Using Machine Learning to Reduce Food Wastage and Fight Hunger

Guruprasad Velikadu Krishnamoorthy

2024-06-30

```r
# Calling the Libraries used
library(readxl)
library(dplyr)
library(lubridate)
library(readr)
library(ggplot2)
library(ggthemes)
library(tidyr)
library(DT)
library(scales)
library(stringr)
library(knitr)
library(FactoMineR)
library(ggpubr)
library(kableExtra)
library(magrittr)
library(ggfortify)
library(reshape2)
library(treemap)
library(leaflet)
library(plotly)
library(gt)
library(forcats)
library(caret)
library(fastDummies)
library(tidyverse)
library(purrr)
library(vtreat)
library(broom)
library(tidymodels)
library(tidyverse)
library(reshape2)
library(plyr)
library(scales)
library(corrplot)
library(ggthemes)
library(ggalt)
library(maps)
library(ggdendro)
library(crosstalk)
library(zoo)
library(glmnet)
library(dplyr)
library(psych)
library(ggcorrplot)
library(factoextra)
library(recipes)
# Setting the preferance
tidymodels_prefer()
conflicted::conflicts_prefer(dplyr::mutate)
conflicted::conflicts_prefer(dplyr::count)
conflicted::conflicts_prefer(dplyr::arrange)
```

Data Extraction

```
# Reading the data from csv file and loading into a Dataframe
food_surplus_dtl_df_orig <-  read_csv("US_State_Food_Surplus_Detail.csv")
food_surplus_summ_df_orig <-  read_csv("US_State_Food_Surplus_Summary.csv")
```

```
# Printing few rows from Dataframe for Display
food_surplus_dtl_df <- food_surplus_dtl_df_orig
food_surplus_summ_df <- food_surplus_summ_df_orig
kbl(head(food_surplus_dtl_df),caption = "Food Surplus Detail Dataframe", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
```

Food Surplus Detail Dataframe

year	state	sector	sub_sector	sub_sector_category	food_type	food_category	tons_su
2022	Alabama	Farm	Not Applicable	NA	Dry Goods	Nuts And Seeds	$8531.13
2022	Alabama	Farm	Not Applicable	NA	Produce	Blueberries	$139.167
2022	Alabama	Farm	Not Applicable	NA	Produce	Cucumbers	$7201.30
2022	Alabama	Farm	Not Applicable	NA	Produce	Peaches	$778.298
2022	Alabama	Farm	Not Applicable	NA	Produce	Potatoes	$7545.41
2022	Alabama	Farm	Not Applicable	NA	Produce	Sweet Corn	$1136.96

```
# Printing the summary of the dataframe
summary(food_surplus_dtl_df)
```

```
##       year         state              sector            sub_sector       
##  Min.   :2010   Length:559855      Length:559855      Length:559855     
##  1st Qu.:2013   Class :character   Class :character   Class :character  
##  Median :2016   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :2016                                                           
##  3rd Qu.:2019                                                           
##  Max.   :2022                                                           
##                                                                         
##  sub_sector_category  food_type          food_category      tons_surplus      
##  Length:559855       Length:559855      Length:559855      Length:559855     
##  Class :character    Class :character   Class :character   Class :character  
##  Mode  :character    Mode  :character   Mode  :character   Mode  :character  
##                                                                              
##                                                                              
##                                                                              
##                                                                              
##  tons_supply        us_dollars_surplus  tons_waste         tons_uneaten      
##  Length:559855      Length:559855      Length:559855      Length:559855     
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##                                                                             
##  tons_donated       tons_biomaterial_processing tons_animal_feed  
##  Length:559855      Length:559855               Length:559855     
##  Class :character   Class :character            Class :character  
##  Mode  :character   Mode  :character            Mode  :character  
##                                                                   
##                                                                   
##                                                                   
##                                                                   
##  tons_anaerobically_digested tons_composted     tons_not_harvested
##  Length:559855               Length:559855      Length:559855     
##  Class :character            Class :character   Class :character  
##  Mode  :character            Mode  :character   Mode  :character  
##                                                                   
##                                                                   
##                                                                   
##                                                                   
##  tons_incinerated   tons_land_application tons_landfilled    
##  Min.   :    0.00   Min.   :0.00e+00      Min.   :     0.00  
##  1st Qu.:    0.00   1st Qu.:0.00e+00      1st Qu.:     8.81  
##  Median :    0.00   Median :0.00e+00      Median :    50.39  
##  Mean   :   90.32   Mean   :8.18e+01      Mean   :   699.67  
##  3rd Qu.:    7.56   3rd Qu.:1.00e-01      3rd Qu.:   284.92  
##  Max.   :54371.24   Max.   :1.44e+06      Max.   :218105.81  
##  NA's   :18104      NA's   :18104         NA's   :18104      
##    tons_sewer         tons_refuse_discards upstream_mtco2e_footprint
##  Min.   :     0.00   Min.   :     0.00    Min.   :      0          
##  1st Qu.:     0.00   1st Qu.:     0.00    1st Qu.:     79          
##  Median :     0.00   Median :     0.00    Median :    466          
##  Mean   :   153.43   Mean   :    14.42    Mean   :   8037          
##  3rd Qu.:     7.27   3rd Qu.:     0.00    3rd Qu.:   2929          
```

```
##  Max.   :180511.31   Max.   :150613.52    Max.   :6569443
##  NA's   :18104       NA's   :18104        NA's   :18104
##  downstream_mtco2e_footprint total_mtco2e_footprint gallons_water_footprint
##  Min.   :-71723.59           Min.   :      0        Min.   :1.180e+03
##  1st Qu.:     6.57           1st Qu.:     89        1st Qu.:1.991e+06
##  Median :    39.33           Median :    519        Median :1.423e+07
##  Mean   :   584.61           Mean   :   8622        Mean   :4.857e+08
##  3rd Qu.:   239.72           3rd Qu.:   3219        3rd Qu.:1.103e+08
##  Max.   :163780.38           Max.   :6559442        Max.   :5.620e+11
##  NA's   :18104               NA's   :18104          NA's   :18104
##   meals_wasted
##  Min.   :1.000e+02
##  1st Qu.:3.190e+04
##  Median :1.794e+05
##  Mean   :3.326e+06
##  3rd Qu.:1.054e+06
##  Max.   :3.649e+09
##  NA's   :18104
```

```
# Examining teh sructure of teh Dataframe
str(food_surplus_dtl_df)
```

```
## spc_tbl_ [559,855 × 28] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ year                        : num [1:559855] 2022 2022 2022 2022 2022 ...
##  $ state                       : chr [1:559855] "Alabama" "Alabama" "Alabama" "Alabama"
...
##  $ sector                      : chr [1:559855] "Farm" "Farm" "Farm" "Farm" ...
##  $ sub_sector                  : chr [1:559855] "Not Applicable" "Not Applicable" "Not A
pplicable" "Not Applicable" ...
##  $ sub_sector_category         : chr [1:559855] NA NA NA NA ...
##  $ food_type                   : chr [1:559855] "Dry Goods" "Produce" "Produce" "Produce
" ...
##  $ food_category               : chr [1:559855] "Nuts And Seeds" "Blueberries" "Cucumber
s" "Peaches" ...
##  $ tons_surplus                : chr [1:559855] "$8531.13769" "$139.16745" "$7201.30472
8" "$778.2985368" ...
##  $ tons_supply                 : chr [1:559855] "$288781.1377" "$388.26" "$38176.30473"
"$2048.1054" ...
##  $ us_dollars_surplus          : chr [1:559855] "$4443485.087" "$494817.6" "$3575427.45
5" "$2140320.976" ...
##  $ tons_waste                  : chr [1:559855] "$8531.13769" "$124.1549206" "$7201.3047
28" "$613.0090722" ...
##  $ tons_uneaten                : chr [1:559855] "$8531.13769" "$138.0899453" "$7201.3047
28" "$766.4351011" ...
##  $ tons_donated                : chr [1:559855] "$0" "$1.077504715" "$0" "$11.86343575"
...
##  $ tons_biomaterial_processing: chr [1:559855] "$0" "$0" "$0" "$0" ...
##  $ tons_animal_feed            : chr [1:559855] "$0" "$13.93502464" "$0" "$153.4260289"
...
##  $ tons_anaerobically_digested: chr [1:559855] "$0" "$0" "$0" "$0" ...
##  $ tons_composted              : chr [1:559855] "$0" "$0" "$0" "$0" ...
##  $ tons_not_harvested          : chr [1:559855] "$8531.13769" "$118.26" "$7201.304728"
"$548.1054004" ...
##  $ tons_incinerated            : num [1:559855] 0 0.0208 0 0.2296 3.7992 ...
##  $ tons_land_application       : num [1:559855] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_landfilled             : num [1:559855] 0 0.558 0 6.147 101.734 ...
##  $ tons_sewer                  : num [1:559855] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_refuse_discards        : num [1:559855] 0 5.32 0 58.53 156.23 ...
##  $ upstream_mtco2e_footprint   : num [1:559855] 35772.7 29.2 1517.9 162.8 1577.7 ...
##  $ downstream_mtco2e_footprint: num [1:559855] 1.76 4.3 307.61 15.18 282.28 ...
##  $ total_mtco2e_footprint      : num [1:559855] 35774.5 33.5 1825.5 178 1860 ...
##  $ gallons_water_footprint     : num [1:559855] 4.82e+09 4.47e+06 2.32e+08 2.49e+07 2.41
e+08 ...
##  $ meals_wasted                : num [1:559855] 14218563 230150 12002175 1277392 1237539
9 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   year = col_double(),
##   ..   state = col_character(),
##   ..   sector = col_character(),
##   ..   sub_sector = col_character(),
##   ..   sub_sector_category = col_character(),
##   ..   food_type = col_character(),
##   ..   food_category = col_character(),
##   ..   tons_surplus = col_character(),
##   ..   tons_supply = col_character(),
```

```
##   ..   us_dollars_surplus = col_character(),
##   ..   tons_waste = col_character(),
##   ..   tons_uneaten = col_character(),
##   ..   tons_donated = col_character(),
##   ..   tons_biomaterial_processing = col_character(),
##   ..   tons_animal_feed = col_character(),
##   ..   tons_anaerobically_digested = col_character(),
##   ..   tons_composted = col_character(),
##   ..   tons_not_harvested = col_character(),
##   ..   tons_incinerated = col_double(),
##   ..   tons_land_application = col_double(),
##   ..   tons_landfilled = col_double(),
##   ..   tons_sewer = col_double(),
##   ..   tons_refuse_discards = col_double(),
##   ..   upstream_mtco2e_footprint = col_double(),
##   ..   downstream_mtco2e_footprint = col_double(),
##   ..   total_mtco2e_footprint = col_double(),
##   ..   gallons_water_footprint = col_double(),
##   ..   meals_wasted = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
# Reusable function to Find Nulls
find_nulls <- function(col) {
    sum(is.na(col))
}
# Reusable function to Remove Dollar sign
remove_dollar_sign <- function(col) {
     as.numeric(stringr::str_remove(col,stringr::fixed("$")))
}

# Reusable function to Remove And sign
replace_and_sign <- function(col) {
     stringr::str_replace_all(col,stringr::fixed("&"),"and")
}

# Reusable function to Remove Special Characters
remove_splchars <- function(col) {
     stringr::str_replace_all(col,stringr::fixed(c("."="","-"=" ")))
}

# Reusable function to convert column values to Upper case
convert_toupper <- function(col) {
    toupper(col)
}

# Reusable function to Treat Numeric columns
treat_numeric <- function(col) {
    round(col/1000,2)
}

# Reusable function to Treat Outliers
treat_outliers <- function(df,col) {
    box_stat_col <- boxplot.stats(col)
    iqr_val <- box_stat_col$stats[4] - box_stat_col$stats[2]
    lower_whisker <- box_stat_col$stats[2] - 1.5 * iqr_val
    upper_whisker <- box_stat_col$stats[4] + 1.5 * iqr_val
    # Excluding the outliers and returns the Dataframe
    df <- df %>% mutate(is_outlier=(col> upper_whisker |   col < lower_whisker)) %>%  filt
er(!is_outlier) %>% select(-is_outlier)
    return(df)
}
```

```r
# Reusable function to map State name to Region
assign_regions <- function(state)
{
    ifelse(state %in% c("MAINE","VERMONT","MASSACHUSETTS","RHODE ISLAND","CONNECTICUT","NE
W HAMPSHIRE"),"NEW_ENGLAND"
           ,ifelse(state %in% c("NEW YORK","PENNSYLVANIA","NEW JERSEY","DELAWARE","MARYLAN
D"),"MID_ATLANTIC",
                   ifelse(state %in% c("ARKANSAS","LOUISIANA","MISSISSIPPI","ALABAMA","GEO
RGIA","FLORIDA","TENNESSEE","KENTUCKY","VIRGINIA","WEST VIRGINIA","NORTH CAROLINA","SOUTH
CAROLINA"),"SOUTH_EAST",
                          ifelse(state %in% c("NORTH DAKOTA","SOUTH DAKOTA","NEBRASKA","KA
NSAS","MISSOURI","IOWA","MINNESOTA","WISCONSIN","ILLINOIS","MICHIGAN","INDIANA","OHIO"),"M
ID_WEST",
                                 ifelse(state %in% c("NEVADA","UTAH","COLORADO","WYOMIN
G","IDAHO","MONTANA"),"ROCKY_MOUNTAIN_STATES",
                                        ifelse(state %in% c("WASHINGTON","CALIFORNIA","ORE
GON","ALASKA","HAWAII"),"PACIFIC_COASTAL",
                                               ifelse(state %in% c("ARIZONA","NEW MEXIC
O","OKLAHOMA","TEXAS"),"SOUTH_WEST",
                                                      ifelse(state %in% c("DISTRICT OF COL
UMBIA"),"DISTRICT OF COLUMBIA","Not a Valid US State"
                                                      )
                                               )
                                        )
                                 )
                          )
                   )


           )


    )

}
```

```r
# Reading the data from US Poverty Rate file
us_poverty_orig <- read_excel("US_Poverty_Rates.xlsx")
us_poverty_df <- us_poverty_orig
colnames(us_poverty_df) <- us_poverty_df[1,]
# Selecting only the required columns
us_poverty_df <- us_poverty_df[3:53,1:4]
us_poverty_df["State"] <- lapply(us_poverty_df["State"], convert_toupper)
us_poverty_df %<>% rename(state=State)
kbl(head(us_poverty_df),caption = "US Poverty Rates", booktabs = T) %>% kable_styling(late
x_options = c("striped", "hold_position"))
```

US Poverty Rates

state	2022	2021	2020
ALABAMA	13.6	15.9	14.9

state	2022	2021	2020
ALASKA	9.6	11.4	13.4
ARIZONA	12.7	12.6	10.9
ARKANSAS	16.6	16.8	14.1
CALIFORNIA	11.2	11.9	11.1
COLORADO	8.1	7.9	9.5

```
# Reading the data from US Homelessness file
us_homeless_orig <- read_excel("Homeless_Populations_by_State.xlsx")
us_homeless_2022_df <- us_homeless_orig
colnames(us_homeless_2022_df) <- us_homeless_2022_df[1,]
# Renaming columns and selecting the required columns
us_homeless_2022_df <- us_homeless_2022_df[2:56,1:2] %>% rename(homelessness_2022=`Total Y
ear-Round Beds (ES, TH, SH)`)
us_homeless_2022_df %<>% rename(Code=State)
```

```
# Reading the data from state abbreviation file
state_abbr_df <- read.csv("state_abbr.csv")
state_df <- state_abbr_df %>% select(Code,state)
us_homeless_2022_df1 <- us_homeless_2022_df %>% inner_join(state_df, by="Code") %>% select
(state,homelessness_2022)
kbl(head(us_homeless_2022_df1),caption = "US Homelessness Dataframe", booktabs = T) %>% ka
ble_styling(latex_options = c("striped", "hold_position"))
```

US Homelessness Dataframe

state	homelessness_2022
ALASKA	3088
ALABAMA	2714
ARKANSAS	2318
ARIZONA	6162
CALIFORNIA	68607
COLORADO	9294

```
# Reading the data from US Population file
us_pop_orig <-  read_excel("US_Population.xlsx")
us_pop_df <- us_pop_orig
colnames(us_pop_df) <- us_pop_df[1,]
# Selecting only the required Rows from the Dataframe
us_pop_df <- us_pop_df[7:57,]
us_pop_df["Geographic Area"] <- lapply(us_pop_df["Geographic Area"], convert_toupper)
us_pop_df %<>%  rename(state=`Geographic Area`) %>% select(state,`2020`:`2023`)
kbl(head(us_pop_df),caption = "US Population Dataframe", booktabs = T) %>% kable_styling(l
atex_options = c("striped", "hold_position"))
```

US Population Dataframe

state	2020	2021	2022	2023
ALABAMA	5031864	5050380	5073903	5108468
ALASKA	732964	734923	733276	733406
ARIZONA	7186683	7272487	7365684	7431344
ARKANSAS	3014348	3028443	3046404	3067732
CALIFORNIA	39503200	39145060	39040616	38965193
COLORADO	5785219	5811596	5841039	5877610

Data Wrangling

```
# Finding Nulls in the Food Surplus dataframe
apply(food_surplus_dtl_df, 2, find_nulls)
```

```
##                       year                        state
##                          0                            0
##                     sector                   sub_sector
##                          0                       129960
##        sub_sector_category                    food_type
##                     248607                            0
##              food_category                 tons_surplus
##                          0                        18525
##                tons_supply           us_dollars_surplus
##                      18104                        18104
##                 tons_waste                 tons_uneaten
##                      18104                        18104
##               tons_donated tons_biomaterial_processing
##                      18104                        18104
##           tons_animal_feed tons_anaerobically_digested
##                      18104                        18104
##             tons_composted           tons_not_harvested
##                      18104                        18104
##           tons_incinerated        tons_land_application
##                      18104                        18104
##            tons_landfilled                   tons_sewer
##                      18104                        18104
##       tons_refuse_discards   upstream_mtco2e_footprint
##                      18104                        18104
## downstream_mtco2e_footprint      total_mtco2e_footprint
##                      18104                        18104
##     gallons_water_footprint                 meals_wasted
##                      18104                        18104
```

```
# Finding Nulls using vTreat Package

missing_val_stats <- food_surplus_dtl_df %>%
  gather(Features, value) %>%
  group_by(Features) %>%
  count(na = is.na(value)) %>%
  pivot_wider(names_from = na, values_from = n, values_fill = 0) %>%
  mutate(Percent_missing = (`TRUE`/sum(`TRUE`, `FALSE`))*100) %>%
  ungroup()
# Displaying the results
missing_val_stats %>%  gt()
```

Features	FALSE	TRUE	Percent_missing
downstream_mtco2e_footprint	541751	18104	3.233694
food_category	559855	0	0.000000
food_type	559855	0	0.000000
gallons_water_footprint	541751	18104	3.233694
meals_wasted	541751	18104	3.233694

Features	FALSE	TRUE	Percent_missing
sector	559855	0	0.000000
state	559855	0	0.000000
sub_sector	429895	129960	23.213153
sub_sector_category	311248	248607	44.405605
tons_anaerobically_digested	541751	18104	3.233694
tons_animal_feed	541751	18104	3.233694
tons_biomaterial_processing	541751	18104	3.233694
tons_composted	541751	18104	3.233694
tons_donated	541751	18104	3.233694
tons_incinerated	541751	18104	3.233694
tons_land_application	541751	18104	3.233694
tons_landfilled	541751	18104	3.233694
tons_not_harvested	541751	18104	3.233694
tons_refuse_discards	541751	18104	3.233694
tons_sewer	541751	18104	3.233694
tons_supply	541751	18104	3.233694
tons_surplus	541330	18525	3.308892
tons_uneaten	541751	18104	3.233694
tons_waste	541751	18104	3.233694
total_mtco2e_footprint	541751	18104	3.233694
upstream_mtco2e_footprint	541751	18104	3.233694
us_dollars_surplus	541751	18104	3.233694
year	559855	0	0.000000

```
# Examining the nulls by only extracting the rows with nulls
food_surplus_dtl_df %>% filter(is.na(meals_wasted)) %>% head(5)
```

y… <dbl>	state <chr>	sector <chr>	sub_sector <chr>	sub_sector_category <chr>	food_ty <chr>
2022	Alabama	Foodservice	Bars And Taverns	Bars And Taverns	Frozen
2022	Alabama	Foodservice	Full Service Restaurants	All Other	Frozen
2022	Alabama	Foodservice	Full Service Restaurants	Asian/Noodle	Frozen
2022	Alabama	Food service	Full Service Restaurants	Mexican	Frozen
2022	Alabama	Food service	Full Service Restaurants	Steak	Frozen

5 rows | 1-7 of 28 columns

```
# Excluding the rows with nulls as data was missing for primary columns. In other cases, t
hey were imputed with 0 or NA
food_surplus_dtl_df1 <- food_surplus_dtl_df %>% filter(!is.na(meals_wasted)) %>% mutate(su
b_sector = ifelse(is.na(sub_sector), "NA", sub_sector), sub_sector_category = ifelse(is.na
(sub_sector_category), "NA", sub_sector_category) , tons_surplus = ifelse(is.na(tons_surpl
us), 0, tons_surplus))
food_surplus_dtl_df1 %>% head(5)
```

y… <dbl>	state <chr>	sector <chr>	sub_sector <chr>	sub_sector_category <chr>	food_type <chr>	food_category <chr>
2022	Alabama	Farm	Not Applicable	NA	Dry Goods	Nuts And Seeds
2022	Alabama	Farm	Not Applicable	NA	Produce	Blueberries
2022	Alabama	Farm	Not Applicable	NA	Produce	Cucumbers
2022	Alabama	Farm	Not Applicable	NA	Produce	Peaches
2022	Alabama	Farm	Not Applicable	NA	Produce	Potatoes

5 rows | 1-8 of 28 columns

```
# Examining the Nulls in Food  Surplus Dataframe and everything looks clean
apply(food_surplus_dtl_df1, 2, find_nulls)
```

```
##                        year                        state 
##                           0                            0 
##                      sector                   sub_sector 
##                           0                            0 
##         sub_sector_category                    food_type 
##                           0                            0 
##               food_category                 tons_surplus 
##                           0                            0 
##                 tons_supply            us_dollars_surplus 
##                           0                            0 
##                  tons_waste                 tons_uneaten 
##                           0                            0 
##                tons_donated tons_biomaterial_processing 
##                           0                            0 
##            tons_animal_feed tons_anaerobically_digested 
##                           0                            0 
##              tons_composted           tons_not_harvested 
##                           0                            0 
##            tons_incinerated        tons_land_application 
##                           0                            0 
##             tons_landfilled                   tons_sewer 
##                           0                            0 
##         tons_refuse_discards   upstream_mtco2e_footprint 
##                           0                            0 
## downstream_mtco2e_footprint       total_mtco2e_footprint 
##                           0                            0 
##     gallons_water_footprint                 meals_wasted 
##                           0                            0
```

```
# Extracting the column names with the word tons
tons_colnames <- food_surplus_dtl_df1 %>% select(contains("tons")) %>%  colnames
# Removing Dollar sign by applying the function
food_surplus_dtl_df1[tons_colnames] <- lapply(food_surplus_dtl_df1[tons_colnames], remove_
dollar_sign)
food_surplus_dtl_df1["us_dollars_surplus"] <- lapply(food_surplus_dtl_df1["us_dollars_surp
lus"], remove_dollar_sign)
# Extracting the character columns
char_colnames <- food_surplus_dtl_df1 %>% select(is.character) %>%  colnames
# Removing And Sign from the data
food_surplus_dtl_df1[char_colnames] <- lapply(food_surplus_dtl_df1[char_colnames], replace
_and_sign)
# Removing special characters
food_surplus_dtl_df1[char_colnames] <- lapply(food_surplus_dtl_df1[char_colnames], remove_
splchars)
# Converting the data to Upper case
food_surplus_dtl_df1[char_colnames] <- lapply(food_surplus_dtl_df1[char_colnames], convert
_toupper)
food_surplus_dtl_df1$year <- as.integer(food_surplus_dtl_df1$year)
dbl_colnames <- food_surplus_dtl_df1 %>% select(is.double) %>%  colnames
# Treating Numeric data
food_surplus_dtl_df1[dbl_colnames] <- lapply(food_surplus_dtl_df1[dbl_colnames], treat_num
eric)
```

```r
# Validating Nulls in the Summary Dataframe
apply(food_surplus_summ_df, 2, find_nulls)
```

```
##                             year                            state
##                                0                                0
##                           sector                       sub_sector
##                                0                                0
##              sub_sector_category                        food_type
##                                0                                0
##                     tons_surplus                      tons_supply
##                                0                                0
##               us_dollars_surplus                       tons_waste
##                                0                                0
##                     tons_uneaten              tons_inedible_parts
##                                0                                0
## tons_not_fit_for_human_consumption                     tons_donated
##                                0                                0
##        tons_biomaterial_processing                 tons_animal_feed
##                                0                                0
##        tons_anaerobically_digested                   tons_composted
##                                0                                0
##               tons_not_harvested                 tons_incinerated
##                                0                                0
##              tons_land_application                  tons_landfilled
##                                0                                0
##                       tons_sewer             tons_refuse_discards
##                                0                                0
##         upstream_mtco2e_footprint      downstream_mtco2e_footprint
##                                0                                0
##            total_mtco2e_footprint          gallons_water_footprint
##                                0                                0
##                     meals_wasted
##                                0
```

```r
# Extracting the character columns in Summary Dataframe
char_colnames2 <- food_surplus_summ_df %>% select(is.character) %>%  colnames
# Remove the And Sign
food_surplus_summ_df[char_colnames2] <- lapply(food_surplus_summ_df[char_colnames2], repla
ce_and_sign)
# Removing special characters from Summary Dataframe
food_surplus_summ_df[char_colnames2] <- lapply(food_surplus_summ_df[char_colnames2], remov
e_splchars)
# Convering column values to Upper case
food_surplus_summ_df[char_colnames2] <- lapply(food_surplus_summ_df[char_colnames2], conve
rt_toupper)
dbl_colnames2 <- food_surplus_summ_df %>% select(is.double) %>%  colnames
food_surplus_summ_df[dbl_colnames2] <- lapply(food_surplus_summ_df[dbl_colnames2], treat_n
umeric)
```

```r
kbl(head(food_surplus_summ_df),caption = "Food Surplus Summary Dataframe", booktabs = T)
%>% kable_styling(latex_options = c("striped", "hold_position"))
```

Food Surplus Summary Dataframe

year	state	sector	sub_sector	sub_sector_category	food_type	tons_surplus
2.02	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	DRY GOODS	8.53
2.02	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	55.13
2.02	ALABAMA	FOODSERVICE	BARS AND TAVERNS	BARS AND TAVERNS	BREADS AND BAKERY	0.00
2.02	ALABAMA	FOODSERVICE	BARS AND TAVERNS	BARS AND TAVERNS	DAIRY AND EGGS	0.00
2.02	ALABAMA	FOODSERVICE	BARS AND TAVERNS	BARS AND TAVERNS	DRY GOODS	0.00
2.02	ALABAMA	FOODSERVICE	BARS AND TAVERNS	BARS AND TAVERNS	FRESH MEAT AND SEAFOOD	0.01

```r
# Finding unique values in sector variable in Summary dataframe
food_surplus_summ_df %>% count(sector)
```

sector <chr>	n <int>
FARM	845
FOODSERVICE	171496
MANUFACTURING	5161
RESIDENTIAL	5200
RETAIL	5200

5 rows

```
# Creating a variable that contains the list of sector values that contains SERVICE
FS_categ <- unlist(food_surplus_dtl_df1[str_detect(food_surplus_dtl_df1$sector,"SERVIC
E"),"sector"] %>% unique())
# Creating a variable that contains the list of sector values that contains MANUFACT
MF_categ <- unlist(food_surplus_dtl_df1[str_detect(food_surplus_dtl_df1$sector,"MANUFAC
T"),"sector"] %>% unique())
# Creating a variable that contains the list of sector values that contains RESIDEN
RS_categ <- unlist(food_surplus_dtl_df1[str_detect(food_surplus_dtl_df1$sector,"RESIDE
N"),"sector"] %>% unique())
# Creating a variable that contains the list of sector values that contains BEVERAGE
bev_categ <- unlist(food_surplus_dtl_df1[str_detect(food_surplus_dtl_df1$food_type,"BEVERA
GE"),"food_type"] %>% unique())
NA_categ <- c("NOT APPLICABLE","NA")
```

```
#  Using fct_collapse to standardize the values
food_surplus_dtl_df1 <- food_surplus_dtl_df1 %>%
  mutate(sector = fct_collapse(sector,
                              FOODSERVICE = FS_categ,
                              MANUFACTURING = MF_categ,
                              RESIDENTIAL = RS_categ
                              ))
# Finding unique values in sector variable in Detail dataframe
 food_surplus_dtl_df1   %>% count(sector)
```

sector <fct>	n <int>
FARM	5278
FOODSERVICE	155058
MANUFACTURING	5161
RESIDENTIAL	188709
RETAIL	187545

5 rows

```
# Finding unique values in sub sector variable in Detail dataframe
food_surplus_dtl_df1 %>% count(sub_sector,sort=TRUE)
```

sub_sector <chr>	n <int>
NOT APPLICABLE	256468
NA	130225
FULL SERVICE RESTAURANTS	41498
LIMITED SERVICE RESTAURANTS	39572
HEALTHCARE	15574
BUSINESS AND INDUSTRY	5200

sub_sector <chr>	n <int>
MILITARY	5200
OTHER	5200
RECREATION	5200
CATERERS	5199

1-10 of 17 rows Previous **1** 2 Next

```
# Treating the Not applicable values
food_surplus_dtl_df1 <- food_surplus_dtl_df1 %>%
  mutate(sub_sector = fct_collapse(sub_sector,
                              `NOT APPLICABLE` = NA_categ
                              ))
# Finding unique values in sector variable in Detail dataframe
food_surplus_dtl_df1 %>% count(sub_sector,sort=TRUE)
```

sub_sector <fct>	n <int>
NOT APPLICABLE	386693
FULL SERVICE RESTAURANTS	41498
LIMITED SERVICE RESTAURANTS	39572
HEALTHCARE	15574
BUSINESS AND INDUSTRY	5200
MILITARY	5200
OTHER	5200
RECREATION	5200
CATERERS	5199
COLLEGES AND UNIVERSITIES	5198

1-10 of 16 rows Previous **1** 2 Next

```
# Finding unique values in subsector variable in Summary dataframe
food_surplus_summ_df %>% count(sub_sector,sort=TRUE)
```

sub_sector <chr>	n <int>
FULL SERVICE RESTAURANTS	46800
LIMITED SERVICE RESTAURANTS	46800
NOT APPLICABLE	16406
HEALTHCARE	15600

sub_sector <chr>	n <int>
BUSINESS AND INDUSTRY	5200
CATERERS	5200
COLLEGES AND UNIVERSITIES	5200
CORRECTIONS	5200
K 12 EDUCATION	5200
LODGING	5200

1-10 of 16 rows Previous **1** 2 Next

```
# Finding unique values in sub sector category variable in Detail dataframe
food_surplus_dtl_df1 %>%  count(sub_sector_category,sort=TRUE)
```

sub_sector_category <chr>	n <int>
NA	248548
NOT APPLICABLE	138145
MEXICAN	8099
ASIAN/NOODLE	7826
ALL OTHER	7254
BURGER	5200
BUSINESS AND INDUSTRY	5200
FAMILY STYLE	5200
FROZEN DESSERT	5200
MILITARY	5200

1-10 of 32 rows Previous **1** 2 3 4 Next

```
# Treating Not applicable values in the sub sector category values
food_surplus_dtl_df1 <- food_surplus_dtl_df1 %>%
  mutate(sub_sector_category = fct_collapse(sub_sector_category,
                             `NOT APPLICABLE` = NA_categ
                             ))
# Finding unique values in sub sector category variable in Detail dataframe
food_surplus_dtl_df1 %>% count(sub_sector_category,sort=TRUE)
```

sub_sector_category <fct>	n <int>
NOT APPLICABLE	386693
MEXICAN	8099

sub_sector_category <fct>	n <int>
ASIAN/NOODLE	7826
ALL OTHER	7254
BURGER	5200
BUSINESS AND INDUSTRY	5200
FAMILY STYLE	5200
FROZEN DESSERT	5200
MILITARY	5200
OTHER	5200

1-10 of 31 rows Previous **1** 2 3 4 Next

```
# Finding unique values in sub sector category variable in summary dataframe
food_surplus_summ_df %>% count(sub_sector_category,sort=TRUE)
```

sub_sector_category <chr>	n <int>
NOT APPLICABLE	16406
ALL OTHER	10400
ASIAN/NOODLE	10400
MEXICAN	10400
BURGER	5200
BUSINESS AND INDUSTRY	5200
CATERERS	5200
CHICKEN	5200
COFFEE CAFE	5200
COLLEGES/UNIVERSITIES	5200

1-10 of 31 rows Previous **1** 2 3 4 Next

```
# Converting the character columns to factor
food_surplus_dtl_df1$food_type <- as.factor(food_surplus_dtl_df1$food_type)
food_surplus_dtl_df1$state <- as.factor(food_surplus_dtl_df1$state)
food_surplus_dtl_df1 <- food_surplus_dtl_df1 %>%
  mutate(food_type = fct_collapse(food_type,
                               `READY TO DRINK BEVERAGES` = bev_categ
                               ))
# Finding unique values in food type variable in Detail dataframe
food_surplus_dtl_df1 %>%  count(food_type,sort=TRUE)
```

food_type <fct>	n <int>
DRY GOODS	130766
PRODUCE	119701
FRESH MEAT AND SEAFOOD	61335
PREPARED FOODS	60530
FROZEN	57144
BREADS AND BAKERY	45032
DAIRY AND EGGS	38523
READY TO DRINK BEVERAGES	28720

8 rows

```
# Finding unique values in food type variable in Summary dataframe
food_surplus_summ_df %>%  count(food_type,sort=TRUE)
```

food_type <chr>	n <int>
PRODUCE	23998
DRY GOODS	23621
BREADS AND BAKERY	23387
DAIRY AND EGGS	23387
FRESH MEAT AND SEAFOOD	23387
FROZEN	23387
PREPARED FOODS	23387
READY TO DRINK BEVERAGES	23348

8 rows

```
# Examining the structure of the Food Surplus Detail Dataframe
str(food_surplus_dtl_df1)
```

```
## tibble [541,751 × 28] (S3: tbl_df/tbl/data.frame)
##  $ year                        : int [1:541751] 2022 2022 2022 2022 2022 2022 2022 2022
2022 2022 ...
##  $ state                       : Factor w/ 50 levels "ALABAMA","ALASKA",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ sector                      : Factor w/ 5 levels "FARM","FOODSERVICE",..: 1 1 1 1 1 1
1 1 2 2 ...
##  $ sub_sector                  : Factor w/ 16 levels "BARS AND TAVERNS",..: 12 12 12 12
12 12 12 12 1 1 ...
##  $ sub_sector_category         : Factor w/ 31 levels "ALL OTHER","ASIAN/NOODLE",..: 19 1
9 19 19 19 19 19 19 19 3 3 ...
##  $ food_type                   : Factor w/ 8 levels "READY TO DRINK BEVERAGES",..: 4 8 8
8 8 8 8 8 2 3 ...
##  $ food_category               : chr [1:541751] "NUTS AND SEEDS" "BLUEBERRIES" "CUCUMBER
S" "PEACHES" ...
##  $ tons_surplus                : num [1:541751] 8.53 0.14 7.2 0.78 7.55 ...
##  $ tons_supply                 : num [1:541751] 288.78 0.39 38.18 2.05 34.17 ...
##  $ us_dollars_surplus          : num [1:541751] 4443 495 3575 2140 3229 ...
##  $ tons_waste                  : num [1:541751] 8.53 0.12 7.2 0.61 6.93 ...
##  $ tons_uneaten                : num [1:541751] 8.53 0.14 7.2 0.77 7.43 ...
##  $ tons_donated                : num [1:541751] 0 0 0 0.01 0.12 0.01 0.01 0.28 0 0 ...
##  $ tons_biomaterial_processing: num [1:541751] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_animal_feed            : num [1:541751] 0 0.01 0 0.15 0.5 0.11 0.17 1.15 0 0 ...
##  $ tons_anaerobically_digested: num [1:541751] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_composted              : num [1:541751] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_not_harvested          : num [1:541751] 8.53 0.12 7.2 0.55 6.67 0.97 3.13 32.9 0
0 ...
##  $ tons_incinerated            : num [1:541751] 0 0 0 0 0 0 0 0.01 0 0 ...
##  $ tons_land_application       : num [1:541751] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_landfilled             : num [1:541751] 0 0 0 0.01 0.1 0 0.01 0.24 0 0 ...
##  $ tons_sewer                  : num [1:541751] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tons_refuse_discards        : num [1:541751] 0 0.01 0 0.06 0.16 0.04 0.07 0.36 0 0
...
##  $ upstream_mtco2e_footprint   : num [1:541751] 35.77 0.03 1.52 0.16 1.58 ...
##  $ downstream_mtco2e_footprint: num [1:541751] 0 0 0.31 0.02 0.28 0.04 0.12 1.4 0 0 ...
##  $ total_mtco2e_footprint      : num [1:541751] 35.77 0.03 1.83 0.18 1.86 ...
##  $ gallons_water_footprint     : num [1:541751] 4822024 4470 232228 24907 241387 ...
##  $ meals_wasted                : num [1:541751] 14219 230 12002 1277 12375 ...
```

```
# Creating dataframe for 2022 data
food_surplus_2022_df1 <- food_surplus_dtl_df1 %>% filter(year==2022)
kbl(head(food_surplus_2022_df1),caption = "Food Surplus 2022 Dataframe", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
```

Food Surplus 2022 Dataframe

year	state	sector	sub_sector	sub_sector_category	food_type	food_category	tons
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	DRY GOODS	NUTS AND SEEDS	

year	state	sector	sub_sector	sub_sector_category	food_type	food_category	tons
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	BLUEBERRIES	
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	CUCUMBERS	
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	PEACHES	
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	POTATOES	
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	SWEET CORN	

```
# Creating a Combined Dataframe from the Food Surplus, Population, Poverty and Homelessnes
s dataframe
join_df <- food_surplus_2022_df1 %>% inner_join(us_pop_df[c("state","2022")], by="state")
%>% rename(population_2022=`2022`) %>% inner_join(us_poverty_df[c("state","2022")], by="st
ate") %>% rename(poverty_2022=`2022`) %>% inner_join(us_homeless_2022_df1, by="state")
# Checking for Nulls in the combined dataframe
apply(join_df, 2, find_nulls)
```

```
##                      year                      state 
##                         0                          0 
##                    sector                 sub_sector 
##                         0                          0 
##       sub_sector_category                  food_type 
##                         0                          0 
##             food_category               tons_surplus 
##                         0                          0 
##               tons_supply         us_dollars_surplus 
##                         0                          0 
##                tons_waste               tons_uneaten 
##                         0                          0 
##              tons_donated tons_biomaterial_processing 
##                         0                          0 
##          tons_animal_feed tons_anaerobically_digested 
##                         0                          0 
##            tons_composted          tons_not_harvested 
##                         0                          0 
##          tons_incinerated       tons_land_application 
##                         0                          0 
##           tons_landfilled                 tons_sewer 
##                         0                          0 
##       tons_refuse_discards   upstream_mtco2e_footprint 
##                         0                          0 
## downstream_mtco2e_footprint      total_mtco2e_footprint 
##                         0                          0 
##     gallons_water_footprint                meals_wasted 
##                         0                          0 
##           population_2022               poverty_2022 
##                         0                          0 
##         homelessness_2022 
##                         0
```

```r
# Adding Region column to the join dataframe by calling the assign regions function
join_df <- join_df %>% mutate(region = assign_regions(join_df$state))
join_df$region <- as.factor(join_df$region)
# Treating outliers in the meals wasted column of join Datframe
join_df <- treat_outliers(join_df,join_df$meals_wasted)
join_df <- join_df[join_df$meals_wasted>10,]
join_df %>% dim
```

```
## [1] 30264    32
```

```r
kbl(head(join_df),caption = "Merged Dataframe", booktabs = T) %>% kable_styling(latex_opti
ons = c("striped", "hold_position"))
```

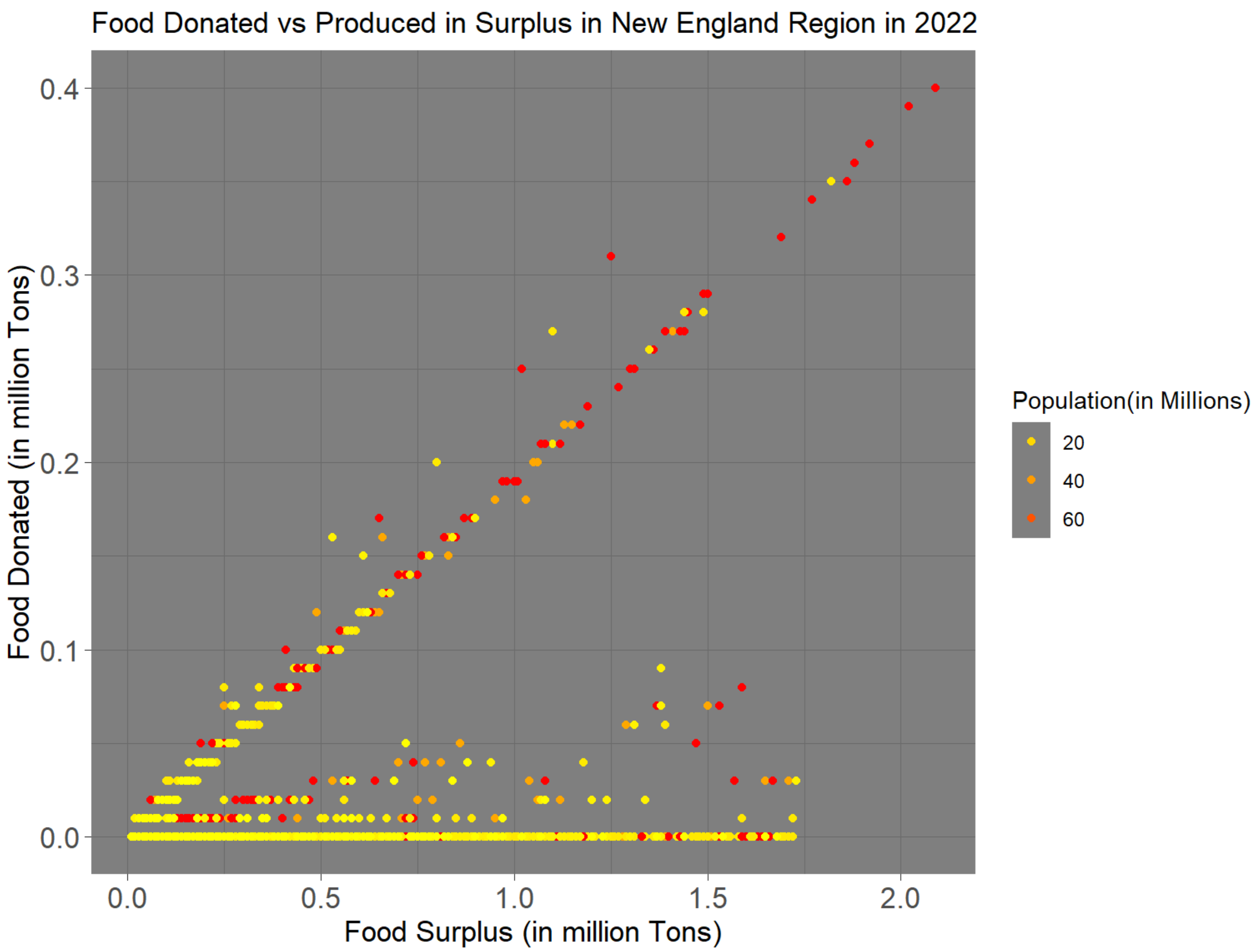
Merged Dataframe

year	state	sector	sub_sector	sub_sector_category	food_type	food_catego

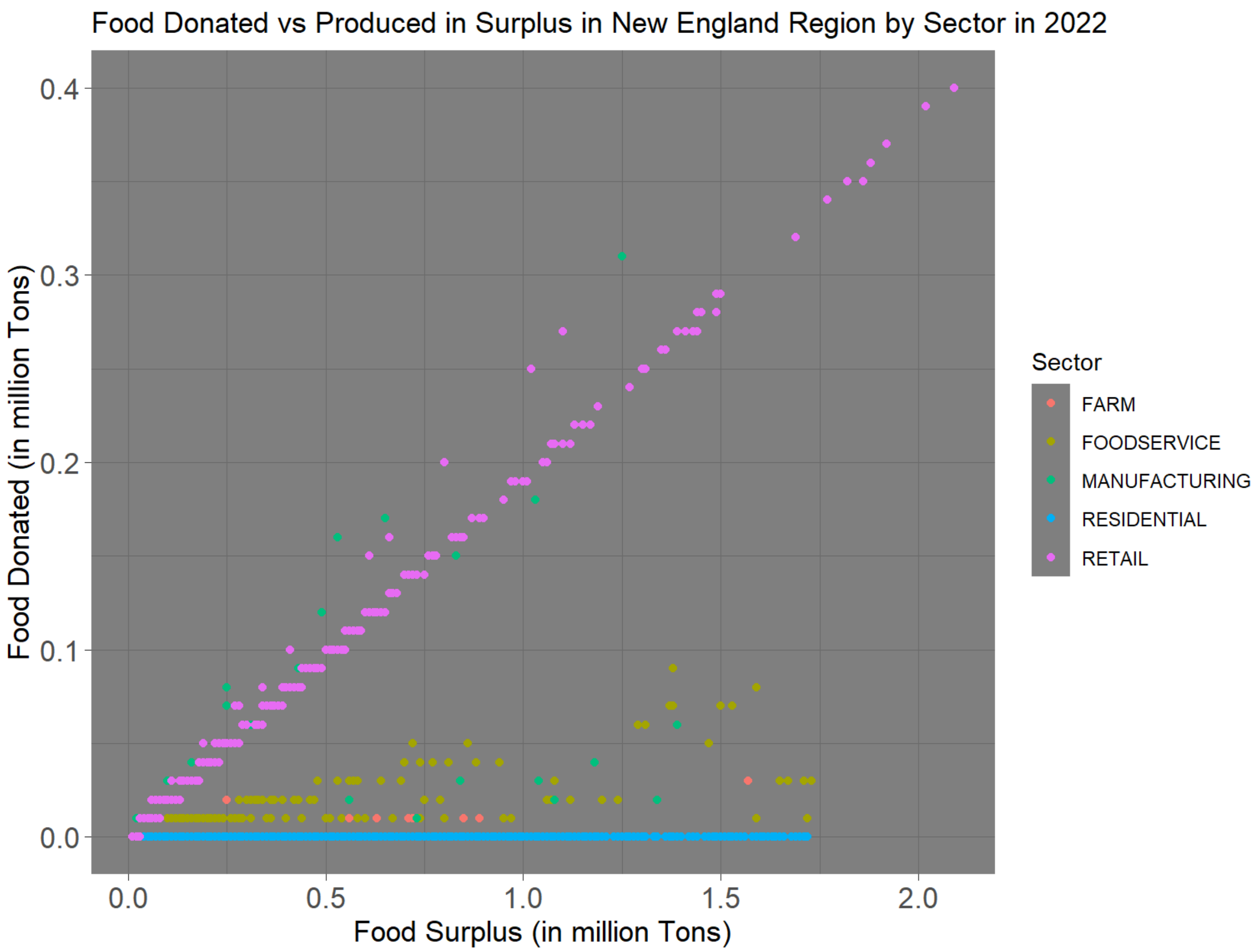
year	state	sector	sub_sector	sub_sector_category	food_type	food_catego
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	BLUEBERRI
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	PEACHES
2022	ALABAMA	FARM	NOT APPLICABLE	NOT APPLICABLE	PRODUCE	SWEET COI
2022	ALABAMA	FOODSERVICE	BARS AND TAVERNS	BARS AND TAVERNS	PREPARED FOODS	NOT APPLICABLI
2022	ALABAMA	FOODSERVICE	BUSINESS AND INDUSTRY	BUSINESS AND INDUSTRY	BREADS AND BAKERY	NOT APPLICABLI
2022	ALABAMA	FOODSERVICE	BUSINESS AND INDUSTRY	BUSINESS AND INDUSTRY	DAIRY AND EGGS	NOT APPLICABLI

Exploratory Data Analysis

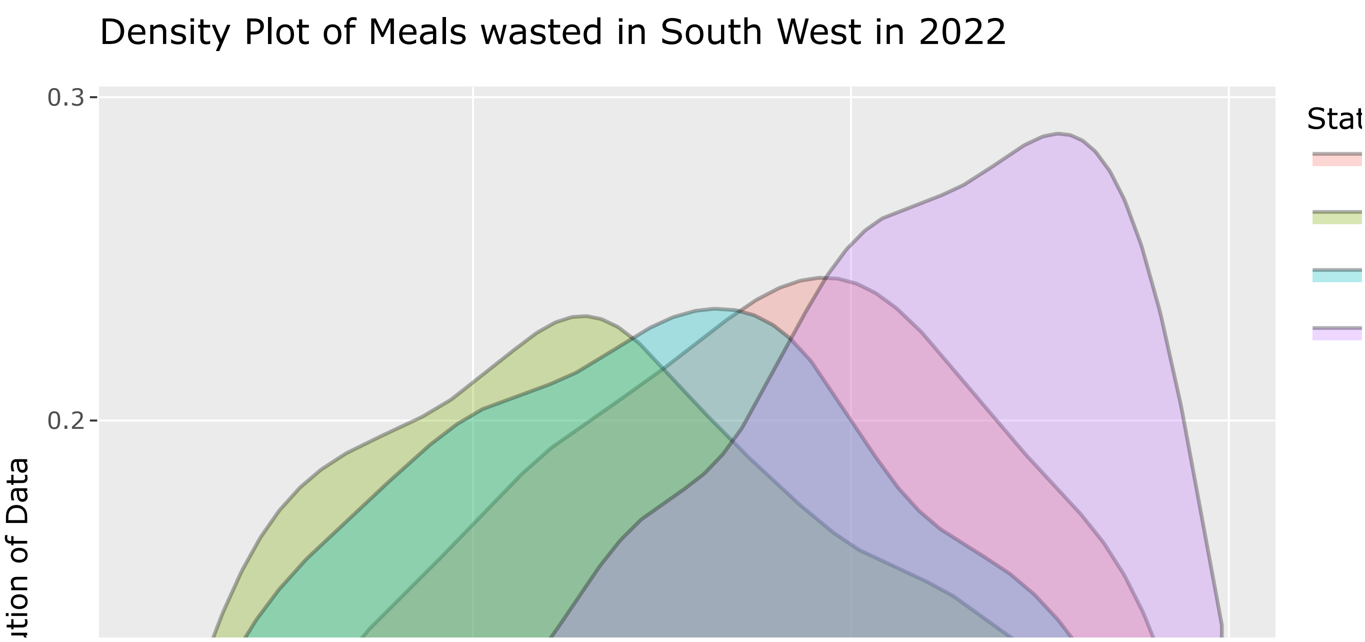
```
# Creating a Dataframe for New England Data
new_england_df1 <- join_df %>% filter(region=="NEW_ENGLAND")
# Plotting a scatter plot of Donated Food vs Food surplus based on the population
ggplot(new_england_df1, aes(x=tons_surplus,y=tons_donated,color=population_2022/100000)) +
  geom_point()+    scale_color_gradient(low = 'yellow', high = 'red')+
    labs(y="Food Donated (in million Tons)",
       x="Food Surplus (in million Tons)",
       title="Food Donated vs Produced in Surplus in New England Region in 2022")+
       theme_dark()+
  guides(color = guide_legend(title = "Population(in Millions)")) +
      theme(plot.title = element_text(size=13),axis.text.x= element_text(size=13),
                            axis.text.y= element_text(size=13), axis.title=element_text(si
ze=13))
```

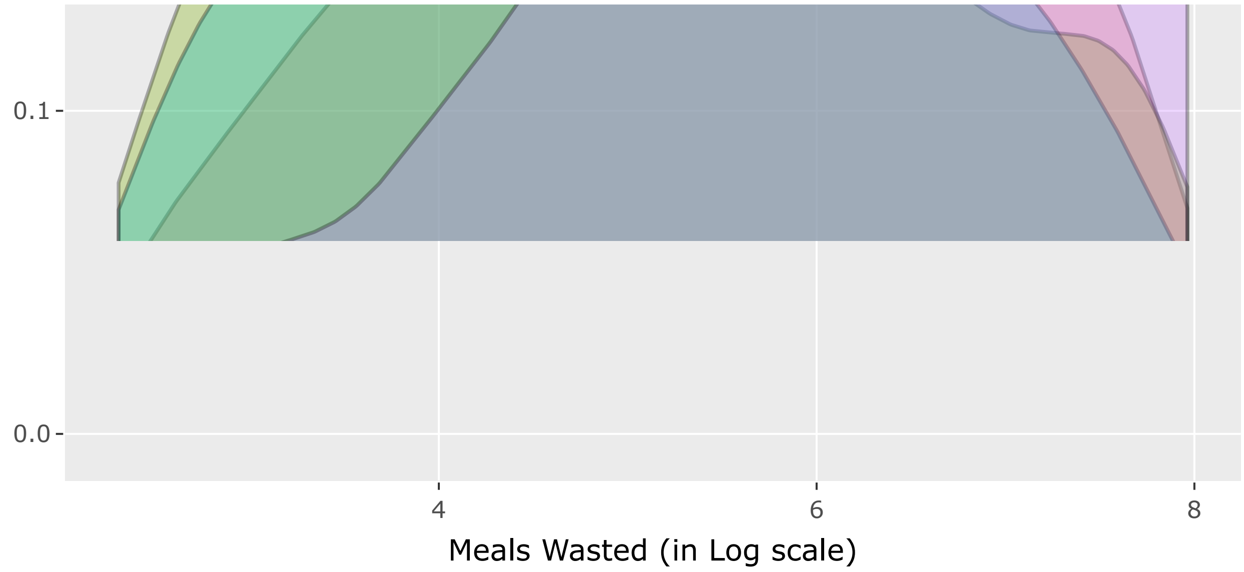



```
# Plotting a scatter plot of Donated Food vs Food surplus based on the sector
ggplot(new_england_df1, aes(x=tons_surplus,y=tons_donated,color=sector)) +
  geom_point()+    #scale_color_gradient(low = 'yellow', high = 'red')+
    labs(y="Food Donated (in million Tons)",
       x="Food Surplus (in million Tons)",
       title="Food Donated vs Produced in Surplus in New England Region by Sector in 202
2")+
       theme_dark()+
  guides(color = guide_legend(title = "Sector")) +
      theme(plot.title = element_text(size=13),axis.text.x= element_text(size=13),
                           axis.text.y= element_text(size=13), axis.title=element_text(si
ze=13))
```

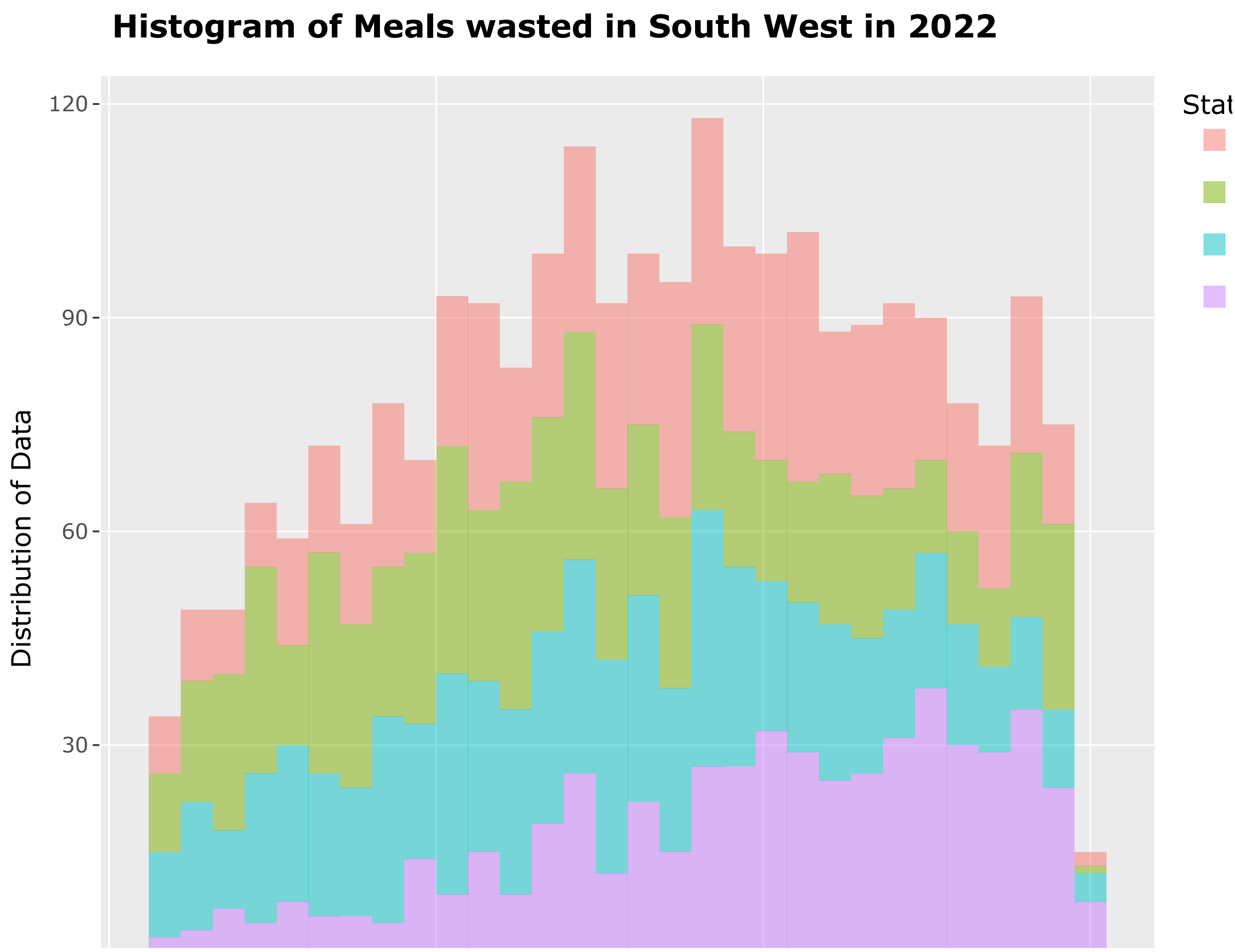


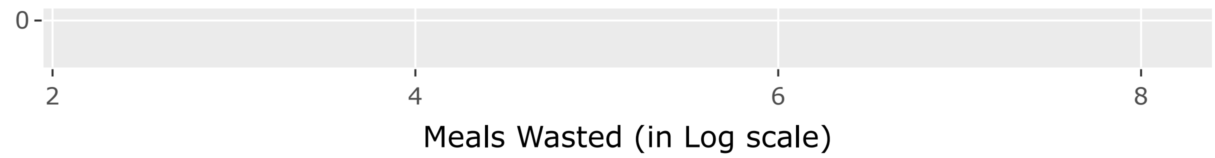

```
# Creating  dataframe for South West Region
density_df <- join_df %>% filter(region=="SOUTH_WEST")
# Plotting Density Plot
plot4 <- ggplot(density_df,aes(log(meals_wasted), fill=as.factor(state)))+geom_density(alp
ha=0.3)+labs(x="Meals Wasted (in Log scale)",y="Distribution of Data",title="Density Plot
of Meals wasted in South West in 2022")
p4 <- ggplotly(plot4, width=800, height=600)  %>% plotly::layout(legend = list(title=list
(text='State')))
p4
```



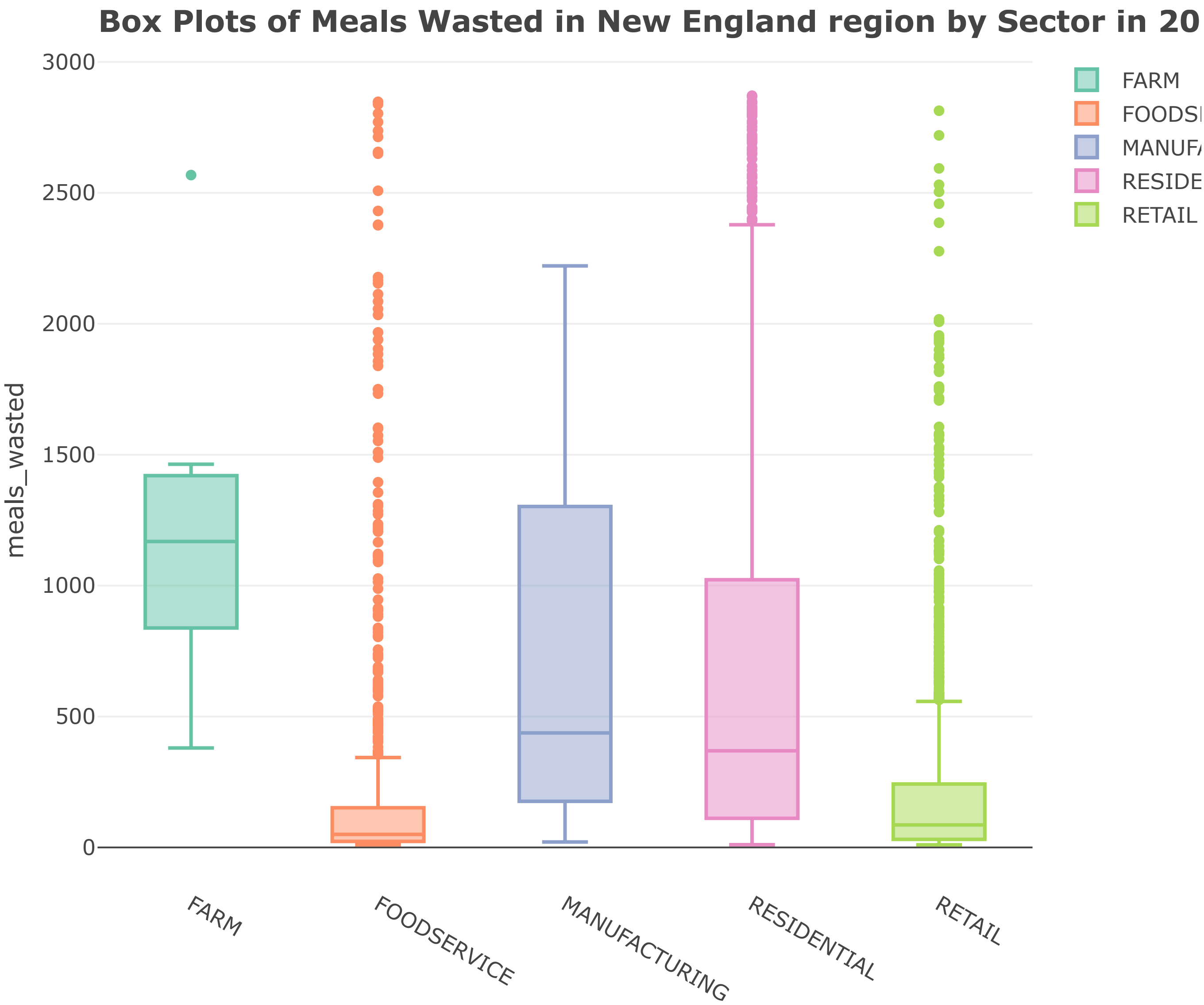



```
# Creating a Dataframe for Histogram
hist_df <- join_df %>% filter(region=="SOUTH_WEST")
# Plotting Histogram on Log scale of Meals wasted
hist_plot <- ggplot(hist_df,aes(log(meals_wasted), fill=as.factor(state)))+geom_histogram
(alpha=0.5)+labs(x="Meals Wasted (in Log scale)",y="Distribution of Data",title="<b> Histo
gram of Meals wasted in South West in 2022</b>")
p5 <- ggplotly(hist_plot, width=800, height=600)  %>% plotly::layout(legend = list(title=l
ist(text='State')))
p5
```

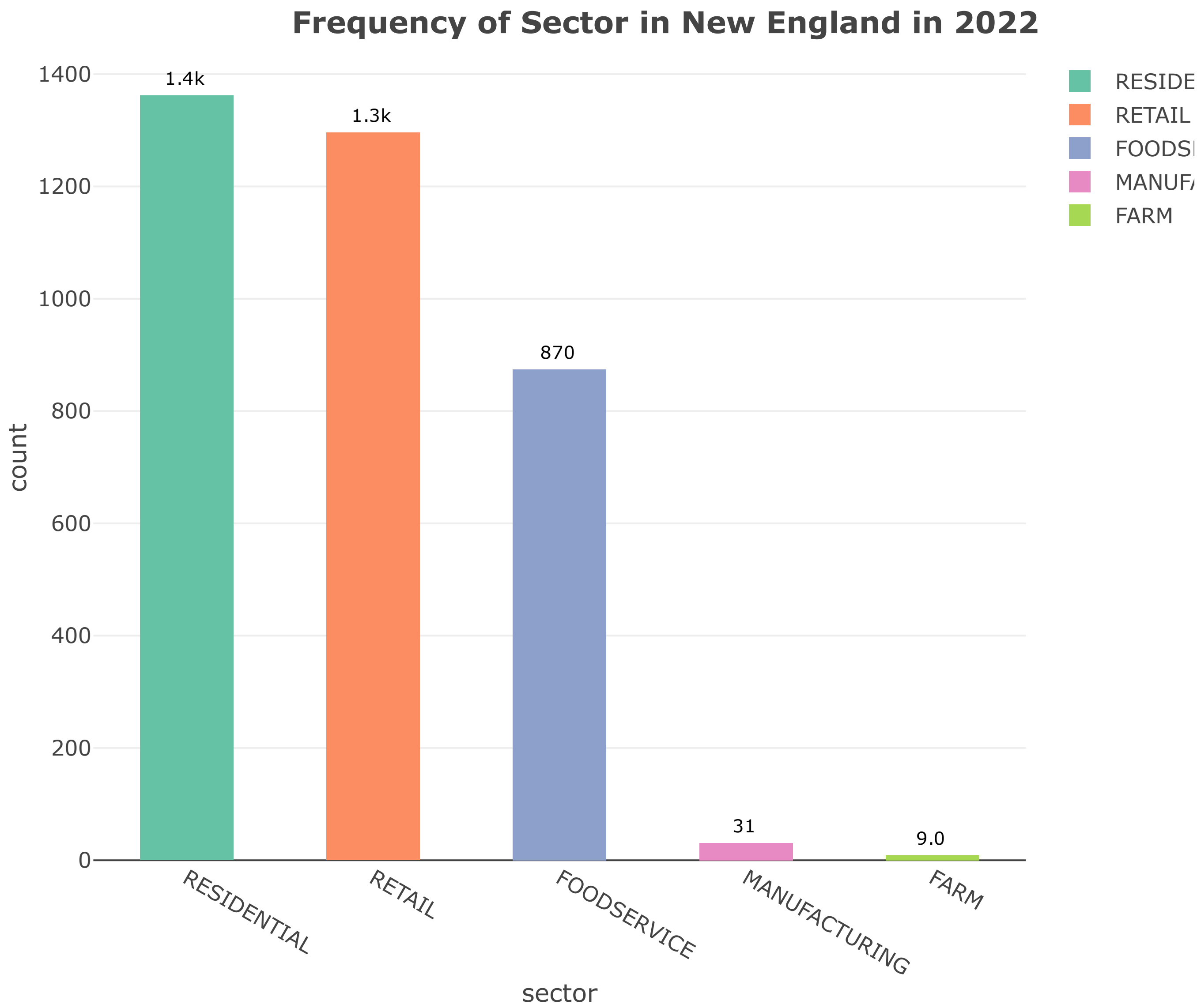





```
# Creating  Boxplot of Meals wasted based on the sector
boxplot_fig <- plot_ly(new_england_df1, y = ~meals_wasted, color = ~sector, type = "box")
boxplot_fig <-  boxplot_fig %>% plotly::layout(title = "<b>Box Plots of Meals Wasted in New England region by Sector in 2022</b>")
boxplot_fig
```



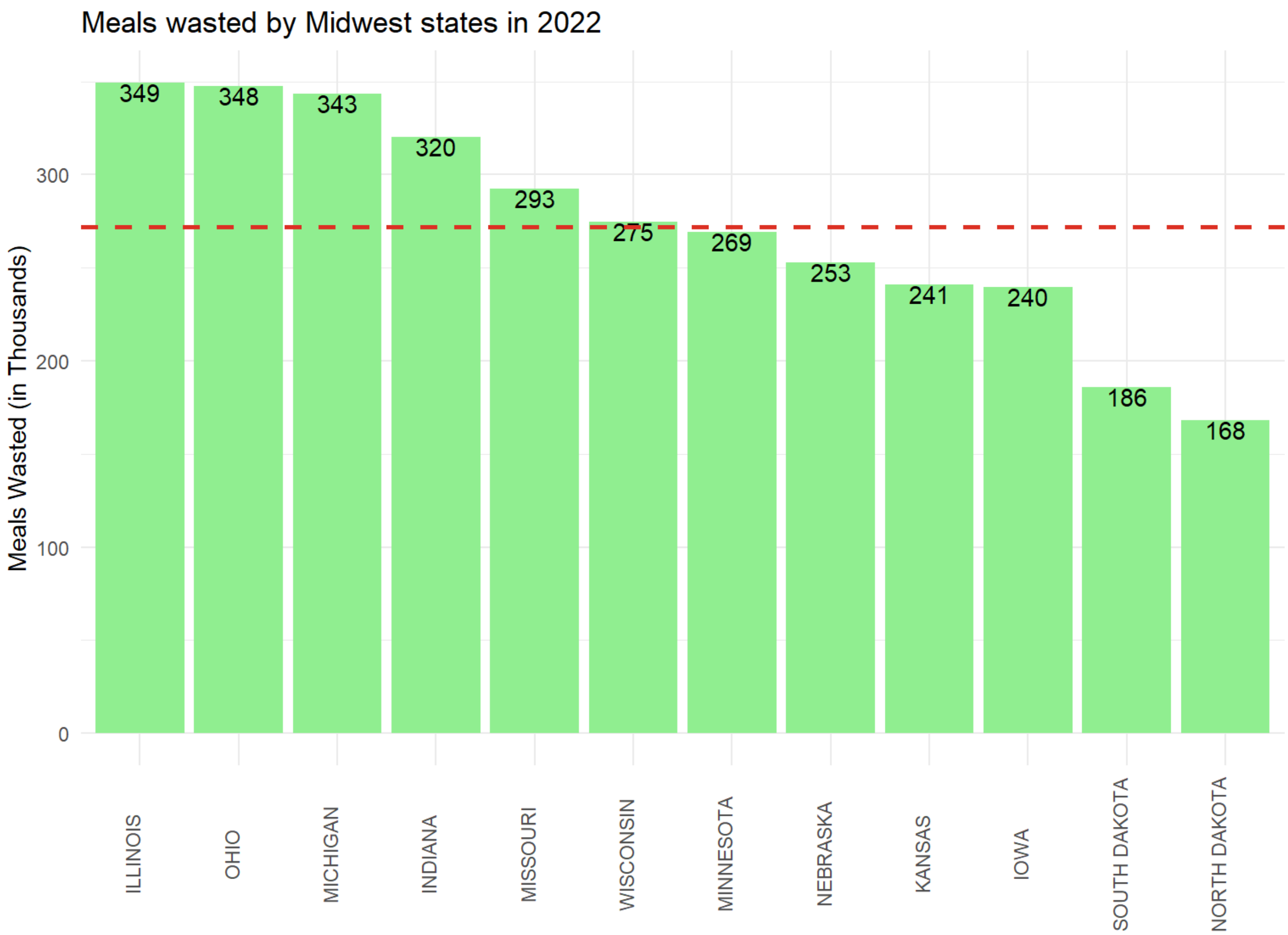
```
# Creating a subset of New England Dataframe
new_england_df2 <- new_england_df1 %>% count(sector,sort=TRUE) %>% mutate(count=n)
new_england_df2$sector <- factor(new_england_df2$sector , levels = unique(new_england_df2
$sector )[order(new_england_df2$count, decreasing = TRUE)])
# Plotting count plot of the sectors in New England Region
new_england_df2 %>%
    plot_ly(x=~sector,
          y=~count,
          color=~sector,
          text = ~paste("$",count),
          textfont = list(color = "black",  size = 10),
          ## below 3 lines for the bar label and hover text
          textposition = "outside",
          texttemplate = '%{y:.2s}'
          ) %>%
  add_bars(width=0.5) %>% plotly::layout(title="<b>Frequency of Sector in New England in 2
022</b>")
```



Research Questions

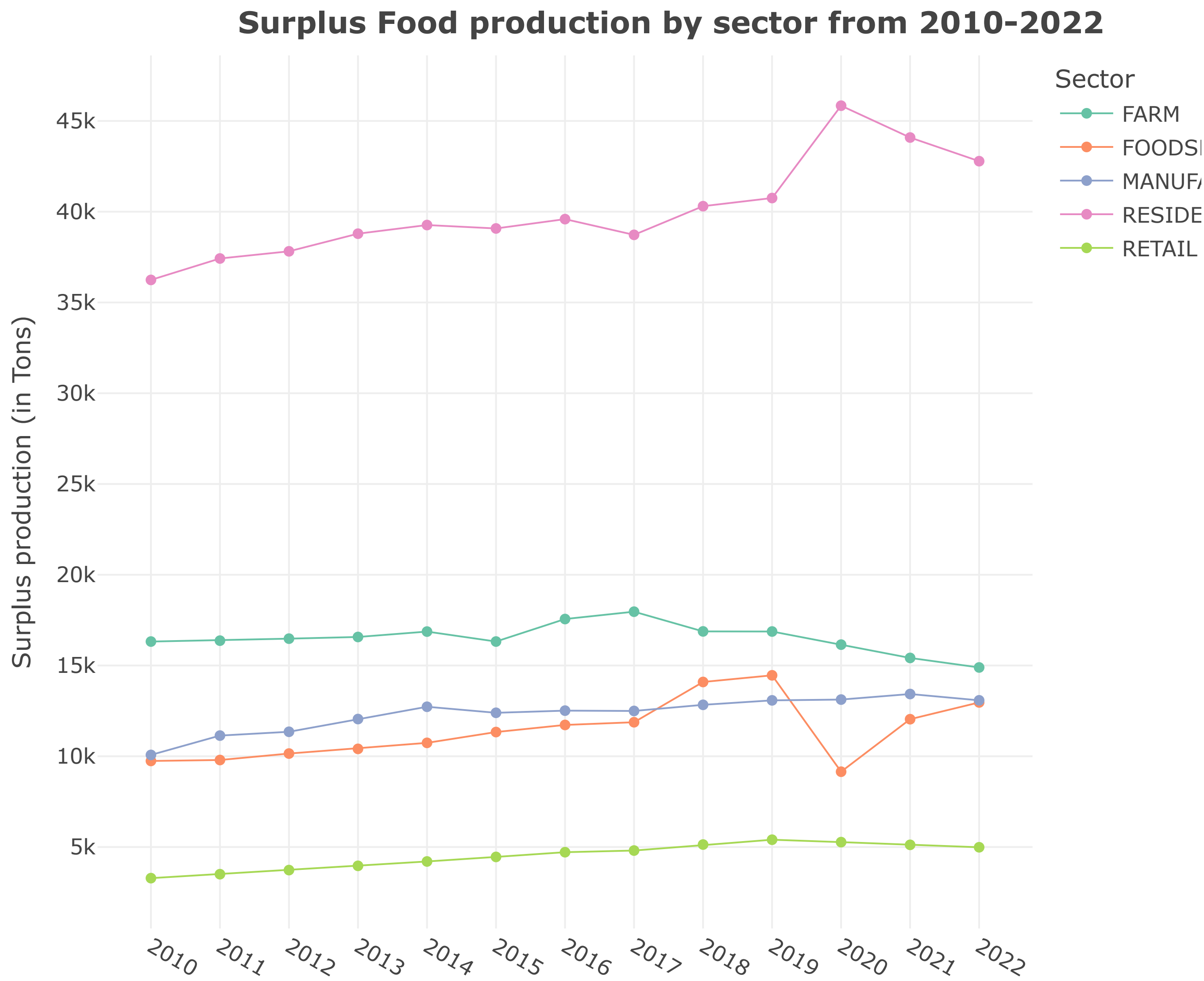
1. Which Mid-western state wasted the most food in 2022 and by how much?

```
# Creating a Dataframe for Mid west data
mid_west_df <- join_df %>%  filter(region == "MID_WEST") %>% group_by(state) %>% dplyr::su
mmarize(total_meals_wasted=sum(meals_wasted)) %>% arrange(desc(total_meals_wasted)) %>%  h
ead(15)
median_meals_wasted <- mid_west_df$total_meals_wasted %>% median()
# Plotting Bar plot with a Median Line
ggplot(data = mid_west_df,
       aes(x = reorder(state, -total_meals_wasted), y = total_meals_wasted/1000)) +
  geom_col( fill = '#90EE90') + geom_text(aes(label = round(total_meals_wasted/1000)), vju
st = 1)+
  geom_hline(yintercept = median_meals_wasted/1000, color = '#DC3023',
             linetype = 2, size = 1) +
  theme_minimal() +
  labs(title = 'Meals wasted by Midwest states in 2022',
       x = '',
       y = 'Meals Wasted (in Thousands)') +
  theme(axis.text.x = element_text(angle = 90, hjust = 0.5, vjust = 0))
```



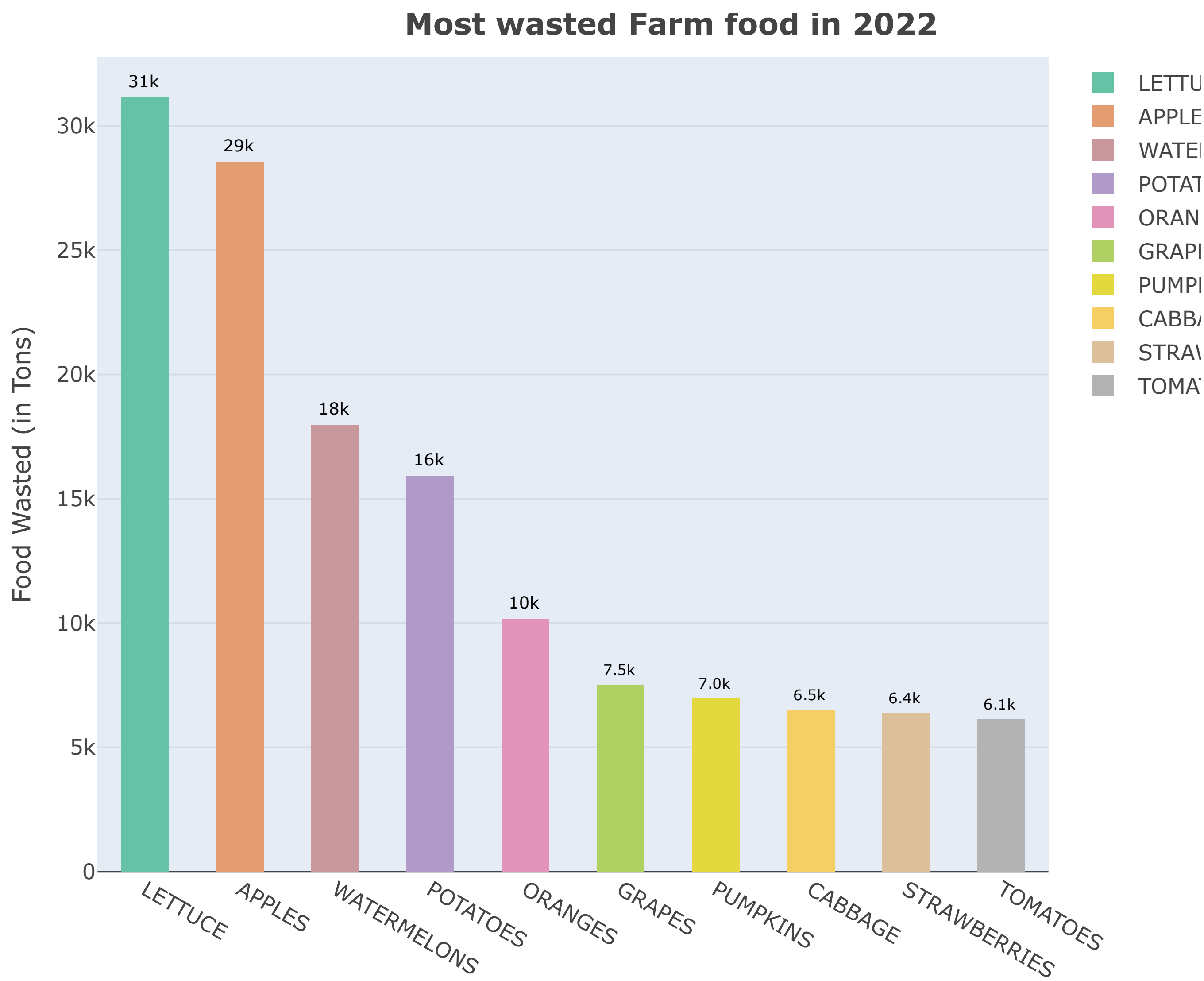
2. What was the trend of surplus food production in the U.S. over the years?

```
conflicted::conflicts_prefer(dplyr::summarize)
# Creating  a subset Dataframe with summary data
food_surplus_dtl_df2 <- food_surplus_dtl_df1 %>% mutate(region = assign_regions(food_surpl
us_dtl_df1$state))
food_surplus_df11 <- food_surplus_dtl_df2 %>% group_by(year,sector) %>% summarize(total_su
rplus=sum(tons_surplus))
food_surplus_df11$year <- as.factor(food_surplus_df11$year)

# Creating a LIne Plot of Surplus Production vs Years
fig1 <- plot_ly(data=food_surplus_df11, x = ~year, y = ~total_surplus, type = 'scatter', m
ode = 'markers',color=~sector, line = list(width = 1, dash = "solid",shape = "linear"))
%>%
  plotly::layout(title="<b>Surplus Food production by sector from 2010-2022</b>",
         xaxis = list(title=" "),
         yaxis = list(title = "Surplus production (in Tons)"),
         legend = list(title = list(text = "Sector")))
fig1
```



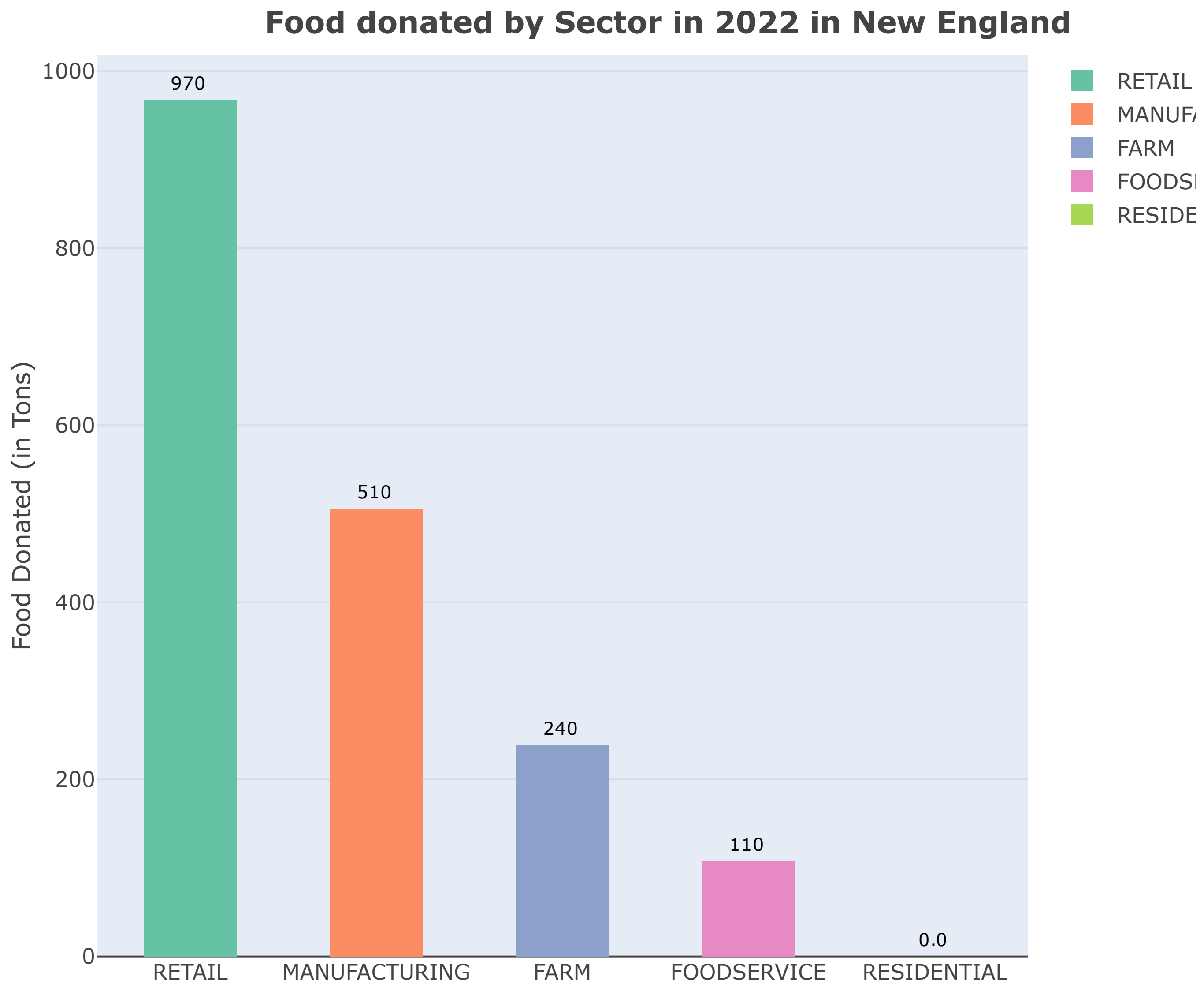
3. What was the most wasted Farm food in 2022?

```
# Creating a dataframe with only Farm Data
wasted_farm_df <- food_surplus_dtl_df2 %>% filter(sector=="FARM") %>% group_by(food_catego
ry) %>% summarize(total_tons_wasted=sum(tons_waste)) %>% arrange(desc(total_tons_wasted))
%>% head(10)
wasted_farm_df$food_category <- factor(wasted_farm_df$food_category , levels = unique(wast
ed_farm_df$food_category )[order(wasted_farm_df$total_tons_wasted, decreasing = TRUE)])
# Plotting Bar plot on the wasted Farm Data
wasted_farm_df %>%
    plot_ly(x=~food_category,
          y=~total_tons_wasted,
          color=~food_category,
          text = ~paste("$",total_tons_wasted),
          textfont = list(color = "black",  size = 10),
          ## below 3 lines for the bar label and hover text
          textposition = "outside",
          texttemplate = '%{y:.2s}'
          ) %>%
  plotly::layout(title = '<b>Most wasted Farm food in 2022</b>', plot_bgcolor = "#e5ecf6",
xaxis = list(title = ''),
         yaxis = list(title = 'Food Wasted (in Tons)')) %>%
  add_bars(width=0.5)
```



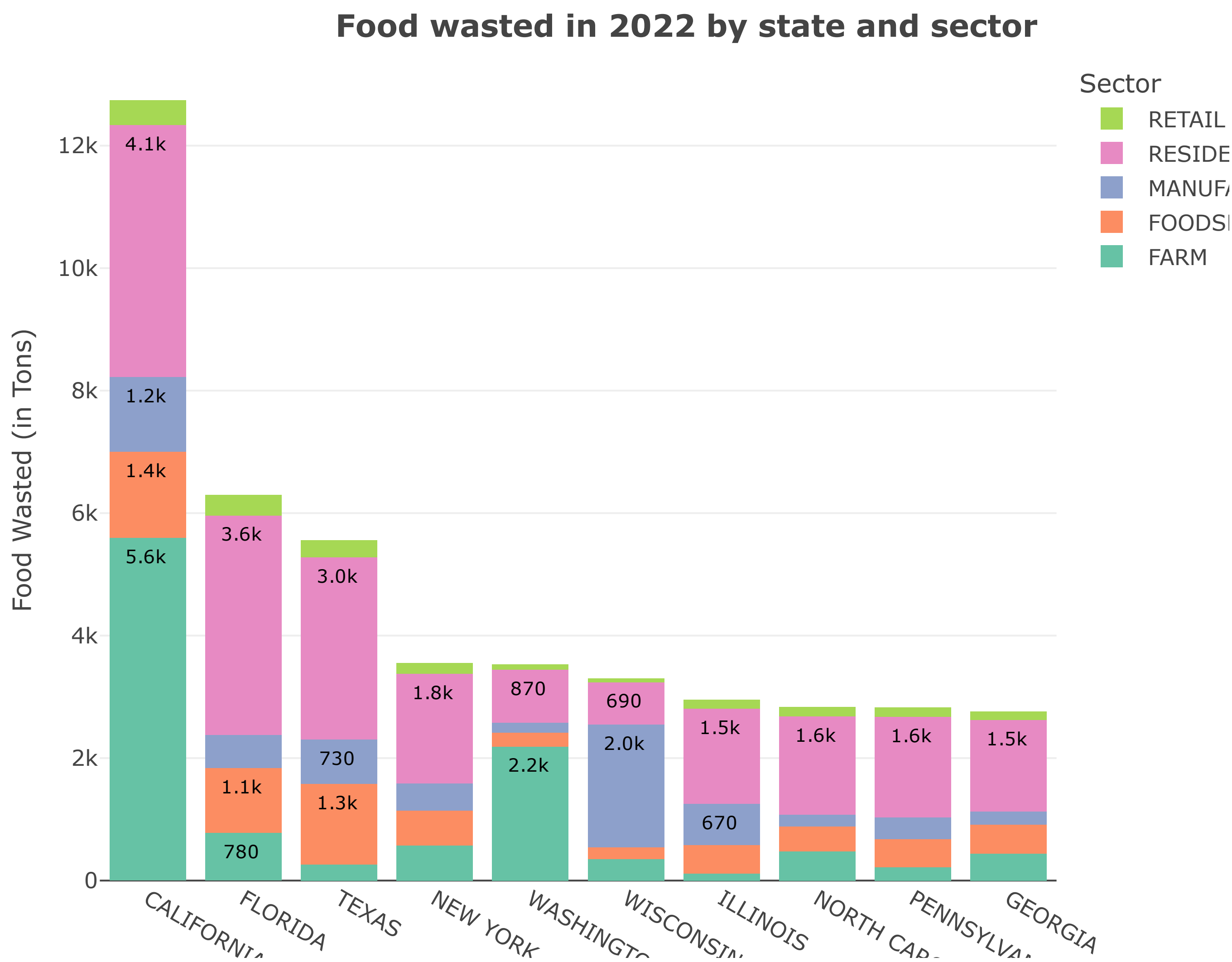
4. Which sector donated the most amount of food?

```
# Creating a Dataframe with summary of Most donated sector
most_donated_df1 <- food_surplus_dtl_df2 %>% filter(year==2022) %>% group_by(sector) %>% s
ummarize(total_tons_donated=sum(tons_donated)) %>% arrange(desc(total_tons_donated))
most_donated_df1$sector <- factor(most_donated_df1$sector , levels = unique(most_donated_d
f1$sector )[order(most_donated_df1$total_tons_donated, decreasing = TRUE)])
# Creating a Bar plot of Food donated by sector
most_donated_df1 %>%
    plot_ly(x=~sector,
          y=~total_tons_donated,
          color=~sector,
          text = ~paste("$",total_tons_donated),
          textfont = list(color = "black",  size = 10),
          ## below 3 lines for the bar label and hover text
          textposition = "outside",
          texttemplate = '%{y:.2s}'
          ) %>%
  plotly::layout(title = '<b>Food donated by Sector in 2022 in New England</b>', plot_bgco
lor = "#e5ecf6", xaxis = list(title = ''),
         yaxis = list(title = 'Food Donated (in Tons)')) %>%
  add_bars(width=0.5)
```



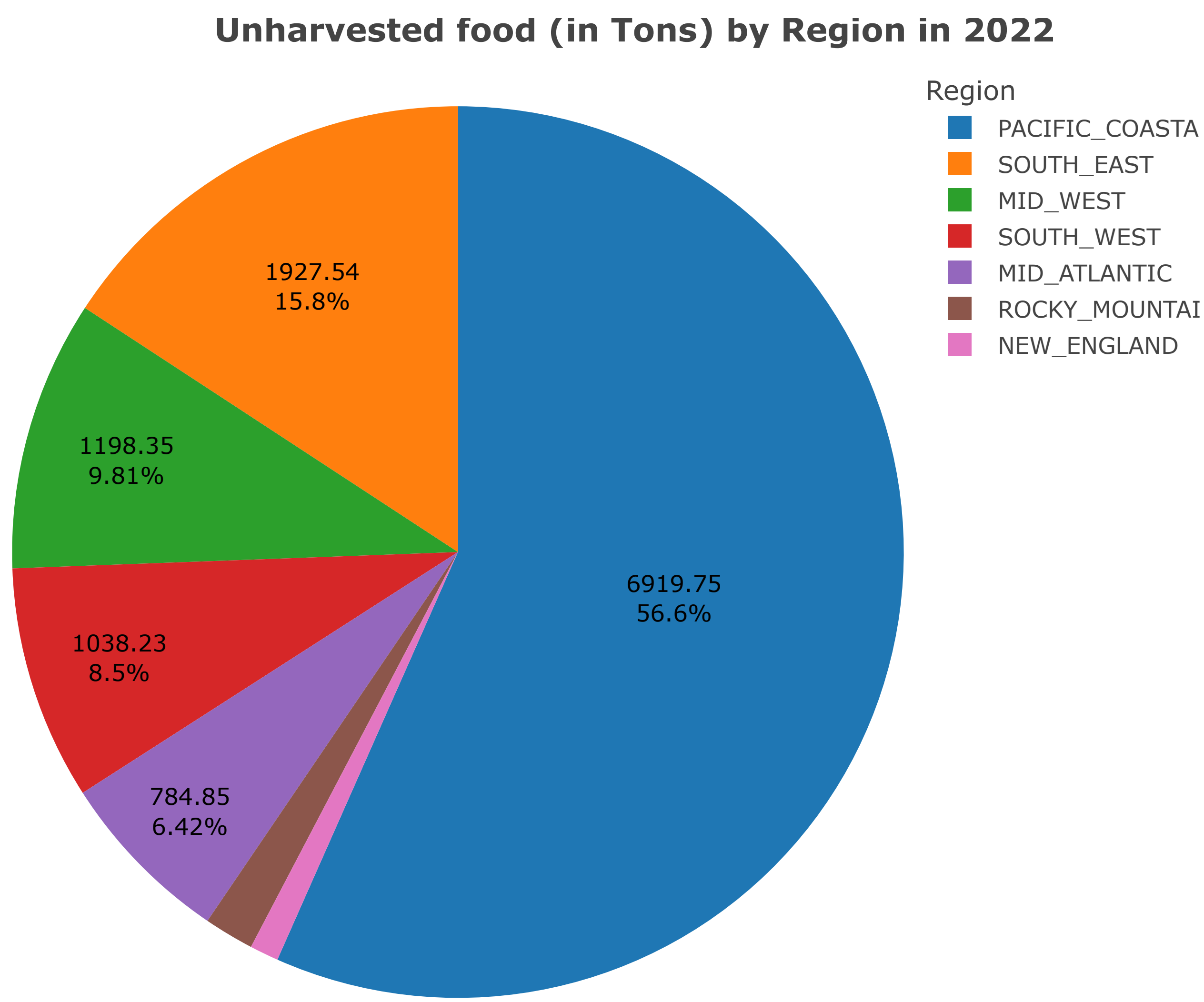
5. Which state wasted the most amount of food and which sectors in 2022?

```
# Creating a  Dataframe to compute the Uneaten food
state_uneaten_df1 <- food_surplus_dtl_df2 %>% filter(year==2022) %>% group_by(state,secto
r) %>% summarize(food_uneaten=sum(tons_uneaten))  %>% ungroup() %>% group_by(state) %>% mu
tate(total_food_uneaten=sum(food_uneaten)) %>% arrange(desc(total_food_uneaten)) %>%  head
(50)
state_uneaten_df1$state <- factor(state_uneaten_df1$state , levels = unique(state_uneaten_
df1$state )[order(state_uneaten_df1$total_food_uneaten, decreasing = TRUE)])
# Plotting a Stacked Bar chart of wasted food by State and sector
state_uneaten_df1 %>%
    plot_ly(x=~state,
          y=~food_uneaten,
          color=~sector,
          textfont = list(color = "black",  size = 10),
          ## below 3 lines for the bar label and hover text
          textposition = "inside",
          texttemplate = '%{y:.2s}'
          ) %>%
  add_bars() %>%
  plotly::layout(title="<b>Food wasted in 2022 by state and sector</b>",
         xaxis = list(title=""),
         yaxis = list(title = "Food Wasted (in Tons)"),
         barmode="stack",
         uniformtext=list(minsize=10, mode='hide'),
         legend = list(title = list(text = "Sector")))
```



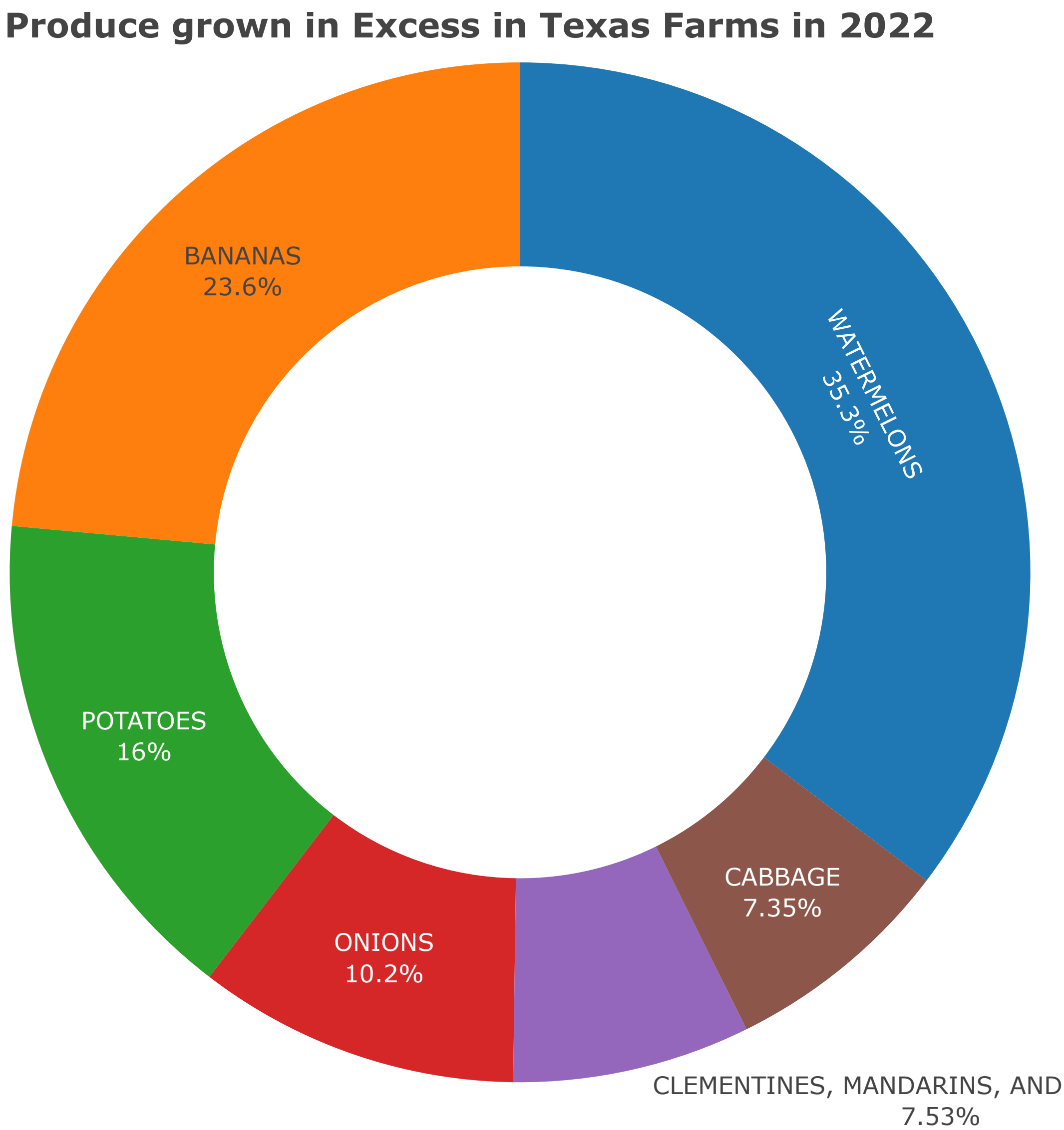
6. Which region wasted most food unharvested?

```
# Creata a dataframe of Unharvested data
unharvested_df <- food_surplus_dtl_df2 %>% filter(year==2022) %>% group_by(region) %>% sum
marize(total_not_harvested=sum(tons_not_harvested))
# Creating a Pie Chart on the Unharvested Dataframe
fig <- plot_ly(unharvested_df, type='pie', labels = ~region, values = ~total_not_harvested
,text = ~paste(total_not_harvested),
          hoverinfo = "text",
          textfont = list(color = "black",  size = 10),
          ## below 3 lines for the bar label and hover text
          textposition = "inside"
          )
# Updating the Title and Legend of the Pie plot
fig <- fig %>% plotly::layout(uniformtext=list(minsize=12, mode='hide'), title="<b> Unharv
ested food (in Tons) by Region in 2022</b>",
                      legend = list(title = list(text = "Region")))
fig
```



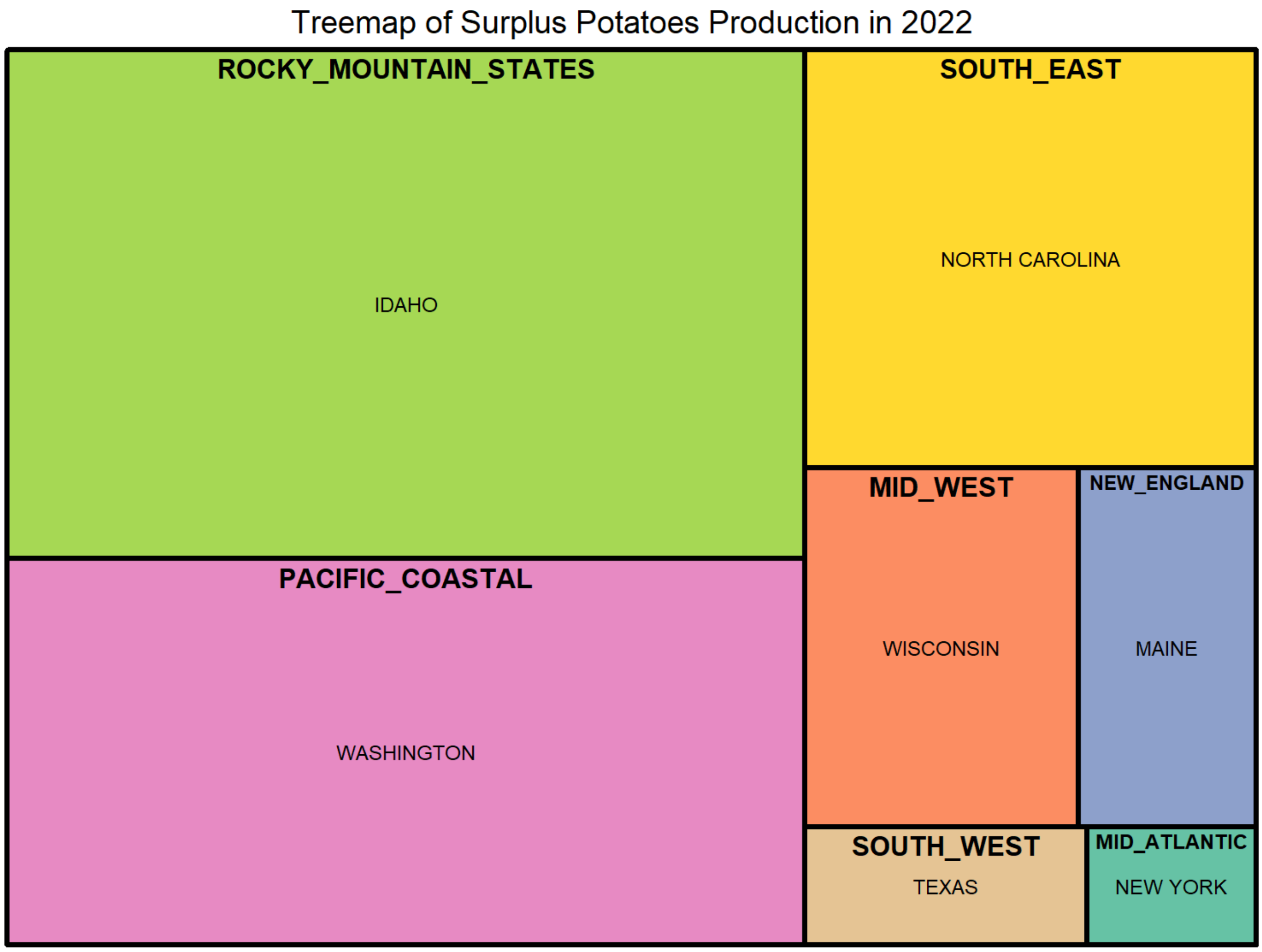
7. Which produce was the most grown in Surplus in Texas Farms in 2022?

```
# Creating a dataframe of Texas Data with only Produce food type
texas_df <- food_surplus_dtl_df2 %>% filter(year==2022,state=="TEXAS",food_type=="PRODUC
E",!food_category=="NOT APPLICABLE") %>% group_by(food_category) %>% summarize(total_surpl
us=sum(tons_surplus)) %>% arrange(desc(total_surplus)) %>% head(6)
# Creating a Donut plot of Surplus grown produce in Texas
plot_ly(texas_df)%>%
  add_pie(texas_df,labels=~factor(food_category),values=~total_surplus,
          textinfo="label+percent",type='pie',hole=0.6)%>%
  plotly::layout(title="<b>Produce grown in Excess in Texas Farms in 2022</b>",
         legend = list(title = list(text = ""))) %>% hide_legend()
```



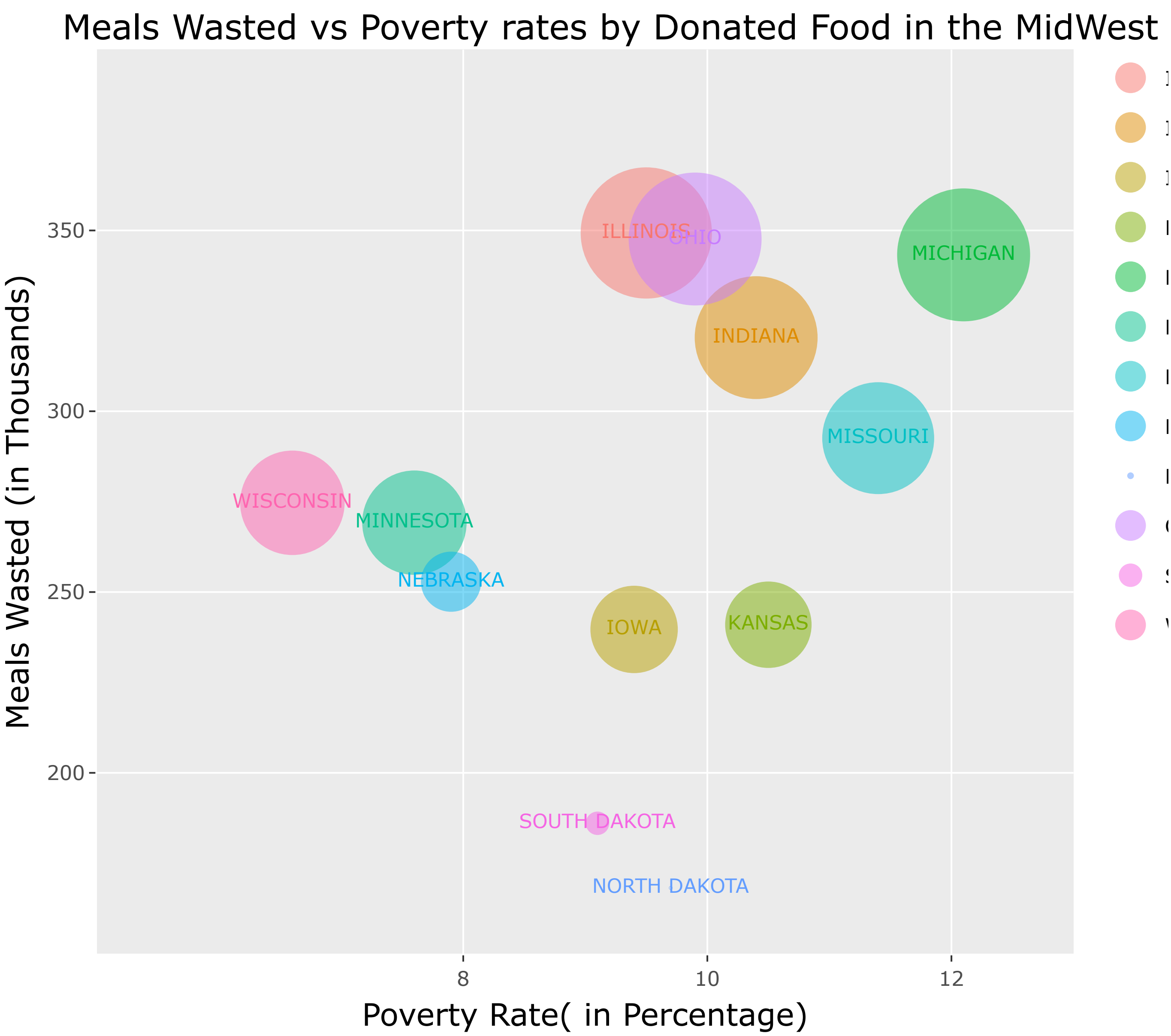
8. Which state produced the most surplus Potatoes in each region in 2022?

```
# Creating datframe to extract the data for Potatoes
potatoes_df <- food_surplus_dtl_df2 %>% filter(year==2022,sector=="FARM",food_type=="PRODU
CE",food_category=="POTATOES") %>% group_by(region,state) %>% summarize(total_surplus=sum
(tons_surplus)) %>%  slice_max(total_surplus,n=1)

# Creating a Treemap of Surplus Potatoes Production
treemap(potatoes_df,index=c("region","state"),vSize="total_surplus",type="index",fontsiz
e.labels = c(12,9),
        fontcolor.labels = c("black","black"),bg.labels = c("transparent"),
        align.labels = list(c("center","top"),c("center","center")),
        border.col = c("black","white"),border.lwds = c(3,1),
        title = "Treemap of Surplus Potatoes Production in 2022",fontfamily.title = 16,pal
ette = "Set2")
```



9. Which Mid west state had highest poverty rate and how much food was Wasted compared in 2022?

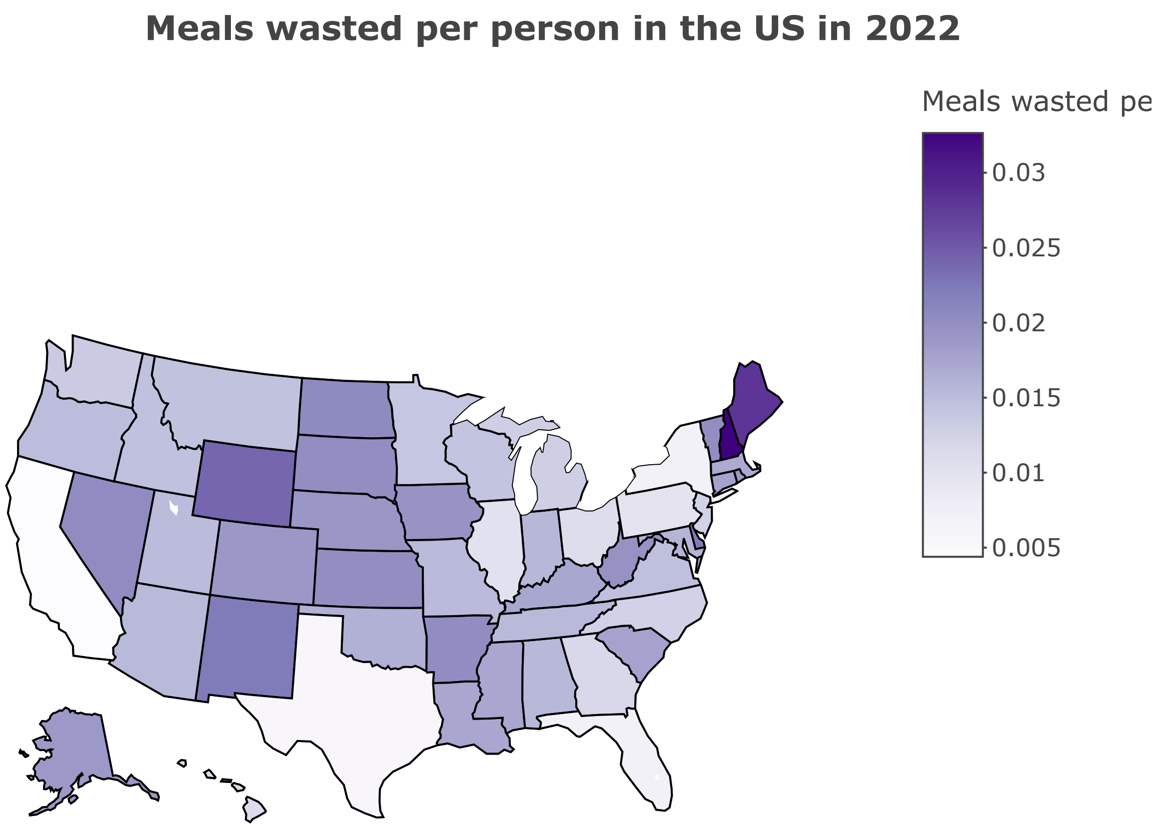
```
# Creating second Dataframe with Mid west data
bubble_df2 <- join_df %>% filter(region %in% c("MID_WEST")) %>% group_by(state) %>% summar
ize(total_wasted=sum(meals_wasted)/1000, total_donated=sum(tons_donated),median_poverty=me
dian(poverty_2022) )
# Creating a Bubble plot of Meals wasted vs Poverty rates by Donated Food
plot3 <- bubble_df2 %>% ggplot(aes(x=median_poverty, y=total_wasted, size=total_donated, c
olor=as.factor(state)))+ geom_point(alpha=0.5)+scale_fill_distiller(palette = "RdPu") +sca
le_size(range=c(0.1,20)) + guides(fill=FALSE) +geom_text(aes(label=state), vjust=1,size=3)
p3 <- ggplotly(plot3, width=800, height=600) %>% plotly::layout(xaxis=list(range=c(5,13),t
itle="Poverty Rate( in Percentage)"),
                                                          yaxis=list(range=c(150,400),titl
e = "Meals Wasted (in Thousands)"),
                                                          title=list(text="Meals Wasted vs
Poverty rates by Donated Food in the MidWest in 2022 ", y = 0.99, x = 0.5, xanchor = 'cent
er', yanchor =  'top'),
                                                          legend = list(title = list(text
= "")))
p3
```



10. Which state wasted the most meals per person in retail?

```r
# Creating a Map Dataframe with Retail Data
map_df1 <- join_df %>% filter(sector %in% c("RETAIL")) %>% group_by(state) %>% summarize(t
otal_meals_wasted=sum(meals_wasted),mean_pop=mean(population_2022)) %>% mutate(meals_per_p
erson=total_meals_wasted/mean_pop) %>% arrange(desc(meals_per_person))
# Joining the dataframe with states data
map_df2 <- map_df1 %>% inner_join(state_df,by="state")
```

```r
# give state boundaries a white border
l <- list(color = toRGB("white"), width = 2)
# specify some map projection/options
g <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
  showlakes = TRUE,
  lakecolor = toRGB('white')
)
fig <- plot_geo(map_df2, locationmode = 'USA-states')
fig <- fig %>% add_trace(
    z = ~meals_per_person, locations = ~Code,
    color = ~meals_per_person, colors = 'Purples', text = ~Code
  )
fig <- fig %>% colorbar(title = "Meals wasted per Person")
fig <- fig %>% plotly::layout(
    title = '<b>Meals wasted per person in the US in 2022</b>',
    geo = g, text = ~Code,
              mode = "text"
  )
fig
```



Feature Engineering

```
target_var <- "meals_wasted"
```

```
# Extracting the columns with DOuble Datatype from merged dataframe
dbl_colnames_join <- join_df %>% select(is.double) %>%  colnames
# Creating a Dataframe with only Numeric features
join_dbl_df0 <- join_df[dbl_colnames_join]
```

```
# Correlation Method to identify columns with Highest correlation
corr_index <- findCorrelation(cor(join_dbl_df0), cutoff=0.9)
highly_correlated_cols <- join_dbl_df0[,corr_index] %>% select(-meals_wasted) %>% colnames
()
highly_correlated_cols
```

```
## [1] "tons_surplus"               "tons_uneaten"
## [3] "tons_waste"                 "total_mtco2e_footprint"
## [5] "upstream_mtco2e_footprint" "tons_donated"
```

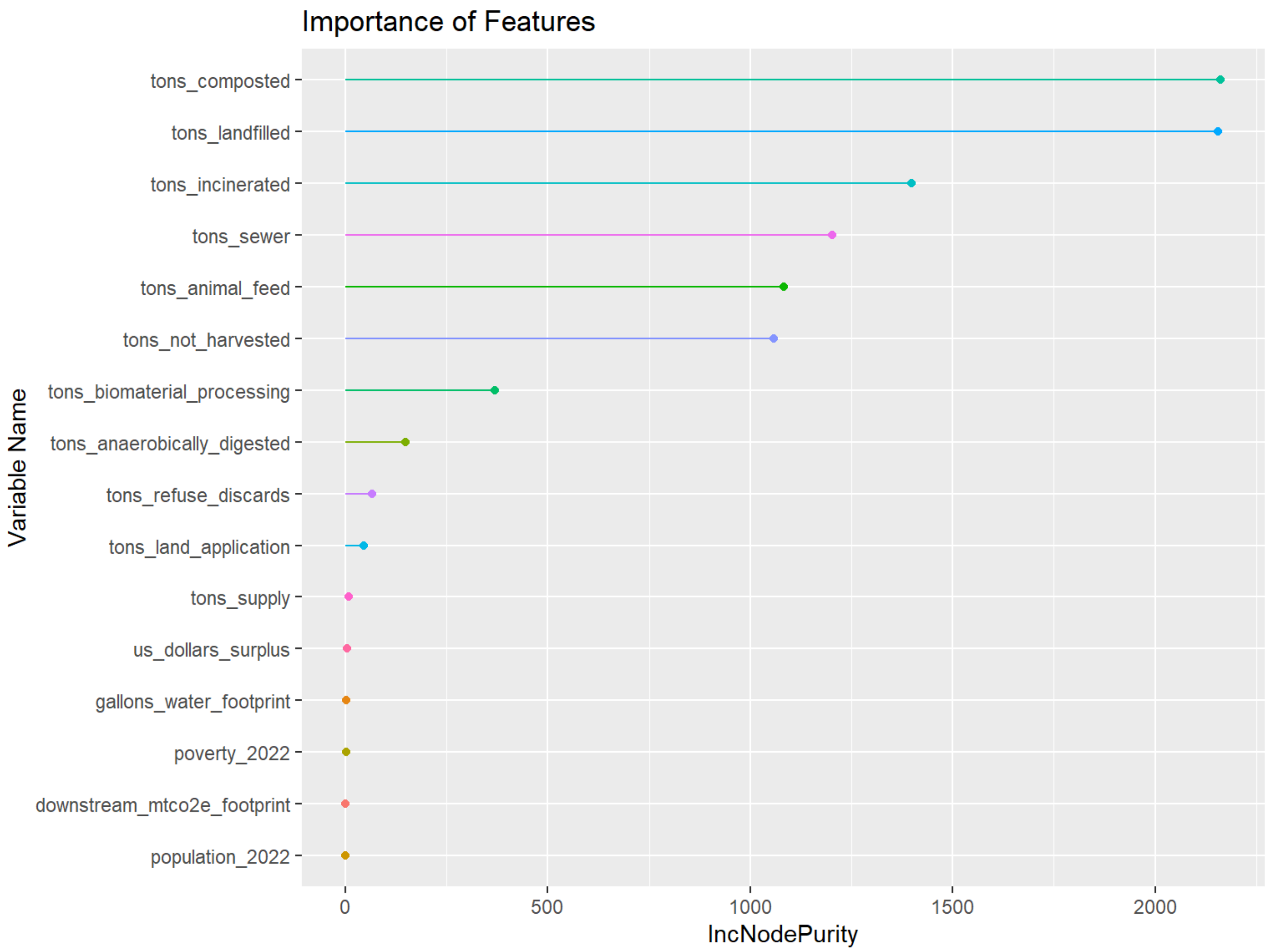
```
# Excluding the Highly correlated columns
join_dbl_df0 <- join_dbl_df0 %>% select(-highly_correlated_cols)
# Creating a Dummy Model based on the Numeric features
lmMod <- lm(meals_wasted ~ . , data = join_dbl_df0)
# Creating a Variable to compute the importance of Features
var_imp <- varImp(lmMod, scale = FALSE)
# Creating a Dataframe with Important features
var_imp_df <- data.frame(cbind(variable = rownames(var_imp), score = var_imp[,1]))
var_imp_df$score <- as.double(var_imp_df$score)
var_imp_df[order(var_imp_df$score,decreasing = TRUE),]
```

	variable <chr>	score <dbl>
6	tons_composted	2162.7805366
10	tons_landfilled	2156.4654818
8	tons_incinerated	1397.6515391
11	tons_sewer	1201.7691128
4	tons_animal_feed	1082.4672382
7	tons_not_harvested	1058.6506660
3	tons_biomaterial_processing	369.7987568

	variable <chr>	score <dbl>
5	tons_anaerobically_digested	149.0063682
12	tons_refuse_discards	65.9560749
9	tons_land_application	46.2169720

1-10 of 16 rows Previous 1 2 Next

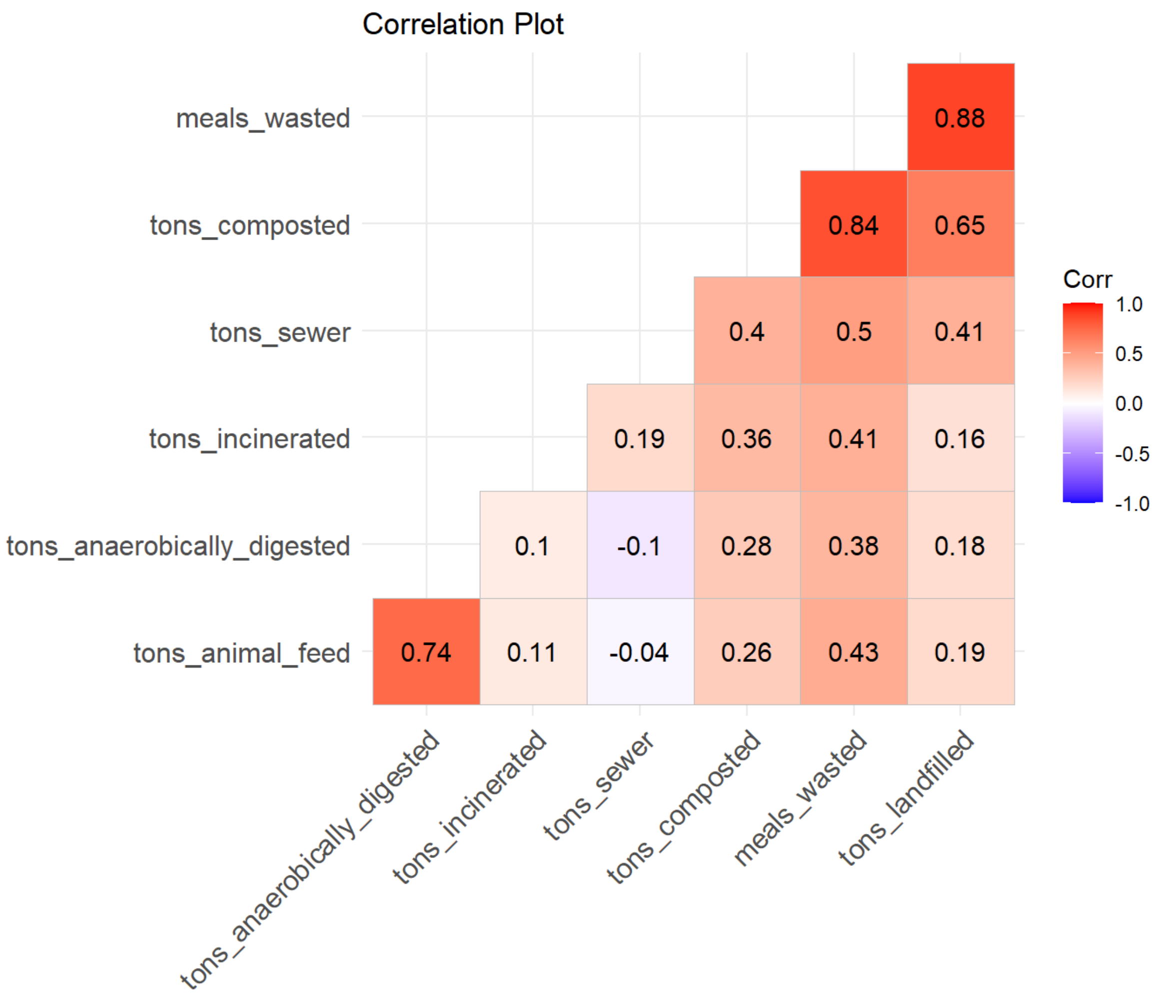
```
# Plotting the Important Features
ggplot(var_imp_df, aes(x=reorder(variable, score), y=score,color=variable)) +
  geom_point() +
  geom_segment(aes(x=variable,xend=variable,y=0,yend=score)) +
  ylab("IncNodePurity") +
  xlab("Variable Name") + ggtitle("Importance of Features")+
  coord_flip()+scale_color_discrete(guide="none")
```




```
# Selecting only the important featues and fing the correlation between them
imp_features_df <- join_dbl_df0 %>% select(meals_wasted,tons_composted,tons_landfilled,ton
s_incinerated,tons_sewer,tons_animal_feed,tons_anaerobically_digested)
df_correlation <- cor(imp_features_df, use="complete.obs")
round(df_correlation,1)
```

```
##                              meals_wasted tons_composted tons_landfilled
## meals_wasted                         1.0            0.8             0.9
## tons_composted                       0.8            1.0             0.6
## tons_landfilled                      0.9            0.6             1.0
## tons_incinerated                     0.4            0.4             0.2
## tons_sewer                           0.5            0.4             0.4
## tons_animal_feed                     0.4            0.3             0.2
## tons_anaerobically_digested          0.4            0.3             0.2
##                              tons_incinerated tons_sewer tons_animal_feed
## meals_wasted                             0.4        0.5              0.4
## tons_composted                           0.4        0.4              0.3
## tons_landfilled                          0.2        0.4              0.2
## tons_incinerated                         1.0        0.2              0.1
## tons_sewer                               0.2        1.0              0.0
## tons_animal_feed                         0.1        0.0              1.0
## tons_anaerobically_digested              0.1       -0.1              0.7
##                              tons_anaerobically_digested
## meals_wasted                                         0.4
## tons_composted                                       0.3
## tons_landfilled                                      0.2
## tons_incinerated                                     0.1
## tons_sewer                                          -0.1
## tons_animal_feed                                     0.7
## tons_anaerobically_digested                          1.0
```

```
# Plotting Correlation Matrix
ggcorrplot(df_correlation,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE) +
  labs(title = "Correlation Plot")
```




```
# Plotting a linear model with cleaned up data
set.seed(38)
# Excluding the Highly correlated columns
join_dbl_df2 <- join_dbl_df0 %>% select(-tons_not_harvested,-tons_composted,-tons_landfill
ed)
N <- nrow(join_dbl_df2)
gp <- runif(N)
df_train_lm1 <- join_dbl_df2[gp < 0.8, ]
df_test_lm1 <- join_dbl_df2[gp >= 0.8, ]
nrow(df_train_lm1)
```

```
## [1] 24126
```

```
nrow(df_test_lm1)
```

```
## [1] 6138
```

```
model_lm1 <- lm(meals_wasted ~ 0+., data=df_train_lm1)
summary(model_lm1)
```

```
##
## Call:
## lm(formula = meals_wasted ~ 0 + ., data = df_train_lm1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1380.91   -64.82   -37.79    -3.89  2533.29
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## tons_supply                   3.116e+00  1.806e-01  17.249   <2e-16 ***
## us_dollars_surplus            1.801e-02  5.961e-04  30.213   <2e-16 ***
## tons_biomaterial_processing 1.941e+03  1.280e+02  15.169   <2e-16 ***
## tons_animal_feed              1.620e+03  4.001e+01  40.496   <2e-16 ***
## tons_anaerobically_digested 2.285e+03  2.659e+02   8.593   <2e-16 ***
## tons_incinerated              1.051e+03  2.644e+01  39.732   <2e-16 ***
## tons_land_application         1.770e+04  9.253e+02  19.128   <2e-16 ***
## tons_sewer                    2.312e+03  3.499e+01  66.072   <2e-16 ***
## tons_refuse_discards          1.579e+04  5.200e+02  30.373   <2e-16 ***
## downstream_mtco2e_footprint 1.579e+03  1.129e+01 139.878   <2e-16 ***
## gallons_water_footprint       2.601e-04  8.984e-06  28.946   <2e-16 ***
## population_2022               3.202e-07  2.272e-07   1.409    0.159
## poverty_2022                  4.469e+00  1.874e-01  23.847   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 230.2 on 24113 degrees of freedom
## Multiple R-squared:  0.9105, Adjusted R-squared:  0.9105
## F-statistic: 1.888e+04 on 13 and 24113 DF,  p-value: < 2.2e-16
```

```
# Identifying columns with p values less than 0.05
lm1_colnames <- tidy(model_lm1)  %>% filter(p.value <0.05) %>% select(term) %>% unlist %>%
as.vector
lm1_colnames
```

```
##  [1] "tons_supply"                 "us_dollars_surplus"
##  [3] "tons_biomaterial_processing" "tons_animal_feed"
##  [5] "tons_anaerobically_digested" "tons_incinerated"
##  [7] "tons_land_application"       "tons_sewer"
##  [9] "tons_refuse_discards"        "downstream_mtco2e_footprint"
## [11] "gallons_water_footprint"     "poverty_2022"
```

```
# Building a second Model to validate the results
df_lm2 <- join_dbl_df2 %>% select(meals_wasted,lm1_colnames)
N_lm2 <- nrow(df_lm2)
gp_lm2 <- runif(N_lm2)
# Creating test and Train datasets
df_train_lm2 <- df_lm2[gp < 0.75, ]
df_test_lm2 <- df_lm2[gp >= 0.75, ]
nrow(df_train_lm2)
```

```
## [1] 22602
```

```
nrow(df_test_lm2)
```

```
## [1] 7662
```

```
model_lm2 <- lm(meals_wasted ~ 0+., data=df_train_lm2)
summary(model_lm2)
```

```
##
## Call:
## lm(formula = meals_wasted ~ 0 + ., data = df_train_lm2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1389.77   -64.40   -37.38    -4.00  2535.30
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## tons_supply                 3.215e+00  1.789e-01  17.967   <2e-16 ***
## us_dollars_surplus          1.771e-02  6.033e-04  29.358   <2e-16 ***
## tons_biomaterial_processing 1.963e+03  1.310e+02  14.985   <2e-16 ***
## tons_animal_feed            1.622e+03  4.069e+01  39.863   <2e-16 ***
## tons_anaerobically_digested 2.362e+03  2.721e+02   8.679   <2e-16 ***
## tons_incinerated            1.039e+03  2.690e+01  38.628   <2e-16 ***
## tons_land_application       1.761e+04  9.481e+02  18.578   <2e-16 ***
## tons_sewer                  2.322e+03  3.605e+01  64.424   <2e-16 ***
## tons_refuse_discards        1.564e+04  5.295e+02  29.547   <2e-16 ***
## downstream_mtco2e_footprint 1.585e+03  1.159e+01 136.775   <2e-16 ***
## gallons_water_footprint     2.573e-04  9.223e-06  27.894   <2e-16 ***
## poverty_2022                4.545e+00  1.642e-01  27.684   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 228.6 on 22590 degrees of freedom
## Multiple R-squared:  0.9117, Adjusted R-squared:  0.9117
## F-statistic: 1.944e+04 on 12 and 22590 DF,  p-value: < 2.2e-16
```

```
# Identifying columns with p values less than 0.05 in the second model
lm2_colnames <- tidy(model_lm2)  %>% filter(p.value <0.05) %>% select(term) %>% unlist %>%
as.vector
lm2_colnames
```

```
##  [1] "tons_supply"                 "us_dollars_surplus"
##  [3] "tons_biomaterial_processing" "tons_animal_feed"
##  [5] "tons_anaerobically_digested" "tons_incinerated"
##  [7] "tons_land_application"       "tons_sewer"
##  [9] "tons_refuse_discards"        "downstream_mtco2e_footprint"
## [11] "gallons_water_footprint"     "poverty_2022"
```

```r
# Selecting only the important featues and fing the correlation between them
imp_features_df2 <- join_dbl_df2 %>% select(lm2_colnames,meals_wasted)
df_correlation2 <- cor(imp_features_df2, use="complete.obs")
round(df_correlation2,1)
```

```
##                             tons_supply us_dollars_surplus
## tons_supply                         1.0                0.2
## us_dollars_surplus                  0.2                1.0
## tons_biomaterial_processing         0.4                0.2
## tons_animal_feed                    0.4                0.3
## tons_anaerobically_digested         0.6                0.3
## tons_incinerated                    0.1                0.4
## tons_land_application               0.5                0.2
## tons_sewer                          0.0                0.4
## tons_refuse_discards                0.0                0.0
## downstream_mtco2e_footprint         0.2                0.7
## gallons_water_footprint             0.2                0.5
## poverty_2022                        0.1                0.0
## meals_wasted                        0.4                0.7
##                             tons_biomaterial_processing tons_animal_feed
## tons_supply                                         0.4              0.4
## us_dollars_surplus                                  0.2              0.3
## tons_biomaterial_processing                         1.0              0.6
## tons_animal_feed                                    0.6              1.0
## tons_anaerobically_digested                         0.7              0.7
## tons_incinerated                                    0.1              0.1
## tons_land_application                               0.5              0.6
## tons_sewer                                         -0.1              0.0
## tons_refuse_discards                                0.0              0.1
## downstream_mtco2e_footprint                         0.0              0.1
## gallons_water_footprint                             0.3              0.3
## poverty_2022                                        0.0              0.0
## meals_wasted                                        0.3              0.4
##                             tons_anaerobically_digested tons_incinerated
## tons_supply                                         0.6              0.1
## us_dollars_surplus                                  0.3              0.4
## tons_biomaterial_processing                         0.7              0.1
## tons_animal_feed                                    0.7              0.1
## tons_anaerobically_digested                         1.0              0.1
## tons_incinerated                                    0.1              1.0
## tons_land_application                               0.8              0.1
## tons_sewer                                         -0.1              0.2
## tons_refuse_discards                                0.0              0.0
## downstream_mtco2e_footprint                         0.0              0.3
## gallons_water_footprint                             0.3              0.2
## poverty_2022                                        0.0             -0.1
## meals_wasted                                        0.4              0.4
##                             tons_land_application tons_sewer
## tons_supply                                   0.5        0.0
## us_dollars_surplus                            0.2        0.4
## tons_biomaterial_processing                   0.5       -0.1
## tons_animal_feed                              0.6        0.0
## tons_anaerobically_digested                   0.8       -0.1
## tons_incinerated                              0.1        0.2
## tons_land_application                         1.0       -0.1
## tons_sewer                                   -0.1        1.0
## tons_refuse_discards                          0.0        0.0
## downstream_mtco2e_footprint                   0.1        0.5
## gallons_water_footprint                       0.1        0.2
```

```
## poverty_2022                                      0.0         0.0
## meals_wasted                                      0.4         0.5
##                              tons_refuse_discards downstream_mtco2e_footprint
## tons_supply                                   0.0                         0.2
## us_dollars_surplus                            0.0                         0.7
## tons_biomaterial_processing                   0.0                         0.0
## tons_animal_feed                              0.1                         0.1
## tons_anaerobically_digested                   0.0                         0.0
## tons_incinerated                              0.0                         0.3
## tons_land_application                         0.0                         0.1
## tons_sewer                                    0.0                         0.5
## tons_refuse_discards                          1.0                         0.0
## downstream_mtco2e_footprint                   0.0                         1.0
## gallons_water_footprint                       0.0                         0.4
## poverty_2022                                  0.0                         0.0
## meals_wasted                                  0.1                         0.8
##                              gallons_water_footprint poverty_2022 meals_wasted
## tons_supply                                      0.2          0.1          0.4
## us_dollars_surplus                               0.5          0.0          0.7
## tons_biomaterial_processing                      0.3          0.0          0.3
## tons_animal_feed                                 0.3          0.0          0.4
## tons_anaerobically_digested                      0.3          0.0          0.4
## tons_incinerated                                 0.2         -0.1          0.4
## tons_land_application                            0.1          0.0          0.4
## tons_sewer                                       0.2          0.0          0.5
## tons_refuse_discards                             0.0          0.0          0.1
## downstream_mtco2e_footprint                      0.4          0.0          0.8
## gallons_water_footprint                          1.0          0.0          0.5
## poverty_2022                                     0.0          1.0          0.0
## meals_wasted                                     0.5          0.0          1.0
```

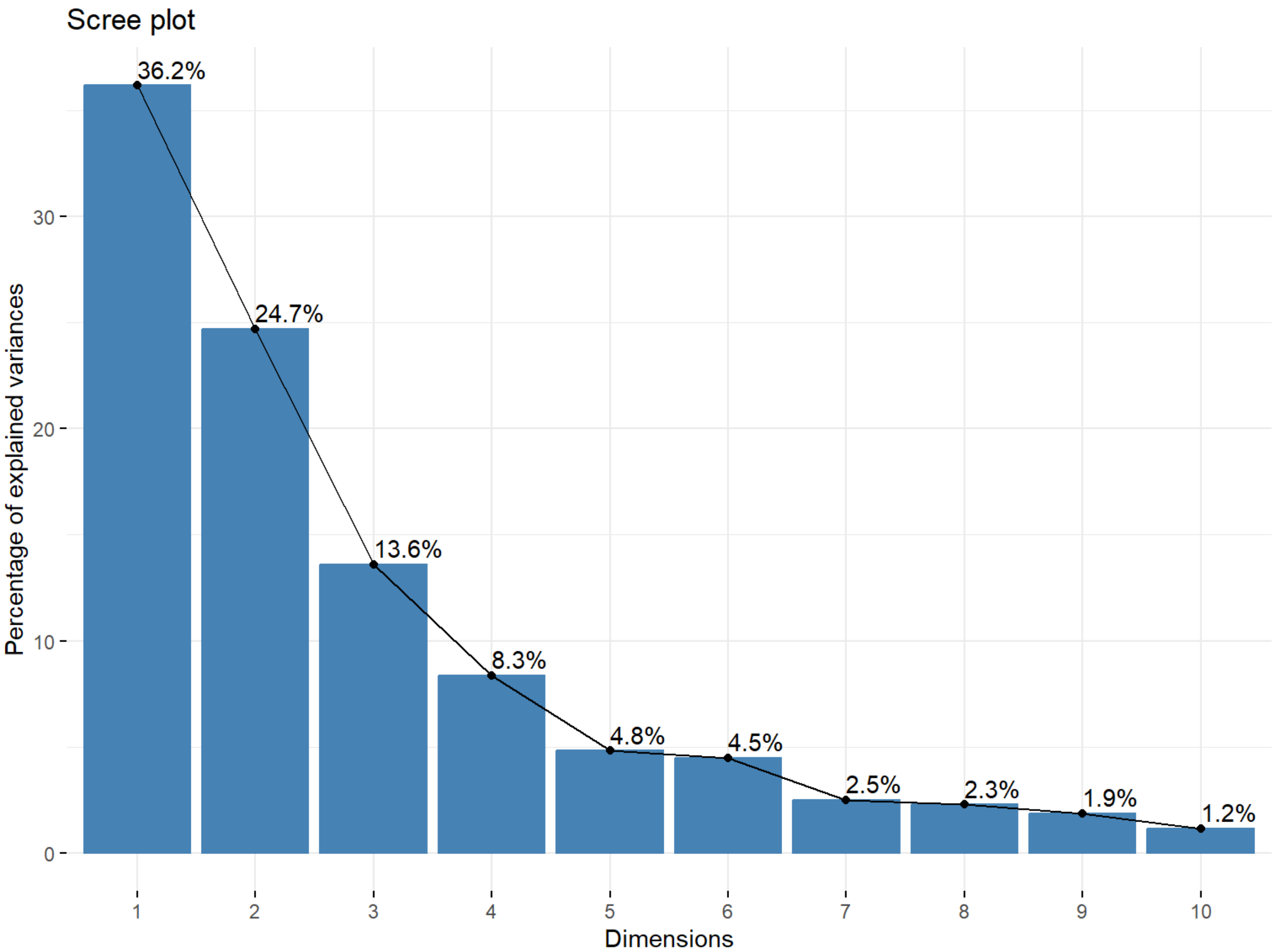
```
# Function for R-Squared Values
r_squared <- function(predcol, ycol) {
  tss = sum( (ycol - mean(ycol))^2 )
  rss = sum( (predcol - ycol)^2 )
  1 - rss/tss
}
# Function to compute RMSE values
rmse_fn <- function(predcol, ycol) {
  res = predcol-ycol
  sqrt(mean(res^2))
}
```

```
# Creating a Dataframe to implement PCA
df101 <- join_dbl_df2 %>% select(lm2_colnames,meals_wasted)
vars <- df101 %>% colnames
# Creating a Train and Test split
data_split <- initial_split(df101, prop = 0.80, strata = meals_wasted)
df_train <- training(data_split)
df_test  <-  testing(data_split)
df_pc_train <- df_train %>% select(-meals_wasted)
# Standardizing the data
train_normalized <- scale(df_pc_train)
pca_new <- PCA(t(train_normalized), ncp=6,graph=FALSE)
pca_df <- as.data.frame(pca_new$var$coord)
summary(pca_new)
```

```
##
## Call:
## PCA(X = t(train_normalized), ncp = 6, graph = FALSE)
##
##
## Eigenvalues
##                          Dim.1    Dim.2    Dim.3    Dim.4    Dim.5    Dim.6
## Variance              8755.424 5974.153 3289.550 2020.905 1164.868 1080.680
## % of var.               36.166   24.677   13.588    8.348    4.812    4.464
## Cumulative % of var.    36.166   60.843   74.432   82.779   87.591   92.055
##                          Dim.7    Dim.8    Dim.9   Dim.10   Dim.11
## Variance               601.309  551.610  448.169  278.829   43.502
## % of var.                2.484    2.279    1.851    1.152    0.180
## Cumulative % of var.    94.539   96.817   98.669   99.820  100.000
##
## Individuals (the 10 first)
##                                     Dist      Dim.1      ctr     cos2      Dim.2
## tons_supply                    |  155.604 |    3.734    0.013    0.001 |  -16.437
## us_dollars_surplus             |  151.586 |   -0.375    0.000    0.000 | -126.340
## tons_biomaterial_processing    |   90.166 |   43.477    1.799    0.233 |   60.877
## tons_animal_feed               |   95.910 |   28.505    0.773    0.088 |   19.382
## tons_anaerobically_digested    |  113.513 |   29.852    0.848    0.069 |   23.804
## tons_incinerated               |  129.082 |   49.861    2.366    0.149 |   20.187
## tons_land_application          |  104.096 |   41.949    1.675    0.162 |   51.301
## tons_sewer                     |  123.429 |   32.436    1.001    0.069 |  -17.695
## tons_refuse_discards           |  174.886 |   68.094    4.413    0.152 |  128.523
## downstream_mtco2e_footprint    |  178.917 |  -15.942    0.242    0.008 | -162.133
##                                    ctr     cos2      Dim.3      ctr     cos2
## tons_supply                      0.377    0.011 |  -99.468   25.064    0.409 |
## us_dollars_surplus              22.265    0.695 |    4.710    0.056    0.001 |
## tons_biomaterial_processing      5.170    0.456 |  -27.198    1.874    0.091 |
## tons_animal_feed                 0.524    0.041 |  -59.748    9.043    0.388 |
## tons_anaerobically_digested      0.790    0.044 |  -84.301   18.003    0.552 |
## tons_incinerated                 0.568    0.024 |   43.653    4.827    0.114 |
## tons_land_application            3.671    0.243 |   -9.573    0.232    0.008 |
## tons_sewer                       0.437    0.021 |   92.341   21.601    0.560 |
## tons_refuse_discards            23.041    0.540 |   78.445   15.589    0.201 |
## downstream_mtco2e_footprint     36.668    0.821 |   32.430    2.664    0.033 |
##
## Variables (the 10 first)
##                                   Dim.1    ctr   cos2    Dim.2    ctr   cos2
## V1                             | -0.821  0.008  0.673 |  0.527  0.005  0.278 |
## V2                             | -0.797  0.007  0.636 |  0.569  0.005  0.324 |
## V3                             | -0.798  0.007  0.637 |  0.563  0.005  0.317 |
## V4                             | -0.783  0.007  0.613 |  0.588  0.006  0.346 |
## V5                             | -0.792  0.007  0.627 |  0.569  0.005  0.324 |
## V6                             | -0.793  0.007  0.629 |  0.569  0.005  0.324 |
## V7                             | -0.796  0.007  0.634 |  0.568  0.005  0.323 |
## V8                             | -0.826  0.008  0.681 |  0.527  0.005  0.278 |
## V9                             | -0.817  0.008  0.668 |  0.533  0.005  0.284 |
## V10                            | -0.799  0.007  0.639 |  0.569  0.005  0.324 |
##                                 Dim.3    ctr   cos2
## V1                              0.207  0.001  0.043 |
## V2                              0.195  0.001  0.038 |
```

```
## V3                          0.207  0.001  0.043 |
## V4                          0.197  0.001  0.039 |
## V5                          0.211  0.001  0.044 |
## V6                          0.208  0.001  0.043 |
## V7                          0.201  0.001  0.040 |
## V8                          0.189  0.001  0.036 |
## V9                          0.202  0.001  0.041 |
## V10                         0.191  0.001  0.036 |
```

```
# Plotting the PCA plot
fviz_eig(pca_new, addlabels = TRUE)
```



Building Regression Models

Linear Regression Models

```r
# Function to create Dataset from Merged dataframe
create_dataset <- function() {
    all_fact_colnames <- join_df %>% select(is.factor) %>%  colnames
    df <- join_df %>% select(all_fact_colnames,lm2_colnames,meals_wasted)
    vars <- df %>% colnames
    df <- df %>%  select(all_of(vars)) %>%  filter(complete.cases(.))
    }

# Function to create a Train and Test split. Returns Train and Test Datasets
create_train_test_split <- function(df){
    data_split <- initial_split(df, prop = 0.80, strata = meals_wasted)
    df_train <- training(data_split)
    df_test  <-  testing(data_split)
    return(list(df_train=df_train,df_test=df_test))
}

# Function to Train model
train_model <- function(df_train){
    model <- train(
    meals_wasted ~ .,
    data = df_train,
    method = 'lm',
    trControl = trainControl(method = 'cv', number = 3)
)
    return(model)
}

# Function that computes the performance of Model such as RMSE and R-squared
predict_publish_results <- function(df_test,model){
    df_test <- df_test %>% add_column(predictions = predict(model, df_test))
    # Get the rmse of the cross-validation predictions
    rmse_val <- rmse_fn(df_test$predictions, df_test$meals_wasted)
    # Get the rmse of the cross-validation predictions
    r_squared_val <- r_squared(df_test$predictions, df_test$meals_wasted)
    return(list(df_test=df_test,rmse_val=rmse_val,r_squared_val=r_squared_val))
}

# Function that plots the results of the model. Takes the Model and the train & Test sets
as inputs
create_model_plots <- function(df_train,df_test,model){
    plot1 <- df_test %>%
    ggplot(aes(meals_wasted, predictions)) +
     geom_point(shape = 1, size = 1.5, alpha = 0.6) +
    geom_abline() +
    labs(title = 'Predicted vs. Actual Meals Wasted',
       x = 'Predicted ',
       y = 'Actual ') +
    theme_bw()
    df_lm <- broom::augment(model$finalModel, data = df_train)
    plot2 <- ggplot(df_lm, aes(.fitted, .std.resid)) +
    geom_point(shape = 1, size = 1.5, alpha = 0.6) +
    labs(title = 'Predicted vs. Residuals - Model',
       x = 'Predicted Meals',
       y = 'Residuals') + theme_bw()
```

```
    return(list(plot1=plot1,plot2=plot2))
}

# Function to perform PCA on the Dataframe. Takes number of PCA features as Parameters ncp
perform_pca <- function(df,ncp){
    df_pc <- df %>% select(-meals_wasted )
    normalized_df <- scale(df_pc)
    pca_new <- PCA(t(normalized_df), ncp=ncp,graph=FALSE)
    temp_pca_df <- as.data.frame(pca_new$var$coord)
    pca_df <- bind_cols(temp_pca_df,meals_wasted=df$meals_wasted)
    return(pca_df)
}

# Function to Create Models using PCA. This inturn calls the functions created above
create_models <- function(df,ncp) {
    return_lm <- create_train_test_split(df)
    df_train <- return_lm$df_train
    df_test <- return_lm$df_test
    df_pca_train <- perform_pca(df_train,ncp)
    df_pca_test <- perform_pca(df_test,ncp)
    model <- train_model(df_pca_train)
    predict_results <- predict_publish_results(df_pca_test,model)
    df_pca_test <- predict_results$df_test
    rmse_val <- predict_results$rmse_val
    r_squared_val <- predict_results$r_squared_val
    plot_object <- create_model_plots(df_pca_train,df_pca_test,model)
    return(list(rmse_val=rmse_val,r_squared_val=r_squared_val,plot1=plot_object$plot1,plot
2=plot_object$plot2))
}

# Function to Create Models without PCA. This inturn calls the functions created above
create_models_without_pca <- function(df) {
    return_lm <- create_train_test_split(df)
    df_train <- return_lm$df_train
    df_test <- return_lm$df_test
    model <- train_model(df_train)
    predict_results <- predict_publish_results(df_test,model)
    df_test <- predict_results$df_test
    rmse_val <- predict_results$rmse_val
    r_squared_val <- predict_results$r_squared_val
    plot_object <- create_model_plots(df_train,df_test,model)
    return(list(rmse_val=rmse_val,r_squared_val=r_squared_val,plot1=plot_object$plot1,plot
2=plot_object$plot2))
}
```

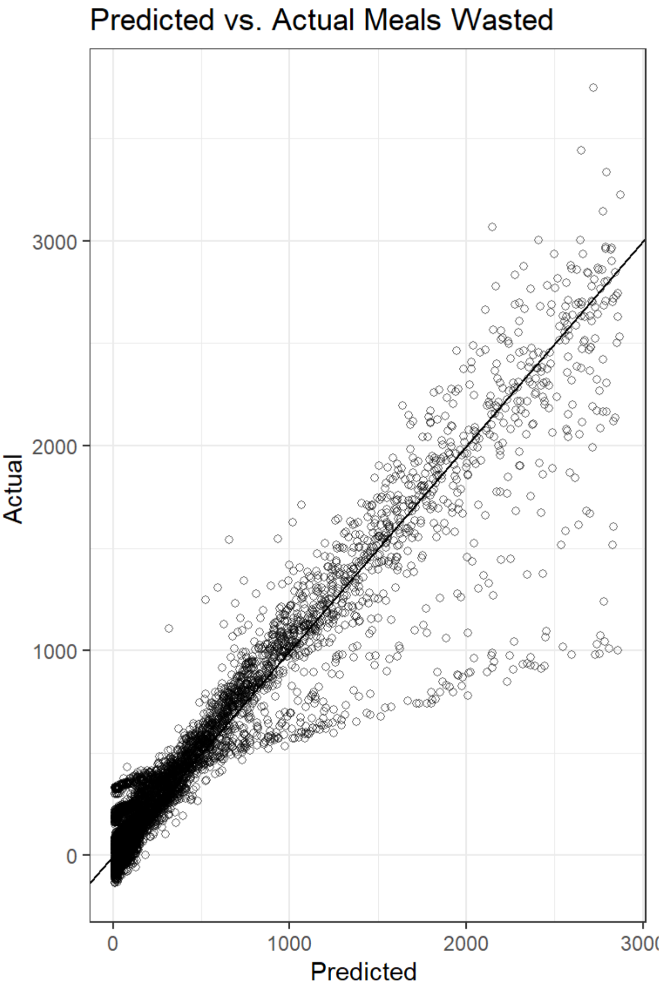
```
# Creating a adTaframe to capture the Model results
model_results_df <- data.frame(ncp=integer(),
                 modelname=character(),
                 Rsquared = double(), RMSEmodel = double(),
                 stringsAsFactors=FALSE)
```

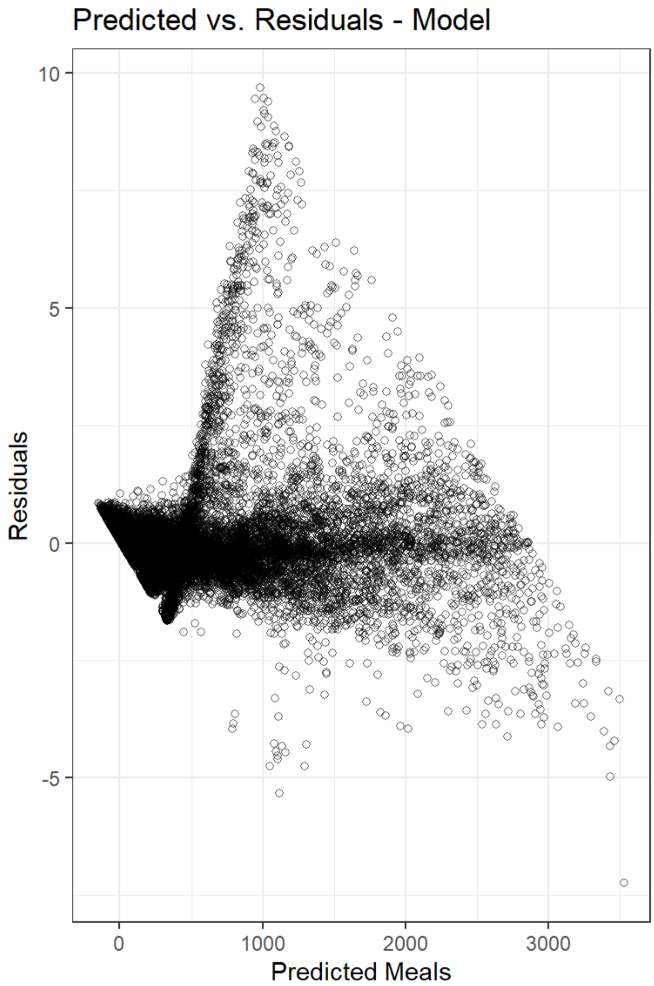
```
# Creating a Simple Regression Model
df_lm4 <- create_dataset()
return_model_lm4 <- create_models_without_pca(df_lm4)
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Simple Model",Rsquared=re
turn_model_lm4$r_squared_val,RMSEmodel=return_model_lm4$rmse_val))
model_results_df
```

ncp	modelname	Rsquared	RMSEmodel
<dbl>	<chr>	<dbl>	<dbl>
0	Simple Model	0.9057294	191.5152

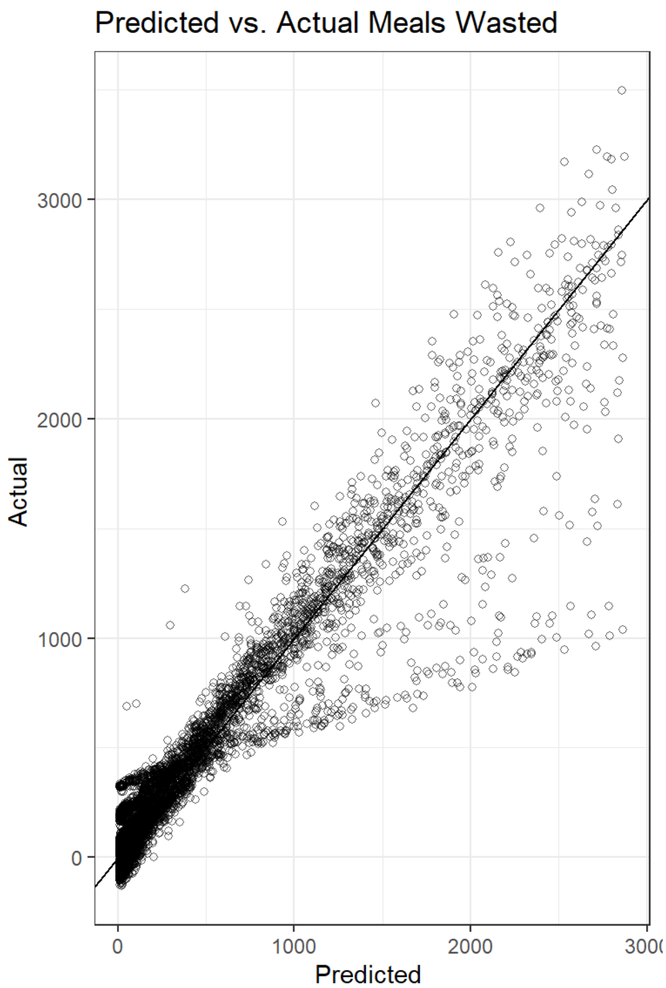
1 row

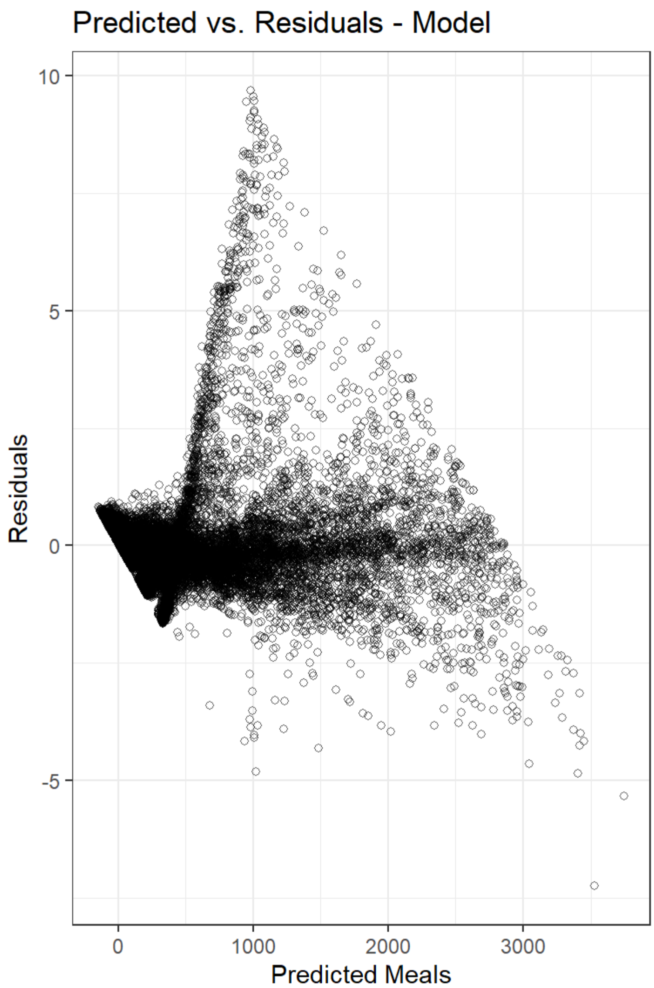
```
gridExtra::grid.arrange(return_model_lm4$plot1, return_model_lm4$plot2, nrow = 1)
```





```
# Creating a Caret Regression Model
df_caret5 <- create_dataset()
# Handling the dummy variables
dummies_model_caret <- dummyVars(meals_wasted ~ ., data = df_caret5, fullRank = T)
df_caret5_mat <- predict(dummies_model_caret, newdata = df_caret5)
meals_wasted <- df_caret5 %>%     select(meals_wasted) %>%    unlist(use.names = FALSE)
df_caret5_dummy <- data.frame(df_caret5_mat) %>%  add_column(meals_wasted = meals_wasted)
%>%  select(meals_wasted, everything())
return_model_caret5 <- create_models_without_pca(df_caret5_dummy)
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Caret Model",Rsquared=ret
urn_model_caret5$r_squared_val,RMSEmodel=return_model_caret5$rmse_val))
gridExtra::grid.arrange(return_model_caret5$plot1, return_model_caret5$plot2, nrow = 1)
```






```
# Creating Regression MOdel using vTreat Package
df_vtreat6 <- create_dataset()
# Creating a Treatment plan
treatment_plan <- designTreatmentsN(
      df_vtreat6,
      colnames(df_vtreat6),
      outcomename = 'meals_wasted',
      codeRestriction = c('lev', 'catN', 'clean', 'isBAD')
)
```

```
## [1] "vtreat 1.6.5 inspecting inputs Sun Jun 30 13:52:58 2024"
## [1] "designing treatments Sun Jun 30 13:52:58 2024"
## [1] " have initial level statistics Sun Jun 30 13:52:58 2024"
## [1] " scoring treatments Sun Jun 30 13:53:00 2024"
## [1] "have treatment plan Sun Jun 30 13:53:02 2024"
## [1] "rescoring complex variables Sun Jun 30 13:53:02 2024"
## [1] "done rescoring complex variables Sun Jun 30 13:53:05 2024"
```

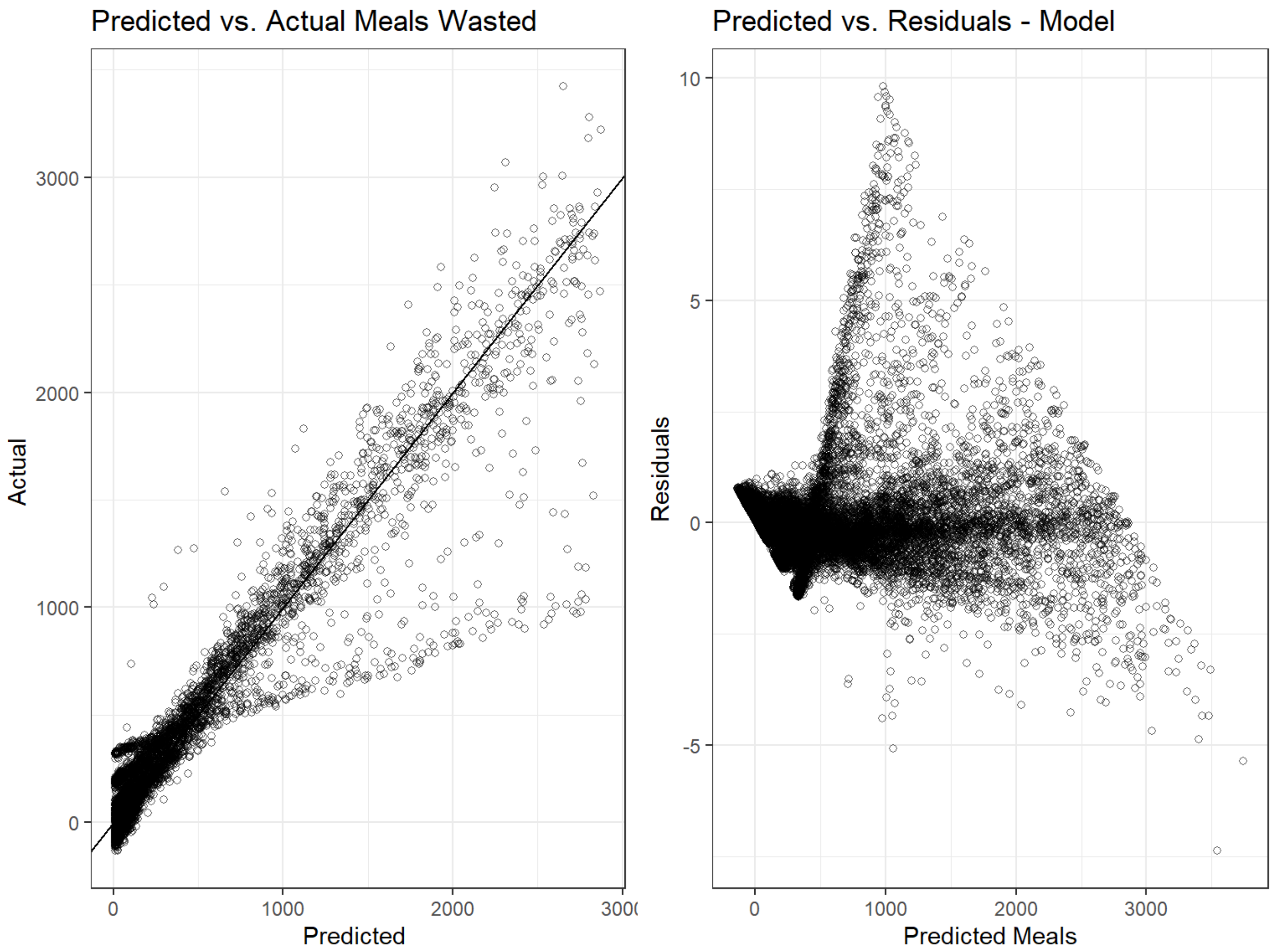
```
# Preparing the Data for Model building
df_trt6 <- vtreat::prepare(treatment_plan, df_vtreat6)
# Creating models without implementing the PCA
return_model_vtreat6 <- create_models_without_pca(df_trt6)
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="vTreat Model",Rsquared=re
turn_model_vtreat6$r_squared_val,RMSEmodel=return_model_vtreat6$rmse_val))
return_model_vtreat6$rmse_val
```

```
## [1] 197.5938
```

```
return_model_vtreat6$r_squared_val
```

```
## [1] 0.8994878
```

```
gridExtra::grid.arrange(return_model_vtreat6$plot1, return_model_vtreat6$plot2, nrow = 1)
```



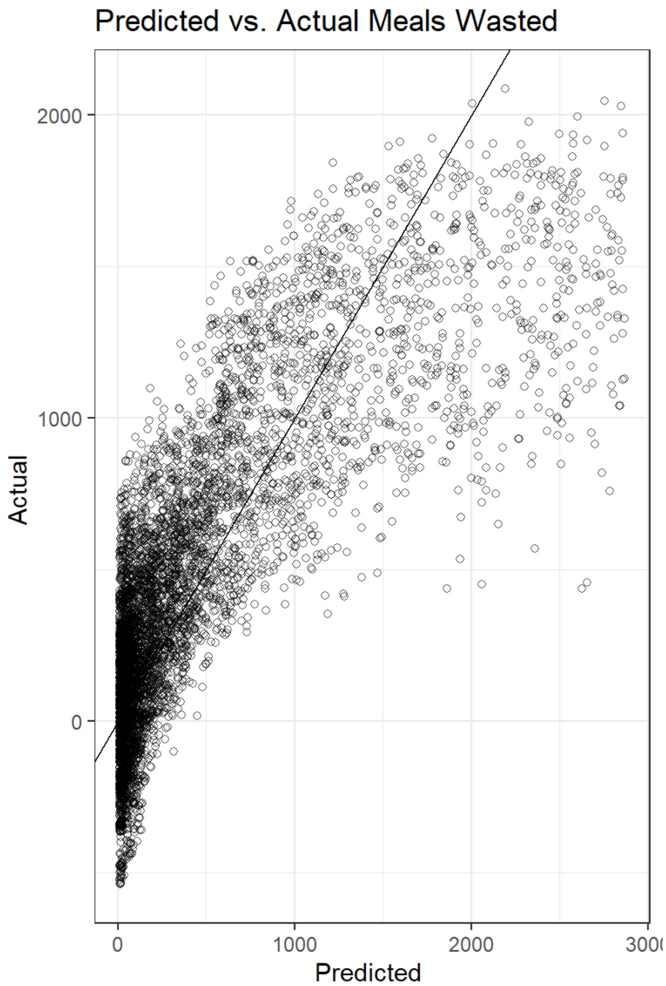
```
# Creating models by implementing the PCA
return_model_vtreat6 <- create_models(df_trt6,ncp=12)
return_model_vtreat6$rmse_val
```

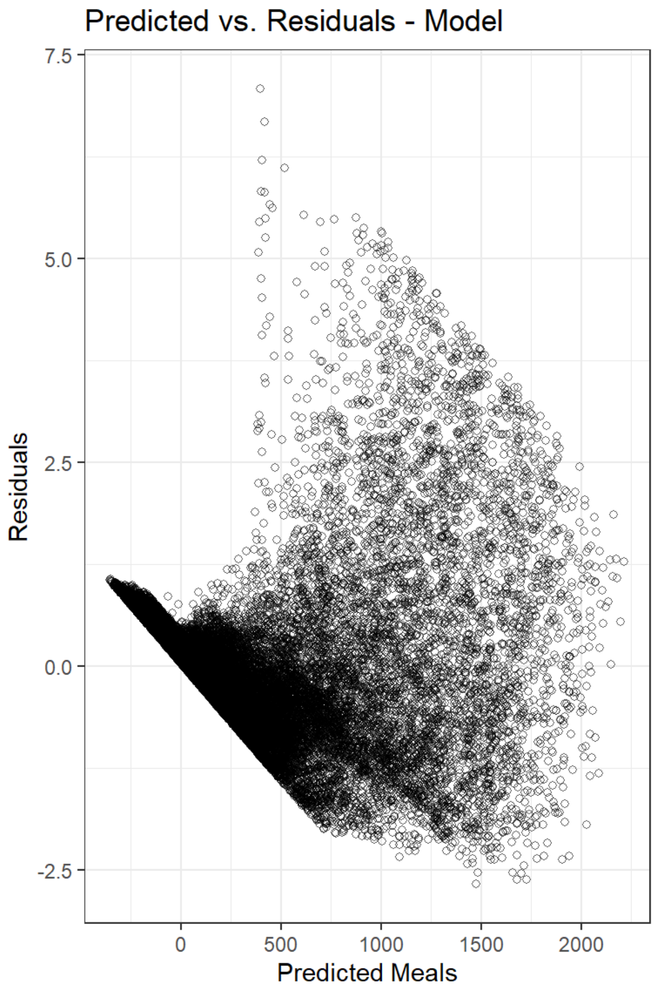
```
## [1] 368.4639
```

```
return_model_vtreat6$r_squared_val
```

```
## [1] 0.6469139
```

```
gridExtra::grid.arrange(return_model_vtreat6$plot1, return_model_vtreat6$plot2, nrow = 1)
```





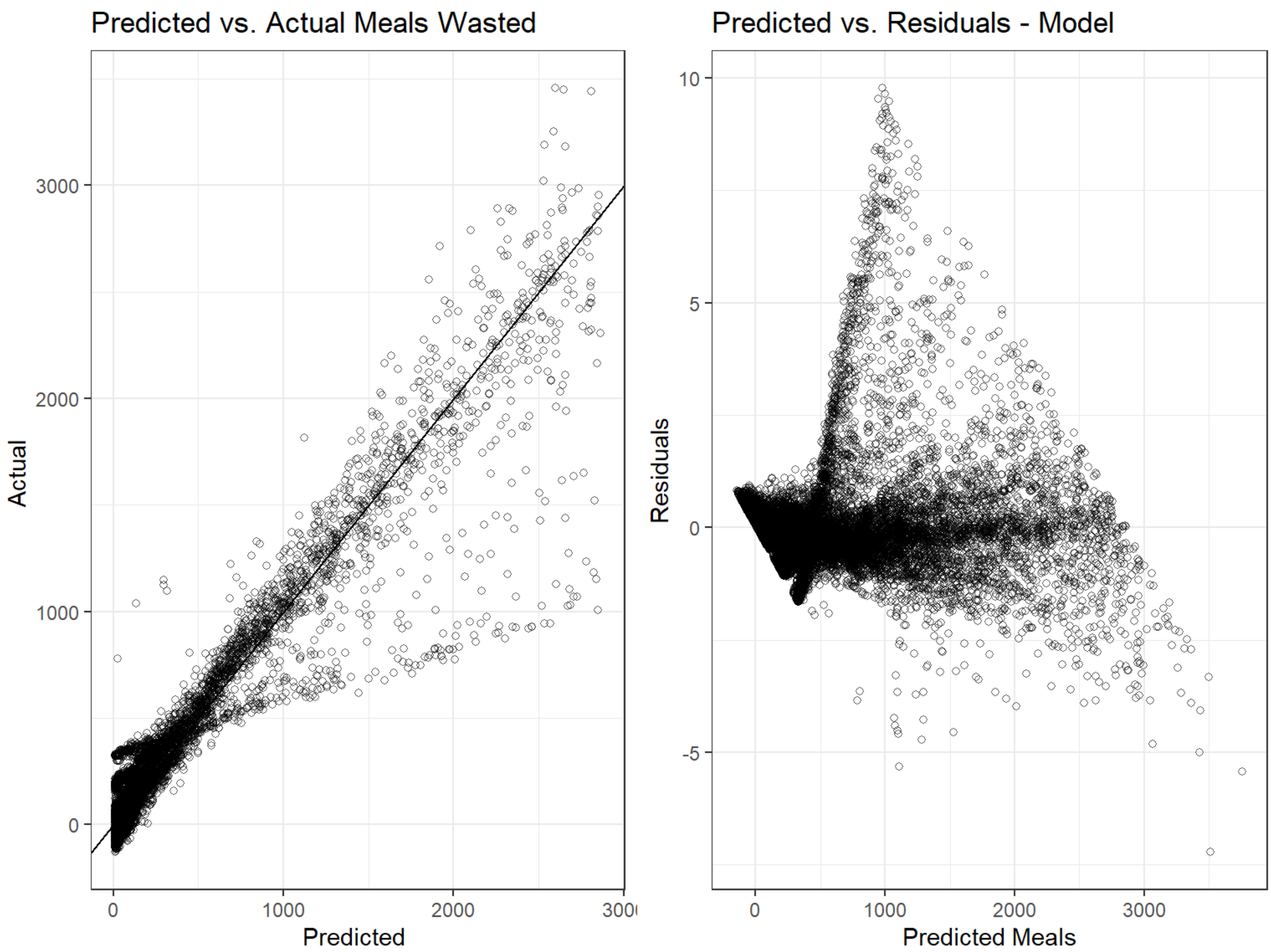
```
# Building a Regression model using Recipe Package
df_recipe7 <- create_dataset()
# Creating a Blueprint for Recipe Model
blueprint <- recipe(meals_wasted ~ ., data = df_recipe7) %>%
      step_dummy(all_nominal(), one_hot = FALSE)
prepare <- recipes::prep(blueprint, training = df_recipe7)
df_recipe7_baked <- bake(prepare, new_data = df_recipe7)
# Creating the model without PCA
return_model_recipe7 <- create_models_without_pca(df_recipe7_baked)
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Recipe Model",Rsquared=re
turn_model_recipe7$r_squared_val,RMSEmodel=return_model_recipe7$rmse_val))
return_model_recipe7$rmse_val
```

```
## [1] 195.803
```

```
return_model_recipe7$r_squared_val
```

```
## [1] 0.9015776
```

```
gridExtra::grid.arrange(return_model_recipe7$plot1, return_model_recipe7$plot2, nrow = 1)
```



Regularized Regression Models

```r
# Preparing the data for Regularized models
df_modeldata <- df_trt6
features_mat <- model.matrix(meals_wasted~., df_modeldata)
target <-  df_modeldata %>%
  select(meals_wasted) %>%
  unlist() %>%
  as.numeric()
```

Building Ridge Model

```r
# Creating a Grid of Lambda values
lambda_grid <-  10^seq(10, -2, length = 100)
# Creating Train and Test sets
train_df <-  df_modeldata %>%
  sample_frac(0.8)
test_df <-  df_modeldata %>%
  setdiff(train_df)
x_train <-  model.matrix(meals_wasted~., train_df)
x_test <-  model.matrix(meals_wasted~., test_df)
y_train <-  train_df %>%
  select(meals_wasted) %>%
  unlist() %>%
  as.numeric()
y_test <-  test_df %>%
  select(meals_wasted) %>%
  unlist() %>%
  as.numeric()
```

```r
cv_ridge <-  cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on traini
ng data
bestlam_ridge <-  cv_ridge$lambda.min  # Select lambda that minimizes training RMSE
bestlam_ridge
```

```
## [1] 50.52503
```

```r
# Creating Ridge Model with all values in the Lambda Grid
ridge_model <-  glmnet(x_train, y_train, alpha=0, lambda = lambda_grid)
# Use best lambda to predict test data
ridge_pred <-  predict(ridge_model, s = bestlam_ridge, newx = x_test)
# Calculate test RMSE
rmse_ridge_cv <- mean((ridge_pred - y_test)^2) %>% sqrt()
rmse_ridge_cv
```

```
## [1] 198.5632
```

```r
# Calculate test R squared Values
rsq_ridge_cv <- cor(ridge_pred, y_test)^2
rsq_ridge_cv
```

```
##         [,1]
## s1 0.8981801
```

```
# Appending the results to the Model Results Dataframe
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Ridge Model",Rsquared=rsq
_ridge_cv,RMSEmodel=rmse_ridge_cv))
```

Building Lasso Model

```
# Training Lasso Model on Train sets
lasso_model = glmnet(x_train,
                   y_train,
                   alpha = 1,
                   lambda = lambda_grid) # Fit lasso model on training data
# Fit lasso model on training data
cv_lasso = cv.glmnet(x_train, y_train, alpha = 1)
# Select lambda that minimizes training RMSE
bestlam_lasso =cv_lasso$lambda.min
# Use best lambda to predict test data
lasso_pred = predict(lasso_model, s = bestlam_lasso, newx = x_test)
# Calculate test RMSE
rmse_lasso_cv <- mean((lasso_pred - y_test)^2) %>% sqrt()
rmse_lasso_cv
```

```
## [1] 195.8501
```

```
# Calculate test R-squared
rsq_lasso_cv <- cor(lasso_pred, y_test)^2
rsq_lasso_cv
```

```
##         [,1]
## s1 0.8999931
```

```
# Appending the results to the Model Results Dataframe
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Lasso Model",Rsquared=rsq
_lasso_cv,RMSEmodel=rmse_lasso_cv))
```

Building Elastic Net Model

```r
# Creating X and Y datasets
X <- df_modeldata %>%
     select(meals_wasted) %>%
     scale(center = TRUE, scale = FALSE) %>%
     as.matrix()
Y <- df_modeldata %>%
    select(-meals_wasted) %>%
    as.matrix()

# Model Building : Elastic Net Regression
control <- trainControl(method = "repeatedcv",
                              number = 2,
                              repeats = 2,
                              search = "random",
                              verboseIter = TRUE)

# Training ELastic Net Regression model
elastic_model <- train(meals_wasted ~ .,
                           data = cbind(X, Y),
                           method = "glmnet",
                           preProcess = c("center", "scale"),
                           tuneLength = 25,
                           trControl = control)
```

```
## + Fold1.Rep1: alpha=0.50379, lambda=0.228205
## - Fold1.Rep1: alpha=0.50379, lambda=0.228205
## + Fold1.Rep1: alpha=0.42584, lambda=7.298406
## - Fold1.Rep1: alpha=0.42584, lambda=7.298406
## + Fold1.Rep1: alpha=0.58888, lambda=7.475775
## - Fold1.Rep1: alpha=0.58888, lambda=7.475775
## + Fold1.Rep1: alpha=0.81231, lambda=5.109245
## - Fold1.Rep1: alpha=0.81231, lambda=5.109245
## + Fold1.Rep1: alpha=0.42032, lambda=0.481028
## - Fold1.Rep1: alpha=0.42032, lambda=0.481028
## + Fold1.Rep1: alpha=0.59412, lambda=3.582145
## - Fold1.Rep1: alpha=0.59412, lambda=3.582145
## + Fold1.Rep1: alpha=0.16027, lambda=0.011730
## - Fold1.Rep1: alpha=0.16027, lambda=0.011730
## + Fold1.Rep1: alpha=0.92199, lambda=0.060467
## - Fold1.Rep1: alpha=0.92199, lambda=0.060467
## + Fold1.Rep1: alpha=0.25165, lambda=0.001111
## - Fold1.Rep1: alpha=0.25165, lambda=0.001111
## + Fold1.Rep1: alpha=0.89901, lambda=0.253633
## - Fold1.Rep1: alpha=0.89901, lambda=0.253633
## + Fold1.Rep1: alpha=0.15105, lambda=0.001894
## - Fold1.Rep1: alpha=0.15105, lambda=0.001894
## + Fold1.Rep1: alpha=0.62902, lambda=0.048747
## - Fold1.Rep1: alpha=0.62902, lambda=0.048747
## + Fold1.Rep1: alpha=0.38633, lambda=0.035336
## - Fold1.Rep1: alpha=0.38633, lambda=0.035336
## + Fold1.Rep1: alpha=0.36592, lambda=1.200832
## - Fold1.Rep1: alpha=0.36592, lambda=1.200832
## + Fold1.Rep1: alpha=0.41757, lambda=0.079175
## - Fold1.Rep1: alpha=0.41757, lambda=0.079175
## + Fold1.Rep1: alpha=0.06885, lambda=0.180278
## - Fold1.Rep1: alpha=0.06885, lambda=0.180278
## + Fold1.Rep1: alpha=0.60355, lambda=0.137023
## - Fold1.Rep1: alpha=0.60355, lambda=0.137023
## + Fold1.Rep1: alpha=0.59368, lambda=0.010212
## - Fold1.Rep1: alpha=0.59368, lambda=0.010212
## + Fold1.Rep1: alpha=0.78970, lambda=0.022890
## - Fold1.Rep1: alpha=0.78970, lambda=0.022890
## + Fold1.Rep1: alpha=0.75177, lambda=0.463799
## - Fold1.Rep1: alpha=0.75177, lambda=0.463799
## + Fold1.Rep1: alpha=0.39850, lambda=3.014910
## - Fold1.Rep1: alpha=0.39850, lambda=3.014910
## + Fold1.Rep1: alpha=0.49186, lambda=0.273147
## - Fold1.Rep1: alpha=0.49186, lambda=0.273147
## + Fold1.Rep1: alpha=0.07849, lambda=0.011496
## - Fold1.Rep1: alpha=0.07849, lambda=0.011496
## + Fold1.Rep1: alpha=0.94913, lambda=1.102175
## - Fold1.Rep1: alpha=0.94913, lambda=1.102175
## + Fold1.Rep1: alpha=0.56537, lambda=2.864272
## - Fold1.Rep1: alpha=0.56537, lambda=2.864272
## + Fold2.Rep1: alpha=0.50379, lambda=0.228205
## - Fold2.Rep1: alpha=0.50379, lambda=0.228205
## + Fold2.Rep1: alpha=0.42584, lambda=7.298406
## - Fold2.Rep1: alpha=0.42584, lambda=7.298406
```

```
## + Fold2.Rep1: alpha=0.58888, lambda=7.475775
## - Fold2.Rep1: alpha=0.58888, lambda=7.475775
## + Fold2.Rep1: alpha=0.81231, lambda=5.109245
## - Fold2.Rep1: alpha=0.81231, lambda=5.109245
## + Fold2.Rep1: alpha=0.42032, lambda=0.481028
## - Fold2.Rep1: alpha=0.42032, lambda=0.481028
## + Fold2.Rep1: alpha=0.59412, lambda=3.582145
## - Fold2.Rep1: alpha=0.59412, lambda=3.582145
## + Fold2.Rep1: alpha=0.16027, lambda=0.011730
## - Fold2.Rep1: alpha=0.16027, lambda=0.011730
## + Fold2.Rep1: alpha=0.92199, lambda=0.060467
## - Fold2.Rep1: alpha=0.92199, lambda=0.060467
## + Fold2.Rep1: alpha=0.25165, lambda=0.001111
## - Fold2.Rep1: alpha=0.25165, lambda=0.001111
## + Fold2.Rep1: alpha=0.89901, lambda=0.253633
## - Fold2.Rep1: alpha=0.89901, lambda=0.253633
## + Fold2.Rep1: alpha=0.15105, lambda=0.001894
## - Fold2.Rep1: alpha=0.15105, lambda=0.001894
## + Fold2.Rep1: alpha=0.62902, lambda=0.048747
## - Fold2.Rep1: alpha=0.62902, lambda=0.048747
## + Fold2.Rep1: alpha=0.38633, lambda=0.035336
## - Fold2.Rep1: alpha=0.38633, lambda=0.035336
## + Fold2.Rep1: alpha=0.36592, lambda=1.200832
## - Fold2.Rep1: alpha=0.36592, lambda=1.200832
## + Fold2.Rep1: alpha=0.41757, lambda=0.079175
## - Fold2.Rep1: alpha=0.41757, lambda=0.079175
## + Fold2.Rep1: alpha=0.06885, lambda=0.180278
## - Fold2.Rep1: alpha=0.06885, lambda=0.180278
## + Fold2.Rep1: alpha=0.60355, lambda=0.137023
## - Fold2.Rep1: alpha=0.60355, lambda=0.137023
## + Fold2.Rep1: alpha=0.59368, lambda=0.010212
## - Fold2.Rep1: alpha=0.59368, lambda=0.010212
## + Fold2.Rep1: alpha=0.78970, lambda=0.022890
## - Fold2.Rep1: alpha=0.78970, lambda=0.022890
## + Fold2.Rep1: alpha=0.75177, lambda=0.463799
## - Fold2.Rep1: alpha=0.75177, lambda=0.463799
## + Fold2.Rep1: alpha=0.39850, lambda=3.014910
## - Fold2.Rep1: alpha=0.39850, lambda=3.014910
## + Fold2.Rep1: alpha=0.49186, lambda=0.273147
## - Fold2.Rep1: alpha=0.49186, lambda=0.273147
## + Fold2.Rep1: alpha=0.07849, lambda=0.011496
## - Fold2.Rep1: alpha=0.07849, lambda=0.011496
## + Fold2.Rep1: alpha=0.94913, lambda=1.102175
## - Fold2.Rep1: alpha=0.94913, lambda=1.102175
## + Fold2.Rep1: alpha=0.56537, lambda=2.864272
## - Fold2.Rep1: alpha=0.56537, lambda=2.864272
## + Fold1.Rep2: alpha=0.50379, lambda=0.228205
## - Fold1.Rep2: alpha=0.50379, lambda=0.228205
## + Fold1.Rep2: alpha=0.42584, lambda=7.298406
## - Fold1.Rep2: alpha=0.42584, lambda=7.298406
## + Fold1.Rep2: alpha=0.58888, lambda=7.475775
## - Fold1.Rep2: alpha=0.58888, lambda=7.475775
## + Fold1.Rep2: alpha=0.81231, lambda=5.109245
## - Fold1.Rep2: alpha=0.81231, lambda=5.109245
## + Fold1.Rep2: alpha=0.42032, lambda=0.481028
```

```
## - Fold1.Rep2: alpha=0.42032, lambda=0.481028
## + Fold1.Rep2: alpha=0.59412, lambda=3.582145
## - Fold1.Rep2: alpha=0.59412, lambda=3.582145
## + Fold1.Rep2: alpha=0.16027, lambda=0.011730
## - Fold1.Rep2: alpha=0.16027, lambda=0.011730
## + Fold1.Rep2: alpha=0.92199, lambda=0.060467
## - Fold1.Rep2: alpha=0.92199, lambda=0.060467
## + Fold1.Rep2: alpha=0.25165, lambda=0.001111
## - Fold1.Rep2: alpha=0.25165, lambda=0.001111
## + Fold1.Rep2: alpha=0.89901, lambda=0.253633
## - Fold1.Rep2: alpha=0.89901, lambda=0.253633
## + Fold1.Rep2: alpha=0.15105, lambda=0.001894
## - Fold1.Rep2: alpha=0.15105, lambda=0.001894
## + Fold1.Rep2: alpha=0.62902, lambda=0.048747
## - Fold1.Rep2: alpha=0.62902, lambda=0.048747
## + Fold1.Rep2: alpha=0.38633, lambda=0.035336
## - Fold1.Rep2: alpha=0.38633, lambda=0.035336
## + Fold1.Rep2: alpha=0.36592, lambda=1.200832
## - Fold1.Rep2: alpha=0.36592, lambda=1.200832
## + Fold1.Rep2: alpha=0.41757, lambda=0.079175
## - Fold1.Rep2: alpha=0.41757, lambda=0.079175
## + Fold1.Rep2: alpha=0.06885, lambda=0.180278
## - Fold1.Rep2: alpha=0.06885, lambda=0.180278
## + Fold1.Rep2: alpha=0.60355, lambda=0.137023
## - Fold1.Rep2: alpha=0.60355, lambda=0.137023
## + Fold1.Rep2: alpha=0.59368, lambda=0.010212
## - Fold1.Rep2: alpha=0.59368, lambda=0.010212
## + Fold1.Rep2: alpha=0.78970, lambda=0.022890
## - Fold1.Rep2: alpha=0.78970, lambda=0.022890
## + Fold1.Rep2: alpha=0.75177, lambda=0.463799
## - Fold1.Rep2: alpha=0.75177, lambda=0.463799
## + Fold1.Rep2: alpha=0.39850, lambda=3.014910
## - Fold1.Rep2: alpha=0.39850, lambda=3.014910
## + Fold1.Rep2: alpha=0.49186, lambda=0.273147
## - Fold1.Rep2: alpha=0.49186, lambda=0.273147
## + Fold1.Rep2: alpha=0.07849, lambda=0.011496
## - Fold1.Rep2: alpha=0.07849, lambda=0.011496
## + Fold1.Rep2: alpha=0.94913, lambda=1.102175
## - Fold1.Rep2: alpha=0.94913, lambda=1.102175
## + Fold1.Rep2: alpha=0.56537, lambda=2.864272
## - Fold1.Rep2: alpha=0.56537, lambda=2.864272
## + Fold2.Rep2: alpha=0.50379, lambda=0.228205
## - Fold2.Rep2: alpha=0.50379, lambda=0.228205
## + Fold2.Rep2: alpha=0.42584, lambda=7.298406
## - Fold2.Rep2: alpha=0.42584, lambda=7.298406
## + Fold2.Rep2: alpha=0.58888, lambda=7.475775
## - Fold2.Rep2: alpha=0.58888, lambda=7.475775
## + Fold2.Rep2: alpha=0.81231, lambda=5.109245
## - Fold2.Rep2: alpha=0.81231, lambda=5.109245
## + Fold2.Rep2: alpha=0.42032, lambda=0.481028
## - Fold2.Rep2: alpha=0.42032, lambda=0.481028
## + Fold2.Rep2: alpha=0.59412, lambda=3.582145
## - Fold2.Rep2: alpha=0.59412, lambda=3.582145
## + Fold2.Rep2: alpha=0.16027, lambda=0.011730
## - Fold2.Rep2: alpha=0.16027, lambda=0.011730
```

```
## + Fold2.Rep2: alpha=0.92199, lambda=0.060467
## - Fold2.Rep2: alpha=0.92199, lambda=0.060467
## + Fold2.Rep2: alpha=0.25165, lambda=0.001111
## - Fold2.Rep2: alpha=0.25165, lambda=0.001111
## + Fold2.Rep2: alpha=0.89901, lambda=0.253633
## - Fold2.Rep2: alpha=0.89901, lambda=0.253633
## + Fold2.Rep2: alpha=0.15105, lambda=0.001894
## - Fold2.Rep2: alpha=0.15105, lambda=0.001894
## + Fold2.Rep2: alpha=0.62902, lambda=0.048747
## - Fold2.Rep2: alpha=0.62902, lambda=0.048747
## + Fold2.Rep2: alpha=0.38633, lambda=0.035336
## - Fold2.Rep2: alpha=0.38633, lambda=0.035336
## + Fold2.Rep2: alpha=0.36592, lambda=1.200832
## - Fold2.Rep2: alpha=0.36592, lambda=1.200832
## + Fold2.Rep2: alpha=0.41757, lambda=0.079175
## - Fold2.Rep2: alpha=0.41757, lambda=0.079175
## + Fold2.Rep2: alpha=0.06885, lambda=0.180278
## - Fold2.Rep2: alpha=0.06885, lambda=0.180278
## + Fold2.Rep2: alpha=0.60355, lambda=0.137023
## - Fold2.Rep2: alpha=0.60355, lambda=0.137023
## + Fold2.Rep2: alpha=0.59368, lambda=0.010212
## - Fold2.Rep2: alpha=0.59368, lambda=0.010212
## + Fold2.Rep2: alpha=0.78970, lambda=0.022890
## - Fold2.Rep2: alpha=0.78970, lambda=0.022890
## + Fold2.Rep2: alpha=0.75177, lambda=0.463799
## - Fold2.Rep2: alpha=0.75177, lambda=0.463799
## + Fold2.Rep2: alpha=0.39850, lambda=3.014910
## - Fold2.Rep2: alpha=0.39850, lambda=3.014910
## + Fold2.Rep2: alpha=0.49186, lambda=0.273147
## - Fold2.Rep2: alpha=0.49186, lambda=0.273147
## + Fold2.Rep2: alpha=0.07849, lambda=0.011496
## - Fold2.Rep2: alpha=0.07849, lambda=0.011496
## + Fold2.Rep2: alpha=0.94913, lambda=1.102175
## - Fold2.Rep2: alpha=0.94913, lambda=1.102175
## + Fold2.Rep2: alpha=0.56537, lambda=2.864272
## - Fold2.Rep2: alpha=0.56537, lambda=2.864272
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0.16, lambda = 0.0117 on full training set
```

```
elastic_model
```

```
## glmnet
##
## 30264 samples
##    40 predictor
##
## Pre-processing: centered (40), scaled (40)
## Resampling: Cross-Validated (2 fold, repeated 2 times)
## Summary of sample sizes: 15131, 15133, 15131, 15133
## Resampling results across tuning parameters:
##
##   alpha       lambda       RMSE      Rsquared   MAE
##   0.06885198  0.180277603  194.6018  0.9018846  110.6066
##   0.07849086  0.011495942  194.6031  0.9018828  110.6041
##   0.15104618  0.001894020  194.6185  0.9018676  110.5933
##   0.16026847  0.011729730  194.6002  0.9018832  110.5970
##   0.25165472  0.001111041  194.6164  0.9018678  110.6082
##   0.36591734  1.200831938  194.6306  0.9018543  110.5768
##   0.38632511  0.035336109  194.6079  0.9018751  110.6256
##   0.39850191  3.014909681  194.9244  0.9015838  110.1523
##   0.41757434  0.079175085  194.6132  0.9018698  110.6315
##   0.42032481  0.481027770  194.6121  0.9018707  110.6274
##   0.42583568  7.298405700  195.4435  0.9011927  108.6643
##   0.49186011  0.273147340  194.6117  0.9018710  110.6405
##   0.50379380  0.228205150  194.6065  0.9018751  110.6217
##   0.56536662  2.864271614  195.0014  0.9015166  109.9332
##   0.58888431  7.475775289  195.7707  0.9009489  107.5987
##   0.59368461  0.010211728  194.6115  0.9018708  110.6308
##   0.59412418  3.582144705  195.1108  0.9014290  109.5735
##   0.60354874  0.137023250  194.6109  0.9018713  110.6313
##   0.62901864  0.048746997  194.6120  0.9018704  110.6378
##   0.75176617  0.463799391  194.6134  0.9018686  110.6299
##   0.78970250  0.022890498  194.6136  0.9018683  110.6302
##   0.81230865  5.109245413  195.6447  0.9010095  107.9501
##   0.89901200  0.253632761  194.6064  0.9018754  110.6061
##   0.92199062  0.060467489  194.6084  0.9018739  110.6177
##   0.94912640  1.102174663  194.8070  0.9016855  110.3208
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1602685 and lambda
##  = 0.01172973.
```

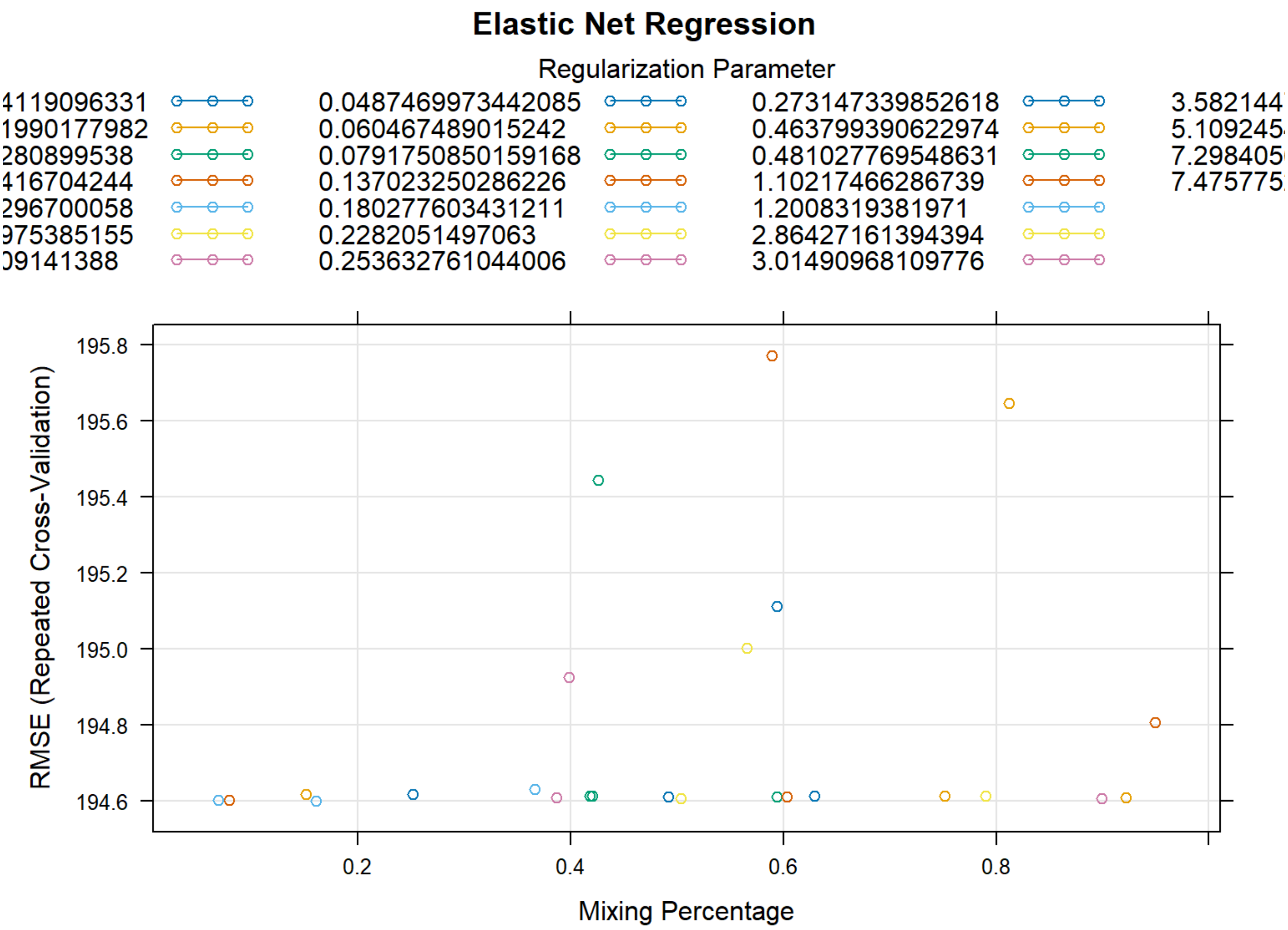
```
# Model Prediction
x_hat_pre <- predict(elastic_model, Y)


# Multiple R-squared
rsq_elastic_net <- cor(X, x_hat_pre)^2
rsq_elastic_net
```

```
##                 [,1]
## meals_wasted 0.90253
```

```
# Calculating the RMSE
rmse_elastic_net <- mean(elastic_model$resample$RMSE)
rmse_elastic_net
```

```
## [1] 194.6002
```

```
# Appending the results to the Model Results Dataframe
model_results_df <- rbind(model_results_df,list(ncp=0,modelname="Elastic Net Model",Rsquar
ed=rsq_elastic_net,RMSEmodel=rmse_elastic_net))
# Ploting the model results
plot(elastic_model, main = "Elastic Net Regression")
```




```
# Displaying all the Model results in the Order of Highest Rsquared values.
model_results_df <- model_results_df %>% select(modelname, Rsquared,RMSEmodel) %>% arrange
(desc(Rsquared))
kbl(model_results_df,caption = "Regression Model results", booktabs = T) %>% kable_styling
(latex_options = c("striped", "hold_position"))
```

Regression Model results

modelname	Rsquared	RMSEmodel
Simple Model	0.9057294	191.5152

modelname	Rsquared	RMSEmodel
Caret Model	0.9056819	190.5132
Elastic Net Model	0.9025300	194.6002
Recipe Model	0.9015776	195.8030
Lasso Model	0.8999931	195.8501
vTreat Model	0.8994878	197.5938
Ridge Model	0.8981801	198.5632