c,
           "predicted_food_insecurity_rate"=NA,
           "actual_food_insecurity_rate"=NA)

    rf_acc <- NA

    return(list(results=rf_results, accuracy=rf_acc))
  }

  model <-
    lm(
      overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled + median_income,
      data = majorDFtrain
    )

  lm_results <- data.frame(
    "county" = majorDFtrain$county,
    "predicted_food_insecurity_rate" = as.numeric(model$coefficients[1]) + as.numeric(model$coefficient
      majorDFtrain$unemployment_rate_avg + as.numeric(model$coefficients[3]) *
```

```
      majorDFtrain$percent_disabled + as.numeric(model$coefficients[4]) * majorDFtrain$median_income,
    "actual_food_insecurity_rate" = majorDFtrain$overall_food_insecurity_rate
  )

  lm_acc <-
    1 - mean(
      abs(
        lm_results$predicted_food_insecurity_rate - lm_results$actual_food_insecurity_rate
      ) / lm_results$actual_food_insecurity_rate
    )

  return(list(results = lm_results, accuracy = lm_acc))
}

results20 <- data.frame(county=c(), predicted_food_insecurity_rate=c(), actual_food_insecurity=c())
accuracy20 <- c()

for (c in unique(majorDF$county)){
  trainr_data <- lm_trainr(c,2020)
  #results20 <- rbind(results20, trainr_data[[1]])
  accuracy20 <- c(accuracy20, trainr_data[[2]])
}

#results20$accuracy20 <- accuracy20
lm_county_acc20 <- mean(accuracy20, na.rm=T)

results19 <- data.frame(county=c(), predicted_food_insecurity_rate=c(), actual_food_insecurity=c())
accuracy19 <- c()

for (c in unique(majorDF$county)){
  trainr_data <- lm_trainr(c,2019)
  results19 <- rbind(results19, trainr_data[[1]])
  accuracy19 <- c(accuracy19, trainr_data[[2]])
}

#results19$accuracy19 <- accuracy19
lm_county_acc19 <- mean(accuracy19, na.rm=T)
```

[By county against 2019,2020] Linear Model: Food Insecurity ~ Unemployment Rate + Disability Rate + Median Income

```
rf_trainr <- function(c,y) {
  # Define Training Data (2010-2019) and Test Data (2020)
  majorDFtrain <- na.omit(majorDF) %>%
    filter(year < y)%>%
    filter(county==c)

  majorDFtest <- na.omit(majorDF) %>%
    filter(year < y)%>%
    filter(county==c)
```

```
  if (nrow(majorDFtrain) <= 1 | nrow(majorDFtest) == 0){

    rf_results <- data.frame("county"=c,
           "predicted_food_insecurity_rate"=NA,
           "actual_food_insecurity_rate"=NA)

    rf_acc <- NA

    return(list(results=rf_results, accuracy=rf_acc))
  }

  train_rf <-
    train(
      overall_food_insecurity_rate ~ unemployment_rate_avg + percent_disabled + median_income,
      method = "rf",
      data = majorDFtrain,
      tuneGrid = data.frame(mtry = 1:2),
      trControl = trainControl(method = "cv", number = 5)
    )

  pred <- as.numeric(predict(train_rf, newdata=majorDFtest))

  rf_results <- data.frame("county"=majorDFtest$county,
                           "predicted_food_insecurity_rate"= pred,
                           "actual_food_insecurity_rate"= majorDFtest$overall_food_insecurity_rate)

  rf_acc <-
    1 - mean((
      rf_results$predicted_food_insecurity_rate - rf_results$actual_food_insecurity_rate
    ) / rf_results$actual_food_insecurity_rate
    )

  return(list(results=rf_results, accuracy=rf_acc))
}

results20 <- data.frame(county=c(), predicted_food_insecurity_rate=c(), actual_food_insecurity=c())
accuracy20 <- c()

for (c in unique(majorDF$county)){
  trainr_data <- rf_trainr(c,2020)
  #results20 <- rbind(results20, trainr_data[[1]])
  accuracy20 <- c(accuracy20, trainr_data[[2]])
}

#results20$accuracy20 <- accuracy20
rf_county_acc20 <- mean(accuracy20, na.rm=T)

results19 <- data.frame(county=c(), predicted_food_insecurity_rate=c(), actual_food_insecurity=c())
accuracy19 <- c()

for (c in unique(majorDF$county)){
  trainr_data <- rf_trainr(c,2019)
  results19 <- rbind(results19, trainr_data[[1]])
```

```
  accuracy19 <- c(accuracy19, trainr_data[[2]])
}

#results19$accuracy19 <- accuracy19
rf_county_acc19 <- mean(accuracy19, na.rm=T)
```

Predictive Model Results

```
kable(data.frame("Model Type"=c("Overall Linear Regression", "Overall Random Forest",
                                "By County Linear Model", "By County Random Forest"),
                 "Test Accuracy 2019"=c(lm_overall_acc19, rf_overall_acc19,
                                        lm_county_acc19, rf_county_acc19),
                 "Test Accuracy 2020"=c(lm_overall_acc20, rf_overall_acc20,
                                        lm_county_acc20, rf_county_acc20)))
```

Model.Type	Test.Accuracy.2019	Test.Accuracy.2020
Overall Linear Regression	0.8671502	0.6799830
Overall Random Forest	0.9323469	0.6666605
By County Linear Model	0.9758715	0.9738838
By County Random Forest	0.9948321	0.9955977

Conclusion