BAN440 - Term Paper Code

Candidate numbers: 74, 79, 85

Table of contents

Packages used

```
# relevant libraries
library(tidyverse)       # For data manipulation
library(readxl)          # For reading Excel files
library(fastDummies)     # For creating dummy variables
library(knitr)           # For creating tables
library(stargazer)       # For creating regression tables
library(caret)           # For data splitting and model evaluation
library(here)            # For file path management
library(httr)            # For API requests
library(jsonlite)        # For JSON parsing
library(readr)           # For reading CSV files
library(stringr)         # For string manipulation
library(tidyr)           # For unnesting
library(writexl)         # For writing Excel files
library(geosphere)       # For distance calculations
library(caret)           # For data splitting and model evaluation
library(kableExtra)      # For creating tables
library(tidymodels)      # For machine learning
library(sf)              # For spatial data handling
library(csmaps)          # For spatial data visualization
```

Data

Vinmonopolet API

Detailed descriptions of each Vinmonopolet store per 2024

```
# Define API URL
url <- "https://apis.vinmonopolet.no/stores/v0/details"

# Define your subscription key (replace with your actual key)
subscription_key <- "3b5b02c6793240fe9e6cb6d176e110e0"

# Send GET request with subscription key in header
response <- GET(url,
                add_headers(
                  Accept = "application/json",
                  `Ocp-Apim-Subscription-Key` = subscription_key  # API authentication
                ))


# Check response status
if (status_code(response) == 200) {
  # Convert API response to JSON and store it
  data <- content(response, as = "text", encoding = "UTF-8")
  store_data <- fromJSON(data)

  # View first few rows
  print(head(store_data))
} else {
  print(paste("Error:", status_code(response)))
}


# --------------------------- Combine API with Vinmonopol Data ---------------------------


# Ensure store_data_clean is correctly formatted
store_data_clean <- store_data %>%
  unnest_wider(address) %>%  # Expands nested address fields
  select(
    storeId,
    storeName,
    status,
    postalCode,
    city,
    gpsCoord
  ) %>%
  rename(
    Store_ID = storeId,
    Store_Name = storeName,
    Store_Status = status,
    Postal_Code = postalCode,
    City = city,
    GPS_Coordinates = gpsCoord
  )
```

```
# Transforming to normal characters
store_data_clean$Store_Name <- iconv(store_data_clean$Store_Name, from = "UTF-8", to = "ASC
store_data_clean$Store_Name <- trimws(store_data_clean$Store_Name)
```

Dimpostnummer merge

As stores have postal codes instead of municipality codes, we need a merge data set inbetween

```
# Read the Dimpostnummer data
dimpostnummer_xlsx <- here("Data", "Vinmonopolet", "dimpostnummer.xlsx")

dimpostnummer_data <- read_excel(dimpostnummer_xlsx) %>%
  select("Postnummer", "Poststed", "Fylke", "KommuneKode", "Kommune")

# Merge with store_data_clean
store_data_clean <- store_data_clean %>%
  left_join(dimpostnummer_data, by = c("Postal_Code" = "Postnummer"))
```

Vinmonopolet 2024

The “foundation” with store names and sales data for 2024

```
# Set locale to UTF-8
Sys.setlocale("LC_ALL", "en_US.UTF-8")

# Use here package to define the working directory
Vinmonopolet_2024 <- here("Data", "Vinmonopolet", "Vinmonopolet_2024.xlsx")

# Get the names of all sheets in the Excel file
sheet_names <- excel_sheets(Vinmonopolet_2024)

# Read each sheet into a list of data frames, skipping the first row
list_of_dfs <- lapply(sheet_names, function(sheet) {
  read_excel(Vinmonopolet_2024, sheet = sheet, skip = 2)
})

# Combine all data frames into a single data frame
```

```
combined_data <- bind_rows(list_of_dfs)

# View the combined data frame
print(combined_data)

# Unique values in the first column
unique_values <- unique(combined_data$...1)

print(unique_values)

# Transforming to normal characters
combined_data$...1 <- iconv(combined_data$...1, from = "UTF-8", to = "ASCII//TRANSLIT")
combined_data$...1 <- trimws(combined_data$...1)

# Define the values to filter out
values_to_exclude <- c(
  "Svakvin", "Rodvin", "Hvitvin", "Musserende vin", "Rosevin",
  "Perlende vin", "Aromatisert vin", "Sider", "Fruktvin",
  "Brennevin", "Vodka", "Likor", "Whisky", "Akevitt",
  "Brennevin, annet", "Gin", "Druebrennevin",
  "Brennevin, noytralt < 37,5 %", "Rom", "Bitter",
  "Fruktbrennevin", "Genever", "Ol", "Alkoholfritt", "Sterkvin", "Totalsum",
  "eLager"
)

# Column names of combined data
colnames(combined_data)

# Filter out the specified values from the first column
filtered_data <- combined_data %>%
  mutate("2024" = as.numeric(`2024`),
         "Store" = as.character(`...1`)) %>%
  filter(!.[[1]] %in% values_to_exclude) %>%
  select("Store", "2024")

# Export the filtered data to an Excel file
#write_xlsx(filtered_data, "filtered_data.xlsx")

# Standardize store names to improve matching
filtered_data <- filtered_data %>%
  mutate(Store = str_trim(str_to_lower(Store)))  # Trim spaces and convert to lowercase

store_data_clean <- store_data_clean %>%
  mutate(Store_Name = str_trim(str_to_lower(Store_Name)))  # Trim spaces and convert to low

# Remove unwanted characters from store names
store_data_clean <- store_data_clean %>%
  mutate(Store_Name = case_when(
    Store_Name == "oslo, thereses gate (stengt ja" ~ "oslo, thereses gate",
    Store_Name == "sandnes, sentrum" ~ "sandnes sentrum",
    Store_Name == "buvika" ~ "buvika, apent 24. oktober",
    Store_Name == "sola, tananger" ~ "sola, tananger, apnet 3. oktober",
    Store_Name == "oslo, bjorvika" ~ "oslo, bjorvika, apnet 14. mars 2024",
    Store_Name == "melhus" ~ "melhus, butikken stengt i 2023 pga kranvelt",
    Store_Name == "bergen, valkendorfsgt." ~ "bergen, valkendorfsgate",
    TRUE ~ Store_Name  # This keeps all other values unchanged
  ))

# Merge filtered_data (sales) with store_data_clean (store details)
final_data <- filtered_data %>%
  left_join(store_data_clean, by = c("Store" = "Store_Name"))  # Match by store name

# Check merged data
head(final_data)

# Write to excel
#write_xlsx(final_data, "final_data.xlsx")
```

Kommuneendringer 2017

```
### Kommuneendringer 2017 ###

data_df <- final_data %>%
  rename(
    Municipality_Code = KommuneKode,
    Municipality_Name = Kommune
  )
```

```r
Kommuneendringer_17_xlsx <- here("Data", "Vinmonopolet", "Kommuneendringer_17.xlsx")

kommuneendringer_df <- read_excel(Kommuneendringer_17_xlsx)

# Clean up column names by using the correct column names
colnames(kommuneendringer_df) <- c("New_Codes", "Old_Codes")

# Split old municipality numbers into separate elements if they are separated by spaces
kommuneendringer_df$Old_Codes <- str_split(kommuneendringer_df$Old_Codes, " ")

# Extract the first four digits from each element in Old_Codes
old_codes_numeric <- lapply(kommuneendringer_df$Old_Codes, function(x) substr(x, 1, 4))

# Create a lookup list that maps old municipality codes to new codes
kommune_mapping <- setNames(rep(kommuneendringer_df$New_Codes, times = sapply(old_codes_num
                            unlist(old_codes_numeric))

# Update Municipality_Code and Municipality_Name in data_df
data_df <- data_df %>%
  rowwise() %>%
  mutate(
    new_val = if (Municipality_Code %in% names(kommune_mapping)) kommune_mapping[[Municipal
    Municipality_Code = if (!is.na(new_val)) substr(new_val, 1, 4) else Municipality_Code,
    Municipality_Name = if (!is.na(new_val)) {
      # Remove the municipality number and hyphen from the new value to get the municipalit
      str_trim(str_remove(new_val, "^[0-9]{4}\\s*-\\s*"))
    } else {
      Municipality_Name
    }
  ) %>%
  ungroup() %>%
  select(-new_val)

# Save the updated data to a new Excel file
#write_xlsx(data_df, "final_data_17.xlsx")
```

Kommuneendringer 2018

```
# Read in data from Excel files
Kommuneendringer_18_xlsx <- here("Data", "Vinmonopolet", "Kommuneendringer_18.xlsx")

kommuneendringer_df <- read_excel(Kommuneendringer_18_xlsx)

# Clean up column names by using the correct column names
colnames(kommuneendringer_df) <- c("New_Code", "Old_Codes")

# Split old municipality numbers (in case multiple old municipalities are separated by spac
kommuneendringer_df$Old_Codes <- str_split(kommuneendringer_df$Old_Codes, " ")

# Create a lookup list for old codes to new codes (one-way mapping)
kommune_mapping <- setNames(rep(kommuneendringer_df$New_Code, times = sapply(kommuneendring
                            unlist(kommuneendringer_df$Old_Codes))

# Update both Municipality_Code and Municipality_Name in data_df
data_df <- data_df %>%
  rowwise() %>%
  mutate(
    new_val = if (Municipality_Code %in% names(kommune_mapping)) kommune_mapping[[Municipal
    Municipality_Code = if (!is.na(new_val)) substr(new_val, 1, 4) else Municipality_Code,
    Municipality_Name = if (!is.na(new_val)) str_trim(str_remove(new_val, "^[0-9]{4}\\s*-\\
  ) %>%
  ungroup() %>%
  select(-new_val)

# Save the updated file
#write_xlsx(data_df, "final_data_18.xlsx")
```

Kommuneendringer 2020

```
# Read in data from Excel files
Kommuneendringer_20_xlsx <- here("Data", "Vinmonopolet", "Kommuneendringer_20.xlsx")

kommuneendringer_df <- read_excel(Kommuneendringer_20_xlsx)
```

```r
# Clean up column names by using the correct column names
colnames(kommuneendringer_df) <- c("New_Code", "Old_Codes")

# Split old municipality numbers (in case multiple old municipalities are separated by spac
kommuneendringer_df$Old_Codes <- str_split(kommuneendringer_df$Old_Codes, " ")

# Create a lookup list for old codes to new codes (one-way mapping)
kommune_mapping <- setNames(rep(kommuneendringer_df$New_Code, times = sapply(kommuneendring
                            unlist(kommuneendringer_df$Old_Codes))

# Update both Municipality_Code and Municipality_Name in data_df
data_df <- data_df %>%
  rowwise() %>%
  mutate(
    new_val = if (Municipality_Code %in% names(kommune_mapping)) kommune_mapping[[Municipal
    Municipality_Code = if (!is.na(new_val)) substr(new_val, 1, 4) else Municipality_Code,
    Municipality_Name = if (!is.na(new_val)) str_trim(str_remove(new_val, "^[0-9]{4}\\s*-\\
  ) %>%
  ungroup() %>%
  select(-new_val)

# Save the updated file
#write_xlsx(data_df, "final_data_20.xlsx")
```

Kommuneendringer 2024

```r
# Read in data from Excel files
Kommuneendringer_24_xlsx <- here("Data", "Vinmonopolet", "Kommuneendringer_24.xlsx")

kommuneendringer_df <- read_excel(Kommuneendringer_24_xlsx)


# Clean up column names by using the correct column names
colnames(kommuneendringer_df) <- c("New_Code", "Old_Codes")

# Split old municipality numbers (in case multiple old municipalities are separated by spac
kommuneendringer_df$Old_Codes <- str_split(kommuneendringer_df$Old_Codes, " ")
```

```
# Create a lookup list for old codes to new codes (one-way mapping)
kommune_mapping <- setNames(rep(kommuneendringer_df$New_Code,
                                times = sapply(kommuneendringer_df$Old_Codes, length)),
                            unlist(kommuneendringer_df$Old_Codes))

# Update both Municipality_Code and Municipality_Name in data_df
data_df <- data_df %>%
  rowwise() %>%
  mutate(
    new_val = if (Municipality_Code %in% names(kommune_mapping)) kommune_mapping[[Municipal
    Municipality_Code = if (!is.na(new_val)) substr(new_val, 1, 4) else Municipality_Code,
    Municipality_Name = if (!is.na(new_val)) str_trim(str_remove(new_val, "^[0-9]{4}\\s*-\\
  ) %>%
  ungroup() %>%
  select(-new_val)

# Hardcode row 121 to set "Municipality_Code" to 1580 and "Municipality_Name" to Haram
data_df[121, "Municipality_Code"] <- "1580"
data_df[121, "Municipality_Name"] <- "Haram"

# Save the updated file
#write_xlsx(data_df, "final_data_24.xlsx")
```

Kommune 2025

Municipality data, including municipality number, population and Area

```
# Kommune data file path
Kommune_data_xlsx <- here("Data", "Vinmonopolet", "Kommune_data.xlsx")

# Read data for total population and area of each municipality
kommune_data <- read_excel(Kommune_data_xlsx, skip = 3) %>%
  rename("Municipality" = "...1",
         "Population" = "2025...2",
         "Area" = "2025...3") %>%
  separate(Municipality, into = c("Mun_num", "Mun_name"), sep = " ", extra = "merge", fill
  filter(Population != 0,
         Area != 0) %>%
  mutate(Population = as.numeric(Population),
         Area = as.numeric(Area))

# Demographic data file path
Kommune_demo_xlsx <- here("Data", "Vinmonopolet", "Kommune_demo.xlsx")

# Read data for demographic data
demographic_data <- read_excel(Kommune_demo_xlsx, skip = 4) %>%
  rename("Municipality" = "...1",
         "0-17" = "0-17 år",
         "18+" = "18 år eller eldre") %>%
  filter(if_all(everything(), ~ !is.na(.) & . != 0)) %>% # Remove rows with NA or 0 in any
  separate(Municipality, into = c("Mun_num", "Mun_name"), sep = " ", extra = "merge", fill
  separate(Mun_num, into = c("K", "Mun_num"), sep = "-") %>%
  select(-"K",
         -"Mun_name")

# Merge the two datasets
kommune_data_final <- kommune_data %>%
  left_join(demographic_data, by = c("Mun_num"))

# Write data to Excel
#write_xlsx(kommune_data_final, "Kommune_data_final.xlsx")
```

Demography data

```
final_data <- data_df

# Transforming to normal characters
final_data$Municipality_Name <- iconv(final_data$Municipality_Name, from = "UTF-8", to = "A

final_data$Municipality_Name <- trimws(final_data$Municipality_Name)

# Standardize store names to improve matching
final_data <- final_data %>%
  mutate(Municipality_Name = str_trim(str_to_lower(Municipality_Name)))  # Trim spaces and

# Loading the kommune data
kommune_data <- kommune_data_final
```

```
# Standardize the kommune data
kommune_data <- kommune_data %>%
  mutate(Mun_name = iconv(Mun_name, from = "UTF-8", to = "ASCII//TRANSLIT"),
         Mun_name = str_trim(str_to_lower(Mun_name)))  # Trim spaces and convert to lowerca

# Perform a full join to include all rows from both datasets
merged_data <- final_data %>%
  full_join(kommune_data, by = c("Municipality_Code" = "Mun_num"))

# Replace NA values in store-related columns with 0
# Assuming 'Store_Info_Column' is the column in final_data that contains store information
# Replace 'Store_Info_Column' with the actual column names you want to fill with 0
merged_data <- merged_data %>%
  mutate(across(where(is.numeric), ~ replace_na(.x, 0)))

# If you have specific columns to replace NA with 0, you can specify them like this:
# merged_data <- merged_data %>%
#   mutate(Store_Info_Column = replace_na(Store_Info_Column, 0))

# Write the merged data to an Excel file
#write_xlsx(merged_data, "final_data_mun.xlsx")
```

Distance data

This is just our code for the calculation of dist_nearest. As the actual data file is too large to submit, we jump to the next step with the resulting data saved as "final_data_mun_dist.xlsx"

```
# -----------------------------
# 1. Load and Prepare Data
# -----------------------------
# Load Vinmonopolet + municipality dataset
#data <- read_excel("final_data_mun.xlsx")

# Load pre-cleaned municipality admin center coordinates
#admin_centers_final <- readRDS("admin_centers_final.rds")

# Ensure join columns match in type
#data <- data %>%
#  mutate(Municipality_Code = as.character(Municipality_Code))

#admin_centers_final <- admin_centers_final %>%
#  mutate(kommunenummer = as.character(kommunenummer))

# -----------------------------
# 2. Merge Coordinates
# -----------------------------
# Merge admin center lat/lon into dataset by municipality
#data <- left_join(data, admin_centers_final, by = c("Municipality_Code" = "kommunenummer")

# Overwrite old coordinates with admin center coordinates
#data <- data %>%
#  mutate(
#    Latitude = as.numeric(lat),
#    Longitude = as.numeric(lon)
#  )

# -----------------------------
# 3. Parse Store GPS Coordinates
# -----------------------------
# Split store GPS into separate numeric lat/lon

# ---------------------------
# STEP 1: Load dataset
# ---------------------------

# Read merged dataset with both Vinmonopolet store info and municipality info
#data <- read_excel("final_data_mun.xlsx")

# ---------------------------
# STEP 2: Parse store coordinates
# ---------------------------

# GPS_Coordinates column contains both latitude and longitude as a string separated by ";"
# We split this into two separate numeric columns: store_lat and store_lon

#data <- data %>%
#  separate(GPS_Coordinates, into = c("store_lat", "store_lon"), sep = ";", convert = TRUE)
#  mutate(
#    store_lat = as.numeric(store_lat),   # ensure store latitude is numeric
#    store_lon = as.numeric(store_lon)    # ensure store longitude is numeric
#  )


# -----------------------------
# 4. Build Store Location Matrix
# -----------------------------
# Extract distinct (lon, lat) of all Vinmonopolet stores
#store_locations <- data %>%
#  filter(!is.na(store_lon), !is.na(store_lat)) %>%

# ---------------------------
# STEP 3: Ensure municipality center coordinates are numeric
# ---------------------------

# These are already separate in the dataset, but stored as characters - we convert them
#data <- data %>%
#  mutate(
#    Longitude = as.numeric(Longitude),  # longitude of the municipality center
#    Latitude = as.numeric(Latitude)     # latitude of the municipality center
#  )

# ---------------------------
# STEP 4: Extract store coordinates for distance calculation
# ---------------------------

# We only want to use valid store locations for calculating distances
# (some rows in the dataset are just municipality data with no store info)
#store_data <- data %>%
#  filter(!is.na(store_lat), !is.na(store_lon))

# Extract a unique matrix of all Vinmonopolet store locations
# Format required by geosphere is matrix of (longitude, latitude)
#store_locations <- store_data %>%

#  select(store_lon, store_lat) %>%
#  distinct() %>%
#  as.matrix()
# ---------------------------
# STEP 5: Define function to calculate distance to nearest store
# ---------------------------

# For a given municipality center (lon, lat), compute distance to nearest store
# Uses Haversine formula (accounts for Earth's curvature)
#min_distance_to_store <- function(lon, lat) {
#  if (is.na(lon) || is.na(lat)) {
#    return(NA)  # return NA if municipality coordinates are missing
#  }
#  muni_coord <- matrix(c(lon, lat), nrow = 1)  # convert to matrix format for geosphere
#  dists <- distHaversine(muni_coord, store_locations)  # distances in meters
#  return(min(dists) / 1000)  # convert to kilometers
#}

# ---------------------------
# STEP 6: Apply distance function to each municipality
# ---------------------------

# For each row (i.e., each municipality center), calculate distance to closest Vinmonopolet
# Note: This includes all rows (even ones without a store)

#data$dist_nearest_store <- mapply(
#  min_distance_to_store,
#  data$Longitude,
#  data$Latitude
#)


# ---------------------------
# STEP 7: Quick check (optional)
# ---------------------------

# Check that coordinates are numeric
#str(data$Longitude)
#str(data$Latitude)


# -----------------------------
# 7. Optional: Drop Redundant Columns
# -----------------------------
#data <- data %>%
#  select(
#    -lat, -lon, -multikurve, -kommunenavn
#  )

# -----------------------------
# 8. Final Checks (Optional)
# -----------------------------
#str(data$dist_nearest_store)
#summary(data$dist_nearest_store)

# -----------------------------
# 9. Does VInmonopolets 30 km threshold 97% goal work based on our data
# -----------------------------
# 1. Total population (all municipalities)
#total_pop <- sum(data$Population, na.rm = TRUE)

# 2. Population in municipalities with distance > 30 km
#pop_far_away <- data %>%
#  filter(dist_nearest_store > 30) %>%
#  summarise(total = sum(Population, na.rm = TRUE)) %>%
#  pull(total)

# 3. Share of population far away
#share_far_away <- pop_far_away / total_pop

# 4. Share WITH access (within 30 km)
#share_within_30km <- 1 - share_far_away

# 5. Print results
#cat(sprintf("Share of population within 30 km of a Vinmonopolet: %.2f%%\n", #share_within_
#cat(sprintf("Target (Vinmonopolet): 97%%\n"))

#underserved <- data %>%
#  filter(dist_nearest_store > 30) %>%
#  select(,Mun_name, Population, dist_nearest_store) %>%
#  arrange(desc(dist_nearest_store))

#print(underserved, n = 50)
```

```
# ------------------------------
# 10. Export the final data to an Excel file
# ------------------------------

#library(writexl)
#write_xlsx(data, "final_data_mun_dist.xlsx")
# ------------------------------
```

Model variables merge

```
### Independent variables merge ###

final_data_mun_dist <- here("Data", "Vinmonopolet", "final_data_mun_dist.xlsx")

# Load data
Vinmonopolet <- read_excel(final_data_mun_dist) %>%
  select(-c(Store_ID, Store_Status, Postal_Code, Poststed,
            PostnummerKategoriKode, PostnummerKategori, Region_Code,
            Municipality_Name)) %>%
  mutate(
    Municipality_Name = Mun_name,
    Region_Name = case_when(
      Region_Name == "AUST-AGDER" ~ "Agder",
      Region_Name == "VEST-AGDER" ~ "Agder",
      Region_Name == "AKERSHUS" ~ "Akershus",
      Region_Name == "OPPLAND" ~ "Innlandet",
      Region_Name == "BUSKERUD" ~ "Buskerud",
      Region_Name == "VESTFOLD" ~ "Vestfold",
      Region_Name == "FINNMARK" ~ "Finnmark",
      Region_Name == "HEDMARK" ~ "Innlandet",
      Region_Name == "MØRE OG ROMSDAL" ~ "Møre og Romsdal",
      Region_Name == "NORDLAND" ~ "Nordland",
      Region_Name == "OSLO" ~ "Oslo",
      Region_Name == "ROGALAND" ~ "Rogaland",
      Region_Name == "TELEMARK" ~ "Telemark",
      Region_Name == "TROMS" ~ "Troms",
      Region_Name == "SØR-TRØNDELAG" ~ "Trøndelag",
      Region_Name == "NORD-TRØNDELAG" ~ "Trøndelag",
```

```r
      Region_Name == "SOGN OG FJORDANE" ~ "Vestland",
      Region_Name == "HORDALAND" ~ "Vestland",
      Region_Name == "ØSTFOLD" ~ "Østfold",
      is.na(Region_Name) & str_starts(Municipality_Code, "03") ~ "Oslo",
      is.na(Region_Name) & str_starts(Municipality_Code, "11") ~ "Rogaland",
      is.na(Region_Name) & str_starts(Municipality_Code, "15") ~ "Møre og Romsdal",
      is.na(Region_Name) & str_starts(Municipality_Code, "18") ~ "Nordland",
      is.na(Region_Name) & str_starts(Municipality_Code, "31") ~ "Østfold",
      is.na(Region_Name) & str_starts(Municipality_Code, "32") ~ "Akershus",
      is.na(Region_Name) & str_starts(Municipality_Code, "33") ~ "Buskerud",
      is.na(Region_Name) & str_starts(Municipality_Code, "34") ~ "Innlandet",
      is.na(Region_Name) & str_starts(Municipality_Code, "39") ~ "Vestfold",
      is.na(Region_Name) & str_starts(Municipality_Code, "40") ~ "Telemark",
      is.na(Region_Name) & str_starts(Municipality_Code, "42") ~ "Agder",
      is.na(Region_Name) & str_starts(Municipality_Code, "46") ~ "Vestland",
      is.na(Region_Name) & str_starts(Municipality_Code, "50") ~ "Trøndelag",
      is.na(Region_Name) & str_starts(Municipality_Code, "55") ~ "Troms",
      is.na(Region_Name) & str_starts(Municipality_Code, "56") ~ "Finnmark",
      TRUE ~ Region_Name  # Keep existing Region_Name if no conditions are met
    )
  ) %>%
  select(-Mun_name)

# Aggregating per municipality data
Vinmonopolet_market <- Vinmonopolet %>%
  group_by(Municipality_Code) %>%
  summarise(
    Mun_name = first(Municipality_Name),
    Region_Name = first(Region_Name),
    Population = first(Population),
    Area = first(Area),
    Number_of_stores = sum(`2024` > 0),  # Count non-zero sales
    Sales = sum(`2024`),
    Lat = first(Latitude),
    Lon = first(Longitude),
    Dist_nearest = first(dist_nearest_store),
  )


# Scaling the variables that have nt been scaled yet
```

```
Vinmonopolet_market <- Vinmonopolet_market %>%
  mutate(Population = Population / 1000,
         Sales = Sales / 1000)

# Now we have loaded and wrangled the main data set, but we can use some
# new variables for our analysis

## Merge 1: Grensehandel ########################################################

Grensehandel_weights <- here("Data", "Vinmonopolet", "Grensehandel_weights.xlsx")

# Load the weights datas
weights <- read_excel(Grensehandel_weights, skip = 3) %>%
  slice(1) %>%
  select(-'...1') %>%
  mutate(
    mean_weight = (as.numeric(`2024K1`) + as.numeric(`2024K2`) + as.numeric(`2024K3`) + as.
  )

weight_grensehandel <- weights$mean_weight / 100

# Load the regional data
Grensehandel_regions <- here("Data", "Vinmonopolet", "Grensehandel_regions.xlsx")

regional <- read_excel(Grensehandel_regions)

total_grensehandel <- sum(regional$"2024")

# Calculate grensehandel per region
regional <- regional %>%
  rename(
    Region = `Fylker`,
    Total_sale = `2024`
  ) %>%
  mutate(
    Grensehandel = Total_sale * weight_grensehandel
  )

# Split the "Vestlandet" region row into three new rows: "Rogaland", "Vestland" and "MC8re
regional <- regional %>%
  rbind(
    regional %>% filter(Region == "Vestlandet") %>% mutate(Region = "Rogaland"),
    regional %>% filter(Region == "Vestlandet") %>% mutate(Region = "Vestland"),
    regional %>% filter(Region == "Vestlandet") %>% mutate(Region = "Møre og Romsdal")
  ) %>%
  filter(Region != "Vestlandet")

# Divide the grensehandel value by three for "Rogaland", "Vestland" and "MC8re og Romsdal"
regional <- regional %>%
  mutate(
    Grensehandel = case_when(
      Region == "Rogaland" ~ Grensehandel * 0.35,
      Region == "Vestland" ~ Grensehandel * 0.46,
      Region == "Møre og Romsdal" ~ Grensehandel * 0.19,
      TRUE ~ Grensehandel  # Keep the original value for other regions
    )
  )

# Split the "Nord-Norge" region row into three new rows: "Nordland", "Troms" and "Finnmark"
# And divide the grensehandel value by three
regional <- regional %>%
  mutate(
    Grensehandel = ifelse(Region == "Nord-Norge", Grensehandel / 3, Grensehandel)
  ) %>%
  rbind(
    regional %>% filter(Region == "Nord-Norge") %>% mutate(Region = "Nordland"),
    regional %>% filter(Region == "Nord-Norge") %>% mutate(Region = "Troms"),
    regional %>% filter(Region == "Nord-Norge") %>% mutate(Region = "Finnmark")
  ) %>%
  filter(Region != "Nord-Norge")

# Divide the grensehandel value by three for "Nordland", "Troms" and "Finnmark"
regional <- regional %>%
  mutate(
    Grensehandel = case_when(
      Region == "Nordland" ~ Grensehandel * 0.5,
      Region == "Troms" ~ Grensehandel * 0.35,
      Region == "Finnmark" ~ Grensehandel * 0.15,
      TRUE ~ Grensehandel  # Keep the original value for other regions
    )
  )

# Split the "Agder, Telemark, Buskerud og Vestfold" column into four new columns: "Agder",
# And divide the grensehandel value by four
regional <- regional %>%
  mutate(
    Grensehandel = ifelse(Region == "Agder, Telemark, Buskerud og Vestfold", Grensehandel /
  ) %>%
  rbind(
    regional %>% filter(Region == "Agder, Telemark, Buskerud og Vestfold") %>% mutate(Regio
    regional %>% filter(Region == "Agder, Telemark, Buskerud og Vestfold") %>% mutate(Regio
    regional %>% filter(Region == "Agder, Telemark, Buskerud og Vestfold") %>% mutate(Regio
    regional %>% filter(Region == "Agder, Telemark, Buskerud og Vestfold") %>% mutate(Regio
  ) %>%
  filter(Region != "Agder, Telemark, Buskerud og Vestfold")

# Divide the grensehandel value by four for "Agder", "Telemark", "Buskerud" and "Vestfold"
regional <- regional %>%
  mutate(
    Grensehandel = case_when(
      Region == "Agder" ~ Grensehandel * 0.31,
      Region == "Telemark" ~ Grensehandel * 0.17,
      Region == "Buskerud" ~ Grensehandel * 0.26,
      Region == "Vestfold" ~ Grensehandel * 0.26,
      TRUE ~ Grensehandel  # Keep the original value for other regions
    )
  )

# Removing the "total_sale" column from the regional data set
regional <- regional %>% select(-Total_sale)

# Merge the regional data with the main data set on Region_Name in the Vinmonopolet_market
Vinmonopolet_market <- left_join(Vinmonopolet_market, regional, by = c("Region_Name" = "Reg

# Add a new column "Region_pop" where "Population" is summarized for each region
Vinmonopolet_market <- Vinmonopolet_market %>%
  group_by(Region_Name) %>%
  mutate(Region_pop = sum(Population)) %>%
  ungroup()
```

```
Vinmonopolet_market <- Vinmonopolet_market %>%
  mutate(Kommune_share = Population / Region_pop,
         Grensehandel_mun = Grensehandel * Kommune_share) %>%
  select(-c("Region_pop", "Kommune_share", "Grensehandel")) %>%
  rename(Grensehandel = Grensehandel_mun)




## Merge 2: Tourism ############################################################

Tourism_xlsx <- here("Data", "Vinmonopolet", "Tourism.xlsx")

# Reading tourism data
Tourism <- read_excel(Tourism_xlsx, skip = 4) %>%
  rename(
    Mun = '...1',
    H = 'Hotell og liknande overnattingsbedrifter',
    C = 'Campingplassar, hyttegrender og vandrarheim',
  ) %>%
  select(-'...2') %>%
  mutate_at(vars(H, C), ~as.numeric(str_replace_all(., ":", "0"))) %>%
  mutate(n_stays = H + C) %>%
  separate(Mun, into = c("Municipality_Code", "Municipality_Name"), sep = " ", remove = FAL
  select(-c("Mun", "H", "C", "Municipality_Name")) %>%
  filter(!is.na(Municipality_Code))

# Merging the data
Vinmonopolet_market <- left_join(Vinmonopolet_market, Tourism, by = "Municipality_Code") %>
  mutate(
    n_stays = ifelse(is.na(n_stays), 0, n_stays),
    n_stays = n_stays / 1000
  )

# There is a great deal of missing data, so we do not know the relevance of
# this data yet


## Merge 3: Income #############################################################
```

```
# Average monthly salary per inhabitant in the municipality

# Load data
Monthly_Salary <- here("Data", "Vinmonopolet", "Monthly_Salary.xlsx")

data <- read_excel(Monthly_Salary)

# Cleaning data by removing rows with missing values and rows with dots
clean_data <- data %>%
  filter(!apply(., 1, function(row) any(grepl("\\.", row)))) %>%
  na.omit()

# Remove the last two rows from the data, using tidyverse
clean_data <- clean_data %>%
  slice(1:(n() - 2)) %>%
  select(-'...2') %>%
  rename(
    Mun = `12852: Kommunefordelt månedslønn, etter region, statistikkmål, statistikkvariabe
    Monthly_salary = '...3'
  ) %>%
  separate(Mun, into = c("Municipality_Code", "Municipality_Name"), sep = " ", remove = FAL
  select(-c("Municipality_Name", "Mun")) %>%
  mutate(Monthly_salary = as.numeric(Monthly_salary),
         Monthly_salary = Monthly_salary / 1000)

# Merge with the main data set
Vinmonopolet_market <- left_join(Vinmonopolet_market, clean_data, by = "Municipality_Code")




## Merge 4: Concentration #######################################################

# Load data
Concentration_xlsx <- here("Data", "Vinmonopolet", "Concentration.xlsx")

concentration <- read_excel(Concentration_xlsx, skip = 5) %>%
  slice(1:357) %>%
  select('...1',
         'Spredtbygd strøk...3') %>%
  rename(Mun = '...1',
         Spread = 'Spredtbygd strøk...3') %>%
  separate(Mun, into = c("Municipality_Code", "Municipality_Name"), sep = " ", remove = FAL
  select(-c("Municipality_Name", "Mun")) %>%
  mutate(Spread = as.numeric(Spread),
         Spread = Spread / 1000)

# Remove the first two characters of each cell in the "Municpality_Code" column
concentration$Municipality_Code <- substr(concentration$Municipality_Code, 3, nchar(concent

# Merge with the main data set
Vinmonopolet_market <- left_join(Vinmonopolet_market, concentration, by = "Municipality_Cod


## Merge 5: "Active" stores #####################################################

# Load data
Active_xlsx <- here("Data", "Vinmonopolet", "Active.xlsx")

A1 <- read_excel(Active_xlsx, sheet = 1, skip = 2)

A2 <- read_excel(Active_xlsx, sheet = 2, skip = 2)

# Merge the two data sets
Active <- A1 %>%
  bind_rows(A2) %>%
  select(-c('1', '...3', Fylke))

# Rename columns
names(Active)[1] <- "Mun_name"

# Remove unncessary spaces and numbers from the "Mun_name" column
Active$Mun_name <- substr(Active$Mun_name, 4, nchar(Active$Mun_name))

Active$Mun_name <- trimws(Active$Mun_name, which = "left")

# Replace norwegian special letters with english ones and make all letters lowercase
Active$Mun_name <- tolower(iconv(Active$Mun_name, from = "UTF-8", to = "ASCII//TRANSLIT"))
```

```
# Recode the "Mun_name" column
Active$Mun_name <- case_when(
  Active$Mun_name == "hamaroy" ~ "habmer - hamaroy",
  Active$Mun_name == "hattfjelldal" ~ "aarborte - hattfjelldal",
  Active$Mun_name == "valer (viken)" ~ "valer (ostfold)",
  TRUE ~ Active$Mun_name)

# Merge with the main data set

# Make a dummy variable for active stores
Vinmonopolet_market$Active <- ifelse(Vinmonopolet_market$Mun_name %in% Active$Mun_name, 1,

## Write to Excel ###############################################################

# Write to Excel
# write_xlsx(Vinmonopolet_market, "demand_data.xlsx")
```

Model applications

Data preparation

```
### Data preparation ############################################################

# Rename relevant columns in accordance with tidyverse standards
Vinmonopolet_market <- Vinmonopolet_market %>%
  rename(
    mun_code = Municipality_Code,
    mun_name = Mun_name,
    region_name = Region_Name,
    population = Population,
    area = Area,
    number_of_stores = Number_of_stores,
    sales = Sales,
    lat = Lat,
    lon = Lon,
    dist_nearest = Dist_nearest,
    grensehandel = Grensehandel,
    n_stays = n_stays,
    monthly_salary = Monthly_salary,
    spread = Spread,
    active = Active
  )


# Narrowing down the data to only contain relevant markets
# Excluding the largest cities because they are not representative

# Train and test split, training data all observations with a store
train_data <- Vinmonopolet_market %>%
  filter(number_of_stores > 0)

# Test data all observations without a store
test_data <- Vinmonopolet_market %>%
  filter(number_of_stores == 0)
```

Model selection and basic regressions

```
### Model selection ############################################################

# Forward selection
forward_model <- step(lm(sales ~ 1, data = train_data),
                      scope = ~ population + grensehandel + n_stays + monthly_salary + area
                      direction = "forward")

#summary(forward_model)

# Backward selection
backward_model <- step(lm(sales ~ population + grensehandel + n_stays + monthly_salary + ar
                          data = train_data),
                       direction = "backward")

#summary(backward_model)


lm_Area <- lm(sales ~ area, data = train_data)
```

```
#summary(lm_Area)

lm_pop <- lm(sales ~ population, data = Vinmonopolet_market)

#summary(lm_pop)

small_data <- Vinmonopolet_market %>%
  filter(number_of_stores == 1 | 0)

lm_pop_test <- lm(sales ~ population, data = small_data)

#summary(lm_pop_test)

# Linear regression model for predicting sales with all the variables
var_test <- lm(sales ~ population + grensehandel + n_stays + monthly_salary + area +
                 number_of_stores + spread,
               data = Vinmonopolet_market)

#stargazer(var_test, type = "text")

# From these regressions we see that we want to remove the "Area" and "prop_spread" variabl
# from the regressions as they are not significant.
```

Demand estimation

```r
### Demand estimation ##########################################################

## Linear regression

# Predicting sales using the training data
reg1 <- lm(sales ~ population + grensehandel + n_stays + monthly_salary,
           data = train_data)

stargazer(reg1, type = "text")
```

```
===============================================
                        Dependent variable:    
                    ---------------------------
                               sales           
-----------------------------------------------
population                   16.449***         
                              (0.490)          
                                               
grensehandel                 -5.264***         
                              (1.456)          
                                               
n_stays                      0.246***          
                              (0.043)          
                                               
monthly_salary                4.885**          
                              (2.290)          
                                               
Constant                    -270.085**         
                             (125.336)         
                                               
-----------------------------------------------
Observations                    237            
R2                             0.990           
Adjusted R2                    0.990           
Residual Std. Error      102.638 (df = 232)    
F Statistic         5,683.347*** (df = 4; 232)
```

```
===============================================
Note:               *p<0.1; **p<0.05; ***p<0.01
```

```r
# Applying the model on the test data
test_data$sales_pred <- predict(reg1, newdata = test_data)

## Merge predicted data into the original data

# Deselect unnecessary columns to merge the data easier
test_data <- test_data %>%
  select(mun_code, sales_pred)

# Merge predicted demand (sales) back into the original data
Vinmonopolet_market <- Vinmonopolet_market %>%
  left_join(test_data, by = "mun_code") %>%
  mutate(sales = ifelse(sales == 0, sales_pred, sales)) %>%
  select(-sales_pred) %>%
  mutate(sales = ifelse(sales < 0, 0, sales),
         number_of_stores = as.integer(number_of_stores)) %>%
  filter(number_of_stores < 2)
```

Logit model

```r
## Logit regression ############################################################

# Make sure the factor for Number_of_stores has valid R variable names
# that won't cause errors in caret. For instance, rename "0" -> "NoStore"
# and "1" -> "OneStore".
data_for_logit <- Vinmonopolet_market %>%
  mutate(number_of_stores = as.factor(number_of_stores))

# Rename factor levels (originally "0" and "1") to "NoStore" and "OneStore"
data_for_logit$number_of_stores <- factor(
  data_for_logit$number_of_stores,
  levels = c("0", "1"),
  labels = c("NoStore", "OneStore")
)

# Set up k-fold cross-validation parameters
set.seed(123)  # for reproducibility

my_control <- trainControl(
  method = "cv",            # k-fold CV
  number = 5,               # 5 folds
  classProbs = TRUE,        # needed for probability output
  summaryFunction = twoClassSummary
)

# Train the logistic model with cross-validation
cv_model <- train(
  number_of_stores ~ sales,
  data = data_for_logit,
  method = "glm",
  family = binomial,
  trControl = my_control,
  metric = "ROC"            # use AUC (Area Under the Curve) as our metric
)

# Review cross-validation results
print(cv_model)
```

```
Generalized Linear Model

316 samples
  1 predictor
  2 classes: 'NoStore', 'OneStore'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 252, 253, 253, 253, 253
Resampling results:

  ROC        Sens       Spec
  0.9397703  0.8666667  0.8670513
```

```
print(cv_model$results)
```

```
  parameter       ROC      Sens      Spec      ROCSD    SensSD     SpecSD
1      none 0.9397703 0.8666667 0.8670513 0.02518379 0.1078515 0.04987575
```

```
# Get predicted probabilities from the final trained model
# caret retrains on the entire dataset after CV by default
Vinmonopolet_market$prob <- predict(cv_model, newdata = data_for_logit, type = "prob")[, "O

# For recommended stores
recommended_stores <- Vinmonopolet_market %>%
  mutate(number_of_stores = as.integer(as.character(number_of_stores))) %>%
  filter(number_of_stores == 0, dist_nearest > 0) %>%
  arrange(desc(prob)) %>%
  select(mun_name, prob, dist_nearest, sales, population, region_name, active) %>%
  mutate(
    dist_nearest = round(dist_nearest, 0),  # Round dist_nearest to 0 decimals
    sales = round(sales, 0),                 # Round sales to 0 decimals
    across(where(is.numeric) & !c(dist_nearest, sales), ~round(., 3))  # Round other numeri
  )

# Create table
kable(head(recommended_stores, 10),
      format = "markdown",
      digits = 3,
      longtable = TRUE) %>%
  kable_styling(latex_options = "scale_down")
```

mun_name	prob	dist_nearest	sales	population	region_name	active
giske	0.995	3	156	8.773	Møre og Romsdal	0
lunner	0.993	7	150	9.420	Akershus	1
rade	0.967	13	118	7.850	Østfold	0
hareid	0.857	10	87	5.320	Møre og Romsdal	1
valer (ostfold)	0.823	11	82	6.162	Østfold	1
aurland	0.817	37	81	1.836	Vestland	1
birkenes	0.752	13	74	5.413	Agder	0
eidskog	0.729	23	71	6.059	Innlandet	1
aure	0.683	28	67	3.394	Møre og Romsdal	1
austrheim	0.658	16	65	2.915	Vestland	1

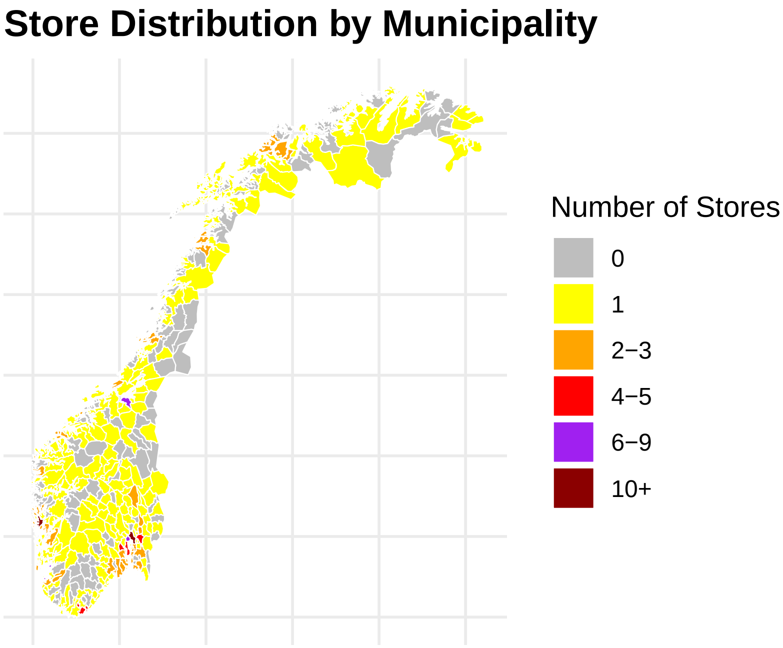
```
# For active stores
Active_stores <- Vinmonopolet_market %>%
  mutate(number_of_stores = as.integer(as.character(number_of_stores))) %>%
  filter(active == 1, dist_nearest > 0) %>%
  arrange(desc(prob)) %>%
  select(mun_name, prob, dist_nearest, sales, population, region_name, active) %>%
  mutate(
    dist_nearest = round(dist_nearest, 0),  # Round dist_nearest to 0 decimals
    sales = round(sales, 0),                # Round sales to 0 decimals
    across(where(is.numeric) & !c(dist_nearest, sales), ~round(., 3))  # Round other numeri
  )

# Create table
kable(head(Active_stores, 10),
      format = "markdown",
      digits = 3,
      longtable = TRUE) %>%
  kable_styling(latex_options = "scale_down")
```

mun_name	prob	dist_nearest	sales	population	region_name	active
lunner	0.993	7	150	9.420	Akershus	1
hareid	0.857	10	87	5.320	Møre og Romsdal	1
valer (ostfold)	0.823	11	82	6.162	Østfold	1
aurland	0.817	37	81	1.836	Vestland	1
eidskog	0.729	23	71	6.059	Innlandet	1
aure	0.683	28	67	3.394	Møre og Romsdal	1
austrheim	0.658	16	65	2.915	Vestland	1
aukra	0.644	15	64	3.759	Møre og Romsdal	1
vaksdal	0.617	19	61	3.875	Vestland	1
sokndal	0.544	21	55	3.371	Rogaland	1

Visualisations

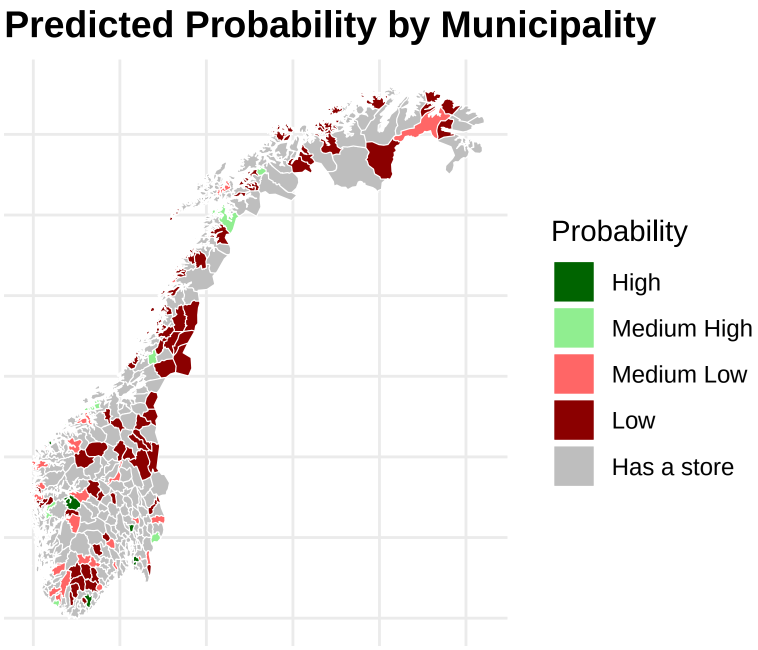
```
# Use the existing municipality map data
municipalities <- nor_municip_map_b2024_default_sf
```

```r
# Read your store dataset
store_data_path <- here("Data", "Vinmonopolet", "final_data_24.xlsx")
store_data <- readxl::read_excel(store_data_path)

# Extract the last 4 digits from "location_code" to get the municipality code
municipalities <- municipalities %>%
  mutate(municip_code = as.integer(str_sub(location_code, -4)))  # Extract last 4 digits

# Count the number of stores per municipality
store_counts <- store_data %>%
  group_by(Municipality_Code) %>%
  summarise(num_stores = n())

# Changing to matching format
store_counts$Municipality_Code <- as.numeric(store_counts$Municipality_Code)

# Merge store counts with the geographical dataset
merged_data <- municipalities %>%
  left_join(store_counts, by = c("municip_code" = "Municipality_Code"))

# Assign store categories, treating NA as 0
merged_data <- merged_data %>%
  mutate(store_category = case_when(
    is.na(num_stores) ~ "0",
    num_stores == 1 ~ "1",
    num_stores %in% 2:3 ~ "2-3",
    num_stores %in% 4:5 ~ "4-5",
    num_stores %in% 6:9 ~ "6-9",
    num_stores >= 10 ~ "10+",
    TRUE ~ NA_character_
  ))

# Make sure factor levels are ordered correctly
merged_data$store_category <- factor(
  merged_data$store_category,
  levels = c("0", "1", "2-3", "4-5", "6-9", "10+")
)

# Plot the heatmap with discrete color bins
ggplot(merged_data) +
  geom_sf(aes(fill = store_category), color = "white", size = 0.1) +
  scale_fill_manual(
    values = c("0" = "grey", "1" = "yellow", "2-3" = "orange", "4-5" = "red", "6-9" = "purp
    name = "Number of Stores"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank()) +
  labs(title = "Store Distribution by Municipality",
       x = NULL, y = NULL)
```



```
# Extract the last 4 digits from "location_code" to get the municipality code
municipalities <- municipalities %>%
  mutate(municip_code = as.integer(str_sub(location_code, -4)))  # Extract last 4 digits

# Merge population data with the existing geographical dataset
Vinmonopolet_market$mun_code <- as.numeric(Vinmonopolet_market$mun_code)
```

```
# Add a "prob_category" for municipalities that have at least one store
Vinmonopolet_market <- Vinmonopolet_market %>%
  mutate(prob_category = case_when(
    number_of_stores > 0 ~ "Has a store",
    prob >= 0 & prob < 0.25 ~ "Low",
    prob >= 0.25 & prob < 0.5 ~ "Medium Low",
    prob >= 0.5 & prob < 0.75 ~ "Medium High",
    prob >= 0.75 & prob <= 1 ~ "High",
    TRUE ~ NA_character_
  ))

# Set factor levels to control legend order
Vinmonopolet_market$prob_category <- factor(Vinmonopolet_market$prob_category,
                                       levels = c("High", "Medium High", "Medium Low", "Low

merged_prob_data <- municipalities %>%
  left_join(Vinmonopolet_market, by = c("municip_code" = "mun_code")) %>%
  mutate(prob_category = replace_na(prob_category, "Has a store"))  # Fill NAs with "Has a

# Plot
ggplot(merged_prob_data) +
  geom_sf(aes(fill = prob_category), color = "white", size = 0.1) +
  scale_fill_manual(
    values = c(
      "High" = "darkgreen",
      "Medium High" = "#90EE90",
      "Medium Low" = "#FF6666",
      "Low" = "darkred",
      "Has a store" = "grey"
    ),
    name = "Probability"
  ) +
  coord_sf() +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank()
  ) +
  labs(title = "Predicted Probability by Municipality",
       x = NULL, y = NULL)
```


Predicted Probability by Municipality
Probability
🟩 High
🟩 Medium High
🟥 Medium Low
🟥 Low
⬜ Has a store