Capstone Report - Climate Change Opinion at the County Level

Michael Beller

01/04/2021

1. Executive Summary

This project involved building a machine learning model to predict the percentage of support for the statement "Global warming is caused mostly by human activities" at the county level in the Unites States. Climate change opinion data at the county level was obtained from the Yale Program on Climate Change Communication. The model used predictors selected from population, economic, and climate-related attributes, with the data for these predictors obtained from various U.S. government agencies. The resulting county-level data set was split into subsets for training, testing, and validation. The final model was run against the validation set. Results for the validation can be found in ***Section 4***.

2. Introduction

While scientific awareness of and concern regarding climate change dates back to at least the 1950s, the concept first started to reach critical mass in the public consciousness in the 1980s, often with the "global warming" moniker. This emergence as a potential threat led to significant research and modeling, while the general public's attitudes ranged from dismissal to grave concern.[1]

Today, there is a near universal consensus in the scientific community that climate change is a real phenomenon, cause mostly by human activity, and a serious threat to humanity, due to the scale of its potential destructive impact and to the possibility of acting quickly to ameliorate some of its worst effects. However, among the general public, the situation is quite different. Some people are even more alarmed than the scientific community, while others believe climate change is overblown at best or an elaborate conspiracy at worst. This lack of strong public support hinders attempts to take action to combat it.

In the United States, climate change has been a topic in numerous national and state-level polls. While such polls provide a measure insight into public opinion, they only provide a high-level, broad-brush view of it. To influence opinions or to build support for action, it is often necessary to understand public opinions at a more local level.

The Yale Program on Climate Change Communication has produced a data set that takes climate change survey data from a variety of national, state, and local levels and uses it to estimate percentages of support for climate change statements at the U.S. county level, among other localized levels. More detail about this data set can be found on their website: https://climatecommunication.yale.edu/visualizations-data/ycom-us/. In addition, the details of the algorithm used to produce these county-level estimates can be found in the following article:

Howe, P., Mildenberger, M., Marlon, J., & Leiserowitz, A. (2015) "Geographic variation in opinions on climate change at state and local scales in the USA," Nature Climate Change. DOI: 10.1038/nclimate2583.

As this source describes, the estimation of climate change opinion at the county level was based on personal characteristics of survey respondents: "…race, education, gender, mode [of survey], time, and geographical variables" (Mildenberger & Leiserowitz, p. 7.)

[1] https://history.aip.org/climate/public.htm#L000

2.1 Project Goal

After viewing the Yale Program on Climate Change Communication's website, two questions arose that became the genesis of this project:

1. Do actual changes in climate cause changes in opinions about climate change?
2. If so, can county-level opinions on climate change be predicted by looking at climate factors?

As the project developed, the scope widened include two other classes of predictors. Thus, the project goal became to predict a county's percentage of support for a climate change statement based on population, economic, and climate-related factors.

Since estimated percentage of support is a continuous outcome, Root Mean Square Error (RMSE) was selected as the loss function for evaluating model effectiveness.

2.2 Background on U.S Counties[2]

In 48 of the 50 U.S. states, the next division of government below the state level is the county. The powers and roles of county government vary widely from state to state, but they exist in some recognizable form. One of the two exceptions is Louisiana, which uses the term parish for its next level subdivisions, all of which effectively function as counties. The other is Alaska, which uses boroughs for a little under half of its land area, with the remainder a large area (the Unorganized Borough) with only town-level governments. The US Census Bureau divides the Unorganized Borough into several county-equivalent census districts for the purpose of population analysis.

The state of Virginia has 38 additional entities called independent cities, which do not fall under the jurisdiction of a county and are treated as county-equivalents. Maryland, Missouri, and Nevada each have one independent city as well, also treated as county-equivalents.

In this report, the term "county" should be interpreted as county-equivalent, referring equally to an actual county, a parish, a borough, an independent city, or an Alaskan census district. This is consistent with U.S. government reporting practices.

Counties in the US vary considerably by area and population. San Bernardino County, California is over 20,000 square miles, around the size of West Virginia. Bristol County, Rhode Island is about 25 square miles. Los Angeles County, California has a population of over 10 million people, bigger than 41 individual states. Loving County, TX's estimated 2019 population is 169.

As of 2019, there are a total of 3,141 counties in the 50 U.S. states (excluding the District of Columbia and U.S. territories). These 3,141 counties would be the level of prediction for the project.

County names are frequently duplicated across states (30 states have a Washington County, for example). In some cases, county names may be the same as state names (such as Iowa County in Iowa), even when they are in different states (such as Iowa County, Wisconsin).

2.3 Range of Predictors Considered

This project sprung from consideration of the relationship between climate changes and changes in climate opinions, so climate-based predictors were a natural inclusion. But the goal was to include other predictors as well, both to improve the performance of the model and to compare the relative strength of climate predictors against other types.

Because the county-level opinion estimates from the Yale Program on Climate Change Communication were projected based on personal characteristics of the survey respondents, any such personal characteristics were

[2]https://en.wikipedia.org/wiki/County_statistics_of_the_United_States

not considered as predictors for this project. With those excluded, brainstorming resulted in three additional classes of predictors that might be considered:

1. Population-related predictors: information relating to county populations, growth or decline in county populations, and proximity to the coast, where rising sea levels might influence opinions.
2. Economic-related predictors: information relating to county economic health from a business and personal perspective, as well as the presence of fossil fuel extraction in a county, which might influence opinions.
3. Political predictors: information related to electoral preferences/results for the two major political parties at the county level.

Population and economic classes were selected to include in the predictor selection process along with climate-related factors. Political factors were not included, both for time constraints and for their potential to overshadow other predictors.

2.4 Data Sources

With population, economic, and climate predictors in scope, the next step was locating data sources with elements that could be used as predictors. Listed here are brief descriptions of the sources used. Specific web sites and file names for these sources can be found in ***Appendix 1***. In addition, all of these files are available on the project github site: **https://github.com/mbeller2016/Climate-Change-Opinion-Prediction/raw/main/Data/**.

Data on climate change opinions was obtained from the Yale Program on Climate Change Communication. This file included estimated support and oppose percentages at the county level for 31 different statements about climate change. The time constraints of this project necessitated focusing on only one statement. The statement selected was "Global warming is caused mostly by human activities", and the data element used was the estimated percentage of support for that statement in each county.

For climate data, the U.S. National Oceanic & Atmospheric Administration (NOAA) has historical data by county going back into the late 1800s. These are stored in a one record per year format, showing means for each month for a given county. There were four data sets used by this project, one for each of the following climate statistics:

- Mean temperature (the mean temperature recorded for the county for that time period)
- Mean maximum temperature (the average maximum daily temperature recorded for the county for that time period)
- Mean minimum temperature (the average minimum daily temperature recorded for the county for that time period)
- Precipitation (the total precipitation recorded for the county for that time period)

For population data, this project used a data set from the U.S. Census Bureau containing the 2010 official census populations, the 2019 population estimates, and the estimated components of the 2010-2019 change in population. Another Census Bureau file provided a list of coastal counties.

For economic data, datasets from the Bureau of Economic Analysis (BEA), a part of the U.S. Department of Commerce, provided information on GDP and personal income in the 2001-2019 timeframe. Two data files from the Energy Information Administration, a part of the U.S. Department of Energy, contained information on counties with fossil fuel extraction/industry.

All of this data would ultimately be linked together in a **county_master** tibble for subsequent analysis. Since some sources used different county identifiers than others, it was necessary to build a cross-reference table. Census Bureau CFIPS codes were used to uniquely identify each county. A list of these codes was downloaded from the Census Bureau, and a list of state names/abbreviations was downloaded to populate this cross-reference table.

2.4 Key Steps

The execution of this project involved the following procedure:

1. Load external data sources into an R-readable format.
2. Wrangle external data to extract potential predictors and store them in the **county_master** tibble.
3. Split the county data into a training set for model building, a test set for model evaluation, and a validation set for determining the accuracy of the final model.
4. Analyze potential predictors, and filter them to focus on the ones most likely to have significant predictive value.
5. Build a series of models using subsets of the selected predictors and three different modeling algorithms, and analyze the results using Root Mean Square Error.
6. Based on the results from step 5, build a final model and run it against the validation set.

As implemented, steps 3 through 5 were first performed individually for the three classes of predictors – population, economic, and climate. The final model was built by combining the results of these three classes.

3. Methods & Analysis

3.1 Data Cleaning & Formatting

Before analysis of potential predictors could begin, there were a significant number of wrangling challenges to overcome. These challenges can be grouped into three categories:

1. **Identification challenges**: Most data sources used the Census Bureau CFIPS code to uniquely identify a county, and this project followed suit. The CFIPS is a 5-digit code, with the first two digits representing the state and the last three the county. However, other data sources used a similar identifier with different state codes, or their own code, or no code at all, just a state and county name.
2. **Geographic scoping challenges**: Some data sources included values for other levels mixed in with county data, such as state, congressional district, or metropolitan area. These had to be filtered out to focus on the county data.
3. **Predictor scoping challenges**: Many files contained a large variety of data elements, or covered large periods of time. The number of predictors selected had to be kept to a manageable number.

The mechanics of loading and wrangling the data, including how these three types of challenges were addressed, are discussed in the subsections below.

3.1.1 Environment Setup

This code block loads packages and libraries required for the project.

```r
############################################
# Load required packages and libraries
############################################
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")
if(!require(httr)) install.packages("httr", repos = "http://cran.us.r-project.org")
if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages("Rborist", repos = "http://cran.us.r-project.org")
```

```
library(tidyverse)
library(readr)
library(httr)  #Because readxl functions don't accept URLs
library(readxl)
library(caret)
```

3.1.2 Data Loading

All external data sources had been downloaded from their respective websites to this project's data subdirectory on github: https://github.com/mbeller2016/Climate-Change-Opinion-Prediction/raw/main/Data/.

The next step was to read these files into the R environment, using different methods based on the type of external data file.

```
###########################################
# Load raw external data files
###########################################

options(digits=3)

#All data sources can be found in this github repository
mb_gitpath <- "https://github.com/mbeller2016/Climate-Change-Opinion-Prediction/raw/main/Data/"

#Get the geocode file (cross-reference county/state and CFIPS code) from Census Bureau
#Note: apparently can't use URLs with the readxl functions, so had to use the
# httr library and write_disk to get these transferred over, per stackoverflow
GET(paste(mb_gitpath,"all-geocodes-v2019.xlsx",sep=""),
            write_disk(gfile <- tempfile(fileext = ".xlsx")))
```

```
#First 4 lines are title information, so skip them
GEOCODE_file <- read_xlsx(path=gfile, skip=4)

#Load the state name/abbreviation file
STABBR_file <- read_csv(paste(mb_gitpath,"csvData.csv",sep=""))

#Load climate change opinion data from Yale Climate Change Communication Project
YCOM_file <- read_csv(paste(mb_gitpath,"YCOM_2020_Data.csv",sep=""))

#Load population-related data from US Census Bureau, at
#https://www2.census.gov/programs-surveys/popest/datasets/2010-2019/counties/totals/co-est2019-alldata.
CEN_file <- read_csv(paste(mb_gitpath,"co-est2019-alldata.csv",sep=""))

#Load economic and income data from BEA at
#https://apps.bea.gov/regional/downloadzip.cfm
GDP_file <- read_csv(paste(mb_gitpath,"CAGDP1__ALL_AREAS_2001_2019.csv",sep=""))
INC_file <- read_csv(paste(mb_gitpath,"CAINC1__ALL_AREAS_1969_2019.csv",sep=""))

#Load oil and gas info from EIA (Energy Information Administration) at
#Coal: https://www.eia.gov/coal/annual/xls/table2.xls
#Oil/NG by region: https://www.eia.gov/petroleum/drilling/xls/dpr-data.xlsx
#Oil/NH region definitions: in RegionCounties tab of the above
```

```r
#Note: apparently can't use URLs with the readxl functions, so had to use the
# httr library and write_disk to get these transferred over, per stackoverflow
GET(paste(mb_gitpath,"table2.xls",sep=""),write_disk(cfile <- tempfile(fileext = ".xls")))
```

```r
#First three lines are title data, so skip them
COAL_file <- read_xls(path=cfile, skip=3)

#Oil&gas data has one sheet per regions, requires multiple reads
#Also has a sheet for region/county mapping
GET(paste(mb_gitpath,"dpr-data.xlsx",sep=""),write_disk(ofile <- tempfile(fileext = ".xlsx")))
```

```r
OIL_file <- read_xlsx(path=ofile,sheet="RegionCounties")

#Load coastline counties list, using the same httr workaround
GET(paste(mb_gitpath,"coastline-counties-list.xlsx",sep=""),
            write_disk(ccfile <- tempfile(fileext = ".xlsx")))
```

```r
#First 3 lines are title information, so skip them
COAST_file <- read_xlsx(path=ccfile, skip=3)

#Load climate data from ftp://ftp.ncdc.noaa.gov/pub/data/cirs/climdiv/
#State codes in county-readme.txt
#Precip: climdiv-pcpncy-v1.0.0-20201205
#Max temp: climdiv-tmaxcy-v1.0.0-20201205
#Min temp: climdiv-tmincy-v1.0.0-20201205
#Temperature: climdiv-tmpccy-v1.0.0-20201205
#These files have no delimiters, but instead come with documentation outlining how each
#column should be interpreted.  Thus, they are read in with the read_lines function, then
#parsed.
CLIMREADME_file <- read_lines(paste(mb_gitpath,"county-readme.txt",sep=""))
TEMP_file <- read_lines(paste(mb_gitpath,"climdiv-tmpccy-v1.0.0-20201205",sep=""))
TMAX_file <- read_lines(paste(mb_gitpath,"climdiv-tmaxcy-v1.0.0-20201205",sep=""))
TMIN_file <- read_lines(paste(mb_gitpath,"climdiv-tmincy-v1.0.0-20201205",sep=""))
PRCP_file <- read_lines(paste(mb_gitpath,"climdiv-pcpncy-v1.0.0-20201205",sep=""))
```

3.1.3 County Identifiers

To address identification challenges in several files, this step built a cross-reference table containing the Census Bureau CFIPS code, state and county names, and state abbreviations. This table also served as the start of the **county__master** tibble.

```r
##########################################
# Preprocess geocode file
##########################################

#This file contains many classifications that go smaller than county.  We can eliminate
#these if we understand what they are.
str(GEOCODE_file)

gtest <- GEOCODE_file %>% group_by(`Summary Level`) %>% summarize(count=n())
gtest
```

```
gtest <- GEOCODE_file %>% filter(`Summary Level`=="040")
print(gtest[,7],n=100)

gtest <- GEOCODE_file %>% filter(`Summary Level`=="170")
print(gtest[,7],n=100)

#Remove anything that isn't a state (Summary Level 040) or a county (Summary Level 050)
#Also remove District of Columbia (State FIPS 11) and Puerto Rico (State FIPS 72)
geocodes <- GEOCODE_file %>%
              filter(`Summary Level`=="040" | `Summary Level`=="050") %>%
              filter(`State Code (FIPS)` != "72" &
                     `State Code (FIPS)` != "11") %>%
              rename(level = `Summary Level`,
                     state = `State Code (FIPS)`,
                     county = `County Code (FIPS)`,
                     name = `Area Name (including legal/statistical area description)`) %>%
              select(level, state, county, name) %>%
              arrange(state, level, county)

#Extract state codes and names, convert to all upper case
geostate <- geocodes %>%
              filter(level=="040") %>%
              select(state, name) %>%
              rename(sname = name) %>%
              mutate(sname = toupper(sname))

#Some data files only have state abbreviations -- add them to geostate
STABBR_file <- STABBR_file %>%
                 mutate(sname = toupper(State))
geostate <- inner_join(geostate,STABBR_file,by="sname")

#Add state name to county record, convert
geocodes <- geocodes %>%
              filter(level=="050") %>%
              rename(cname = name) %>%
              mutate(cname = toupper(cname),
                     cfips = paste(state,county,sep="")) %>%
              select(cfips, state, county, cname)
geocodes <- inner_join(geocodes,geostate,by="state") %>%
              select(-State,-Abbrev)

#Begin master county file
county_master <- geocodes
```

3.1.4 Preprocess coastal county data

The coastal county data was already identified by CFIPS, making it easy to process. It was added to the **county_master** tibble as a binary flag (1 for coastal counties, 0 for non-coastal counties).

```
##########################################
# Preprocess coastal counties file
##########################################
str(COAST_file)
```

```
#There are weird extra spaces and tabs in the column names, so rename them
colnames(COAST_file) <- c("cfips","state","county","county_name","state_name",
                          "coast_region","v2016")
county_coastal <- COAST_file %>%
                    select(cfips,coast_region)

#Add a coastal yes=1,no=0 column to the county master file
county_master <- left_join(county_master,county_coastal,by="cfips")
county_master <- county_master %>%
                   mutate(coast = ifelse(is.na(coast_region),0,1)) %>%
                   select(-coast_region)

rm(county_coastal)
```

3.1.5 Preprocess Coal Production File

This file had a number of identification challenges, since it was an Excel spreadsheet with no county identifier, just a series of row heading with state names, each one followed followed by other rows with county names. County names were duplicated across states, and in some cases were the same as state names (e.g. Ohio). The correct determination of state vs county was calculated by tracking the coal production statistics to tell whether a given line was a state or county.

The coal production file contained data on number of mines and production volumes. This was simplified to a binary flag (1 if the county produced coal, and 0 if it did not) and added to the **county_master** tibble.

```
###########################################
# Preprocess coal file
###########################################
str(COAL_file)

#Lots of issues with states and counties intermingled, blank lines, etc.

#We only want the location and the total number of mines, so remove other fields
county_coal <- COAL_file %>%
                 rename(location = ‘Coal-Producing‘,
                 mines = ‘Number of Mines...6‘) %>%
                 select(location, mines)

#Eliminate any NA state and counties
county_coal <- county_coal %>%
                 filter(is.na(location)==FALSE) %>%
                 filter(is.na(mines)==FALSE) %>%
                 mutate(sname = toupper(location),
                        is_state = -1)

#The last three lines are subtotals, remove them
county_coal <- county_coal[1:(nrow(county_coal)-3),]

#Check for county names = state names
county_coal <- left_join(county_coal,geostate,by="sname")
print(county_coal,n=200)

#Need to assign state codes to each county row, accounting for state/county name
```

```r
# confusion.
statecode = "00"
minesum <- -1
minetot <- -1
for (x in 1:nrow(county_coal)) {
  #If the state code is NA, it is a county
  if (is.na(county_coal$state[x])) {
    county_coal$is_state[x] <- 0
    county_coal$state[x] <- statecode
    minesum <- minesum + county_coal$mines[x]
  } else if (minesum==minetot) {
      #If the mine totals match, it is a state
      county_coal$is_state[x] <- 1
      statecode <- county_coal$state[x]
      minetot <- county_coal$mines[x]
      minesum <- 0
  } else {
      #If the mine totals didn't match, it is a county
      county_coal$is_state[x] <- 0
      county_coal$state[x] <- statecode
      minesum <- minesum + county_coal$mines[x]
  }
}

#Remove the state rows from the table
county_coal <- county_coal %>% filter(is_state==0)

#Now we need to get the county code and the full FIPS.  The names in county_coal do not
#include the word "County" at the end.  Also, in Alaska (state FIPS 02) they use "Borough",
#and in Louisiana (state FIPD 22) they use Parish.  Convert all to upper case.
county_coal <- county_coal %>%
                 mutate(cname = case_when(
                   state=="02" ~ toupper(paste(sname,"Borough")),
                   state=="22" ~ toupper(paste(sname,"Parish")),
                   TRUE ~ toupper(paste(sname,"County")))) %>%
                 select(state,cname,mines)

#Fix one problematic county name
fixcounty <- which(county_coal$cname=="DE KALB COUNTY")
county_coal$cname[fixcounty] <- "DEKALB COUNTY"

#Get the state and county FIPS codes
county_coal <- left_join(county_coal,geocodes,by=c("state","cname")) %>%
               select(cfips,mines)

#Add the coal yes=1,no=0 column to the county master file
county_master <- left_join(county_master,county_coal,by="cfips")
county_master <- county_master %>%
  mutate(coal = ifelse(is.na(mines),0,1)) %>%
  select(-mines)

rm(county_coal)
```

3.1.6 Preprocess Oil/Natural Gas Production File

The county codes used in this file matched the county portion of the CFIPS, but the state codes were different. This was addressed by linking to the cross-reference table to get the correct CFIPS code.

This Excel file contained data on volumes and types of oil & natural gas production. As with the coal file, this was simplified to a binary flag and added to the **county_master** tibble.

```r
###########################################
# Preprocess oil&gas file
###########################################
str(OIL_file)

#A nice easy file, with the fips code components already present.  Or so it seemed.
#The county codes line up, but the state codes don't.

OIL_file <- OIL_file %>%
              mutate(Code = State) %>%
              select(-State)
county_oil <- left_join(OIL_file, geostate, by = "Code")
county_oil <- county_oil %>%
                mutate(cfips = paste(state,CountyID,sep="")) %>%
                select(cfips,Region)

#Add the oil&gas yes=1,no=0 column to the county master file
county_master <- left_join(county_master,county_oil,by="cfips")
county_master <- county_master %>%
  mutate(oilgas = ifelse(is.na(Region),0,1)) %>%
  select(-Region)

rm(county_oil)
```

3.1.7 Preprocess Census Population File

This file used the correct CFIPS codes, so the main identification task was filtering out other geographical slices of the data (state, metro area).

As the 2019 population estimate, this file contained the 2010 official population numbers for each county, as well as a set of columns for each year between 2010 and 2019 showing all the estimated components of population since then – births, deaths, immigration, emigration, etc. The following were selected as potential predictors and added to the **county_master** tibble:

- 2019 population (this gives a sense of urban vs. suburban vs. rural)
- Natural population change, 2010-2019 (births - deaths, or is the county growing or declining from an organic perspective)
- Population change due to domestic immigration, 2010-2019 (people moving into/out of the county from/to other counties in the U.S.)
- Population change due to international immigration, 2010-2019 (people moving into/out of the county from/to other nations)

```r
###########################################
# Preprocess census file
###########################################
str(CEN_file)
```

```
#We only want summary level 050 (county)
#Get current population, natural changes since 2010, immigration changes since 2010
county_census <- CEN_file %>%
                   filter(SUMLEV == "050") %>%
                   mutate(cfips = paste(STATE,COUNTY,sep=""),
                          pop2019 = POPESTIMATE2019,
                          popchgnat = NATURALINC2010 + NATURALINC2011 +
                            NATURALINC2012 + NATURALINC2013 + NATURALINC2014 +
                            NATURALINC2015 + NATURALINC2016 + NATURALINC2017 +
                            NATURALINC2018 + NATURALINC2019,
                          popchgiimm = INTERNATIONALMIG2010 + INTERNATIONALMIG2011 +
                            INTERNATIONALMIG2012 + INTERNATIONALMIG2013 +
                            INTERNATIONALMIG2014 + INTERNATIONALMIG2015 +
                            INTERNATIONALMIG2016 + INTERNATIONALMIG2017 +
                            INTERNATIONALMIG2018 + INTERNATIONALMIG2019,
                          popchgdimm = DOMESTICMIG2010 + DOMESTICMIG2011 +
                            DOMESTICMIG2012 + DOMESTICMIG2013 + DOMESTICMIG2014 +
                            DOMESTICMIG2015 + DOMESTICMIG2016 + DOMESTICMIG2017 +
                            DOMESTICMIG2018 + DOMESTICMIG2019) %>%
                   select(cfips,pop2019,popchgnat,popchgiimm,popchgdimm)

#Add population data to the county master file.
county_master <- inner_join(county_master,county_census,by="cfips")
```

3.1.8 Preprocess GDP File

The GDP file was a significant challenge due to its handling of independent cities in the state of Virginia. Virginia is unique in that any community that charters as a city is now independent of any county, including in some cases a county that completely surrounds it. These independent cities are treated as separate county equivalents by the U.S. Census Bureau, NOAA, the Yale Program on Climate Communications, and this project. However, the Bureau of Economic Analysis groups some independent cities in with their surrounding county, and treats the county+independent city as one entity for reporting purposes. When these Virginia combinations are added to one similar situation in Hawaii, there are a total of 23 such instances.

Other potential predictors had all been processed treating independent cities as county equivalents. To maintain the same number of counties, in these 23 instances the GDP value for the combined county+independent city was used as the GDP value of the independent city and of the surrounding county.

There were several varieties of GDP data in the file, based on inflation adjustments and other modifiers. Current-dollar GDP (as of 2019) was selected for analysis, since it is close in time to the climate change opinion data, and added to the **county_master** tibble.

```
###########################################
# Preprocess GDP file
###########################################
str(GDP_file)

#Need to filter out the non-county data (FIPS ends in "000")
#Need to only include lines from current GDP (line code = 3)
#Filter out some old counties with text <NA>s
#Exclude DC
county_gdp <- GDP_file %>%
                rename(cfips = GeoFIPS,
```

```r
                              cname = GeoName,
                              g2019 = `2019`,
                              g2009 = `2009`) %>%
                 filter(substr(cfips,3,5) != "000" &
                              LineCode == 3 &
                              substr(g2019,1,1) != "(" &
                              substr(cfips,1,2) != "11") %>%
                 mutate(gdp2019 = as.numeric(g2019),
                              gdp2009 = as.numeric(g2009),
                              gdpchg = gdp2019 - gdp2009) %>%
                 select(cfips,cname,gdp2019,gdpchg)

c2 <- anti_join(county_gdp,county_master,by="cfips")
c3 <- anti_join(county_master,county_gdp,by="cfips")

#The issue is with 1 2-county combo in Hawaii, and 23 county/independent city groupings
#in Virginia.  How to handle this?  Good question.  Climate data appears to be by the
#independent city, not combo, so maybe assign gdp values to both

#Process c2 records, split into components, get new cfips, then add them to county_gdp
newrecs <- data.frame(cname="Initial",gdp2019=-1,gdpchg=-1)
for (i in 1:nrow(c2)) {
  #Strip off last 5 characters from county name
  Gname <- substr(c2$cname[i],1,nchar(c2$cname[i])-5)
  #Combinations of 3 use a comma.  Replace it with "+"
  Gname <- str_replace(Gname,","," +")
  Gnames <- str_split(Gname," \\+ ",simplify=TRUE)
  #Create new records
  for (y in 1:dim(Gnames)[2]) {
    # Make new name upper case
    newname <- toupper(Gnames[,y])
    # If not Virginia, add "COUNTY" to end
    if (substr(c2$cfips[i],1,2) != "51") {
      newname <- paste(newname,"COUNTY",sep=" ")
    } else {
      #If Virginia, add "COUNTY" if first one, otherwise add "CITY"
        newname <- ifelse(y == 1,
                          paste(newname,"COUNTY",sep=" "),
                          paste(newname,"CITY",sep= " "))
    }
    newrec <- data.frame(cname = newname,
                         gdp2019 = c2$gdp2019[i],
                         gdpchg = c2$gdpchg[i])
    newrecs <- bind_rows(newrecs,newrec)
  }
}
newrecs <- newrecs[-1,]

#Correct for Fairfax City City
newrecs$cname[newrecs$cname=="FAIRFAX CITY CITY"] <- "FAIRFAX CITY"
#Get correct FIPS code for new records
newFIPS <- inner_join(newrecs,geocodes,by="cname")
#Filter for only VA and HI ones, select correct columns
```

```r
newFIPS <- newFIPS %>%
             filter(state=="51" | state=="15") %>%
             select(cfips,cname,gdp2019,gdpchg)

#Make sure they are all there
c4 <- anti_join(newrecs,newFIPS,by="cname")
nrow(c4)

#Add new split records to county GDP table
county_gdp <- bind_rows(county_gdp,newFIPS)

#Check to see if any don't match
c5 <- anti_join(county_gdp,county_master,by="cfips")
nrow(c5)

#Update county master with GDP data
county_gdp <- county_gdp %>% select(-cname)
county_master <- inner_join(county_master,county_gdp,by="cfips")
```

3.1.9 Preprocess Income File

Since personal income data came from the Bureau of Economic Analysis, it had the same issue that the GDP file had – data for 23 independent cities were combined with their surrounding county. This was resolved in the same manner as GDP, by using the personal income numbers for the combination as values for each of the components.

Of the several flavors of personal income available, per capita personal income was selected and added to the **county__master** tibble.

```r
###########################################
# Preprocess Income file
###########################################
str(INC_file)

#Need to filter out the non-county data (FIPS ends in "000")
#Need to only include lines with per capita income (line code = 3)
#Filter out some old counties with text <NA>s
#Exclude DC
county_inc <- INC_file %>%
  rename(cfips = GeoFIPS,
         cname = GeoName,
         i2019 = `2019`,
         i2009 = `2009`) %>%
  filter(substr(cfips,3,5) != "000" &
           LineCode == 3 &
           substr(i2019,1,1) != "(" &
           substr(cfips,1,2) != "11") %>%
  mutate(inc2019 = as.numeric(i2019),
         inc2009 = as.numeric(i2009),
         incchg = inc2019 - inc2009) %>%
  select(cfips,cname,inc2019,incchg)

i2 <- anti_join(county_inc,county_master,by="cfips")
```

```r
i3 <- anti_join(county_master,county_inc,by="cfips")

#The issue is with 1 2-county combo in Hawaii, and 23 county/independent city groupings
#in Virginia.  How to handle this?  Good question.  Climate data appears to be by the
#independent city, not combo, so maybe assign gdp values to both

#Process i2 records, split into components, get new cfips, then add them to county_inc
newrecs <- data.frame(cname="Initial",inc2019=-1,incchg=-1)
for (i in 1:nrow(i2)) {
  #Strip off last 5 characters from county name
  Gname <- substr(i2$cname[i],1,nchar(i2$cname[i])-5)
  #Combinations of 3 use a comma.  Replace it with "+"
  Gname <- str_replace(Gname,",",",+")
  Gnames <- str_split(Gname," \\+ ",simplify=TRUE)
  #Create new records
  for (y in 1:dim(Gnames)[2]) {
    # Make new name upper case
    newname <- toupper(Gnames[,y])
    # If not Virginia, add "COUNTY" to end
    if (substr(i2$cfips[i],1,2) != "51") {
      newname <- paste(newname,"COUNTY",sep=" ")
    } else {
      #If Virginia, add "COUNTY" if first one, otherwise add "CITY"
      newname <- ifelse(y == 1,
                        paste(newname,"COUNTY",sep=" "),
                        paste(newname,"CITY",sep= " "))
    }
    newrec <- data.frame(cname = newname,
                         inc2019 = i2$inc2019[i],
                         incchg = i2$incchg[i])
    newrecs <- bind_rows(newrecs,newrec)
  }
}
newrecs <- newrecs[-1,]

#Correct for Fairfax City City
newrecs$cname[newrecs$cname=="FAIRFAX CITY CITY"] <- "FAIRFAX CITY"
#Get correct FIPS code for new records
newFIPS <- inner_join(newrecs,geocodes,by="cname")
#Filter for only VA and HI ones, select correct columns
newFIPS <- newFIPS %>%
  filter(state=="51" | state=="15") %>%
  select(cfips,cname,inc2019,incchg)

#Make sure they are all there
i4 <- anti_join(newrecs,newFIPS,by="cname")
nrow(i4)

#Add new split records to county GDP table
county_inc <- bind_rows(county_inc,newFIPS)

#Check to see if any don't match
i5 <- anti_join(county_inc,county_master,by="cfips")
```

```r
nrow(i5)

#Update county master with GDP data
county_inc <- county_inc %>% select(-cname)
county_master <- inner_join(county_master,county_inc,by="cfips")
```

3.1.10 Preprocess Climate Files

Unlike the Excel or csv files loaded thus far, the NOAA climate files were lines of text 95 characters long, with a separate readme file describing the character positions for each column. This readme file can be found in ***Appendix 2***.

The first task was converting rows of text into data frame records with appropriate columns as specified by the layout file. This step also loaded the state codes used by NOAA, which were different from the CFIPS code, and built a cross-reference table.

```r
##########################################
# Convert climate files to data frames
##########################################

#Convert text files to data frames
convert_climate_file <- function (climfile) {
  statecode <- str_sub(climfile,1,2)
  countycode <- str_sub(climfile,3,5)
  elemcode <- str_sub(climfile,6,7)
  year <- str_sub(climfile,8,11)
  janval <- str_sub(climfile,12,18)
  febval <- str_sub(climfile,19,25)
  marval <- str_sub(climfile,26,32)
  aprval <- str_sub(climfile,33,39)
  mayval <- str_sub(climfile,40,46)
  junval <- str_sub(climfile,47,53)
  julval <- str_sub(climfile,54,60)
  augval <- str_sub(climfile,61,67)
  sepval <- str_sub(climfile,68,74)
  octval <- str_sub(climfile,75,81)
  novval <- str_sub(climfile,82,88)
  decval <- str_sub(climfile,89,95)
  clim_tab <- data.frame(statecode = statecode,
                         countycode = countycode,
                         elemcode = elemcode,
                         year = year,
                         janval = janval,
                         febval = febval,
                         marval = marval,
                         aprval = aprval,
                         mayval = mayval,
                         junval = junval,
                         julval = julval,
                         augval = augval,
                         sepval = sepval,
                         octval = octval,
                         novval = novval,
                         decval = decval)
  clim_tab
}

county_temp <- convert_climate_file(TEMP_file)
county_tmax <- convert_climate_file(TMAX_file)
county_tmin <- convert_climate_file(TMIN_file)
county_prcp <- convert_climate_file(PRCP_file)

#Readme file contains the state codes used -- read it in and build a table so we can get
#the correct FIPS codes

#We only care about lines 74-100
CLIMREADME_file <- CLIMREADME_file[74:100]

#Get the state codes out of their two columns and into a workable format
cs1 <- str_sub(CLIMREADME_file,30,56)
climstates <- data.frame(statecode = str_sub(cs1,1,2),
                         sname = toupper(trimws(str_sub(cs1,4,20))))
cs2 <- str_sub(CLIMREADME_file,57,80)
climstates2 <- data.frame(statecode = str_sub(cs2,1,2),
                          sname = toupper(trimws(str_sub(cs2,4,20))))
climstates <- bind_rows(climstates,climstates2)

#Remove blank rows
climstates <- climstates[!apply(climstates=="",1,all),]

#Add record for Alaska (state code 50 in climate data)
climalaska <- data.frame(statecode="50", sname="ALASKA")
climstates <- bind_rows(climstates,climalaska)

#Cross-reference with geostate to get the correct state code
climstates <- inner_join(climstates,geostate,by="sname")
```

Note that despite the layout file listing codes for only the original 48 states, there was also data present for Alaska (state code 50). There was no data present for Hawaii (more on this later).

The next task was determining how to filter and structure the data in the climate files into usable predictors. For each county in each of the four climate files, there is one row per year going back as far as 1895. Each row contains twelve columns of data, one for each month of that year. The data in each column shows the mean value of that file's climate statistic for that county in that month in that year.

The following principles were used to determine potential predictors.

1. Since climate change emerged in the popular consciousness in the 1980s, any data older than 1980 was excluded.
2. Since there was not a complete record for 2020 when this project began, no 2020 data was included.
3. Monthly data was aggregated and averaged into four season buckets, using the climatological season definitions (Spring=March-May, Summer=June-August, Fall=September-November,Winter = December - February). This required some complex wrangling, since two months of the winter season data for a given year was stored in the Jan-Feb columns of the following year's record.
4. 2019 data was included, to see if the most recent full year would have any predictive ability.
5. Predictors based on four windows of time were calculated, to consider the effect of climate changes within those windows on climate change opinion. These four windows were used:

- Climate statistic change, 2009-2019
- Climate statistic change, 1999-2019
- Climate statistic change, 1989-2019
- Climate statistic change, 1980-2019

The extensive block of code below performs the wrangling activities necessary to implement these principles across each of the four climate files. When completed, this added 80 potential predictors to the **county_master** tibble:

5 climate statistics * **4** seasons * **4** climate file types

The NOAA county identifiers were converted to the CFIPS identifiers during this wrangling process.

```r
##########################################
# Convert climate temperature file
##########################################

#Need to filter for the correct years, define values by season, get the fips codes
#Which years?
#Spring: March of year - May of year
#Summer: June of year -- August of year
#Fall: Sep of year -- Nove of year
#Winter: Dec of year -- Feb of next year

county_temp <- county_temp %>%
                 filter(year >= "1980") %>%
                 mutate(year = as.numeric(year),
                        winter23temp = as.numeric(janval) + as.numeric(febval),
                        springtemp = ((as.numeric(marval) + as.numeric(aprval) +
                                  as.numeric(mayval)) / 3),
                        summertemp = ((as.numeric(junval) + as.numeric(julval) +
                                  as.numeric(augval)) / 3),
                        falltemp = ((as.numeric(sepval) + as.numeric(octval) +
                                as.numeric(novval)) / 3  ),
                        winter1temp = as.numeric(decval))

#Get correct state code
county_temp <- inner_join(county_temp,climstates,by="statecode")

#Format the cfips identifier
county_temp <- county_temp %>%
                 mutate(cfips = paste(state,countycode,sep="")) %>%
                 select(cfips, year, winter23temp, springtemp,
                        summertemp, falltemp, winter1temp)

#Fix the "winter crossing year" problem
#Add "winter year" which is the year the Jan and Feb values should be recorded in
county_temp <- county_temp %>%
                 mutate(wintyear = year-1)
#Create a separate file showing Jan/Feb by winter year
county_temp_janfeb <- county_temp %>%
                 filter(wintyear >= 1980) %>%
                 select(-year) %>%
                 rename(janfebval = winter23temp,
                        year = wintyear) %>%
                 select(cfips,year,janfebval)
#Link main file to Jan/Feb by winter year to get the right values.
county_temp <- left_join(county_temp,county_temp_janfeb,by=c("cfips","year"))
#Now calculate winter values
county_temp <- county_temp %>%
                 mutate(wintertemp = (janfebval + winter1temp)/3) %>%
                 select(cfips, year, springtemp, summertemp, falltemp, wintertemp)

#Will look at current temps, change since 1980, change since 1989, change since 1999,
#change since 2009
county_temp1980 <- county_temp %>%
                     filter(year == 1980) %>%
                     rename(spring1980 = springtemp,
                            summer1980 = summertemp,
                            fall1980 = falltemp,
                            winter1980 = wintertemp)
county_temp1989 <- county_temp %>%
                     filter(year == 1989) %>%
                     rename(spring1989 = springtemp,
                            summer1989 = summertemp,
                            fall1989 = falltemp,
                            winter1989 = wintertemp)
county_temp1999 <- county_temp %>%
                     filter(year == 1999) %>%
                     rename(spring1999 = springtemp,
                            summer1999 = summertemp,
                            fall1999 = falltemp,
                            winter1999 = wintertemp)
county_temp2009 <- county_temp %>%
                     filter(year == 2009) %>%
                     rename(spring2009 = springtemp,
                            summer2009 = summertemp,
                            fall2009 = falltemp,
                            winter2009 = wintertemp)
county_temp2019 <- county_temp %>%
                     filter(year == 2019) %>%
                     rename(spring2019 = springtemp,
                     summer2019 = summertemp,
                     fall2019 = falltemp,
                     winter2019 = wintertemp)

county_temp_summ <- inner_join(county_temp2019,county_temp2009,by="cfips")
county_temp_summ <- inner_join(county_temp_summ,county_temp1999,by="cfips")
county_temp_summ <- inner_join(county_temp_summ,county_temp1989,by="cfips")
county_temp_summ <- inner_join(county_temp_summ,county_temp1980,by="cfips")
county_temp_summ <- county_temp_summ %>%
                      mutate(sp2019temp = spring2019,
                             su2019temp = summer2019,
                             fa2019temp = fall2019,
                             wi2019temp = winter2019,
                             sptempd2009 = spring2019 - spring2009,
                             sutempd2009 = summer2019 - summer2009,
                             fatempd2009 = fall2019 - fall2009,
                             witempd2009 = winter2019 - winter2009,
                             sptempd1999 = spring2019 - spring1999,
                             sutempd1999 = summer2019 - summer1999,
                             fatempd1999 = fall2019 - fall1999,
                             witempd1999 = winter2019 - winter1999,
                             sptempd1989 = spring2019 - spring1989,
                             sutempd1989 = summer2019 - summer1989,
                             fatempd1989 = fall2019 - fall1989,
                             witempd1989 = winter2019 - winter1989,
                             sptempd1980 = spring2019 - spring1980,
                             sutempd1980 = summer2019 - summer1980,
                             fatempd1980 = fall2019 - fall1980,
                             witempd1980 = winter2019 - winter1980) %>%
                      select(cfips,
                             sp2019temp,
                             su2019temp,
                             fa2019temp,
                             wi2019temp,
                             sptempd2009,
                             sutempd2009,
                             fatempd2009,
                             witempd2009,
                             sptempd1999,
                             sutempd1999,
                             fatempd1999,
                             witempd1999,
                             sptempd1989,
                             sutempd1989,
                             fatempd1989,
                             witempd1989,
                             sptempd1980,
                             sutempd1980,
                             fatempd1980,
                             witempd1980)

###########################################
# Convert climate max temperature file
###########################################

#Need to filter for the correct years, define values by season, get the fips codes
#Which years?
#Spring: March of year - May of year
#Summer: June of year -- August of year
#Fall: Sep of year -- Nove of year
#Winter: Dec of year -- Feb of next year

county_tmax <- county_tmax %>%
  filter(year >= "1980") %>%
  mutate(year = as.numeric(year),
         winter23tmax = as.numeric(janval) + as.numeric(febval),
         springtmax = ((as.numeric(marval) + as.numeric(aprval) +
                          as.numeric(mayval)) / 3),
         summertmax = ((as.numeric(junval) + as.numeric(julval) +
                          as.numeric(augval)) / 3),
         falltmax = ((as.numeric(sepval) + as.numeric(octval) +
                        as.numeric(novval)) / 3  ),
         winter1tmax = as.numeric(decval))

#Get correct state code
county_tmax <- inner_join(county_tmax,climstates,by="statecode")

#Format the cfips identifier
county_tmax <- county_tmax %>%
  mutate(cfips = paste(state,countycode,sep="")) %>%
  select(cfips, year, winter23tmax, springtmax,
         summertmax, falltmax, winter1tmax)

#Fix the "winter crossing year" problem
#Add "winter year" which is the year the Jan and Feb values should be recorded in
county_tmax <- county_tmax %>%
  mutate(wintyear = year-1)
#Create a separate file showing Jan/Feb by winter year
county_tmax_janfeb <- county_tmax %>%
  filter(wintyear >= 1980) %>%
  select(-year) %>%
  rename(janfebval = winter23tmax,
         year = wintyear) %>%
  select(cfips,year,janfebval)
#Link main file to Jan/Feb by winter year to get the right values.
county_tmax <- left_join(county_tmax,county_tmax_janfeb,by=c("cfips","year"))
#Now calculate winter values
county_tmax <- county_tmax %>%
  mutate(wintertmax = (janfebval + winter1tmax)/3) %>%
  select(cfips, year, springtmax, summertmax, falltmax, wintertmax)

#Will look at current temps, change since 1980, change since 1989, change since 1999,
#change since 2009
county_tmax1980 <- county_tmax %>%
  filter(year == 1980) %>%
  rename(spring1980 = springtmax,
         summer1980 = summertmax,
         fall1980 = falltmax,
         winter1980 = wintertmax)
county_tmax1989 <- county_tmax %>%
  filter(year == 1989) %>%
  rename(spring1989 = springtmax,
         summer1989 = summertmax,
         fall1989 = falltmax,
         winter1989 = wintertmax)
county_tmax1999 <- county_tmax %>%
  filter(year == 1999) %>%
  rename(spring1999 = springtmax,
         summer1999 = summertmax,
         fall1999 = falltmax,
         winter1999 = wintertmax)
county_tmax2009 <- county_tmax %>%
```

```r
  filter(year == 2009) %>%
  rename(spring2009 = springtmax,
         summer2009 = summertmax,
         fall2009 = falltmax,
         winter2009 = wintertmax)
county_tmax2019 <- county_tmax %>%
  filter(year == 2019) %>%
  rename(spring2019 = springtmax,
         summer2019 = summertmax,
         fall2019 = falltmax,
         winter2019 = wintertmax)

county_tmax_summ <- inner_join(county_tmax2019,county_tmax2009,by="cfips")
county_tmax_summ <- inner_join(county_tmax_summ,county_tmax1999,by="cfips")
county_tmax_summ <- inner_join(county_tmax_summ,county_tmax1989,by="cfips")
county_tmax_summ <- inner_join(county_tmax_summ,county_tmax1980,by="cfips")
county_tmax_summ <- county_tmax_summ %>%
  mutate(sp2019tmax = spring2019,
         su2019tmax = summer2019,
         fa2019tmax = fall2019,
         wi2019tmax = winter2019,
         sptmaxd2009 = spring2019 - spring2009,
         sutmaxd2009 = summer2019 - summer2009,
         fatmaxd2009 = fall2019 - fall2009,
         witmaxd2009 = winter2019 - winter2009,
         sptmaxd1999 = spring2019 - spring1999,
         sutmaxd1999 = summer2019 - summer1999,
         fatmaxd1999 = fall2019 - fall1999,
         witmaxd1999 = winter2019 - winter1999,
         sptmaxd1989 = spring2019 - spring1989,
         sutmaxd1989 = summer2019 - summer1989,
         fatmaxd1989 = fall2019 - fall1989,
         witmaxd1989 = winter2019 - winter1989,
         sptmaxd1980 = spring2019 - spring1980,
         sutmaxd1980 = summer2019 - summer1980,
         fatmaxd1980 = fall2019 - fall1980,
         witmaxd1980 = winter2019 - winter1980) %>%
  select(cfips,
         sp2019tmax,
         su2019tmax,
         fa2019tmax,
         wi2019tmax,
         sptmaxd2009,
         sutmaxd2009,
         fatmaxd2009,
         witmaxd2009,
         sptmaxd1999,
         sutmaxd1999,
         fatmaxd1999,
         witmaxd1999,
         sptmaxd1989,
         sutmaxd1989,
         fatmaxd1989,
         witmaxd1989,
         sptmaxd1980,
         sutmaxd1980,
         fatmaxd1980,
         witmaxd1980)

##########################################
# Convert climate min temperature file
##########################################

#Need to filter for the correct years, define values by season, get the fips codes
#Which years?
#Spring: March of year - May of year
#Summer: June of year -- August of year
#Fall: Sep of year -- Nove of year
#Winter: Dec of year -- Feb of next year

county_tmin <- county_tmin %>%
  filter(year >= "1980") %>%
  mutate(year = as.numeric(year),
         winter23tmin = as.numeric(janval) + as.numeric(febval),
         springtmin = ((as.numeric(marval) + as.numeric(aprval) +
                          as.numeric(mayval)) / 3),
         summertmin = ((as.numeric(junval) + as.numeric(julval) +
                          as.numeric(augval)) / 3),
         falltmin = ((as.numeric(sepval) + as.numeric(octval) +
                        as.numeric(novval)) / 3  ),
         winter1tmin = as.numeric(decval))

#Get correct state code
county_tmin <- inner_join(county_tmin,climstates,by="statecode")

#Format the cfips identifier
county_tmin <- county_tmin %>%
  mutate(cfips = paste(state,countycode,sep="")) %>%
  select(cfips, year, winter23tmin, springtmin,
         summertmin, falltmin, winter1tmin)

#Fix the "winter crossing year" problem
#Add "winter year" which is the year the Jan and Feb values should be recorded in
county_tmin <- county_tmin %>%
  mutate(wintyear = year-1)
#Create a separate file showing Jan/Feb by winter year
county_tmin_janfeb <- county_tmin %>%
  filter(wintyear >= 1980) %>%
  select(-year) %>%
  rename(janfebval = winter23tmin,
         year = wintyear) %>%
  select(cfips,year,janfebval)
#Link main file to Jan/Feb by winter year to get the right values.
county_tmin <- left_join(county_tmin,county_tmin_janfeb,by=c("cfips","year"))
#Now calculate winter values
county_tmin <- county_tmin %>%
```

```r
  mutate(wintertmin = (janfebval + winter1tmin)/3) %>%
  select(cfips, year, springtmin, summertmin, falltmin, wintertmin)

#Will look at current temps, change since 1980, change since 1989, change since 1999,
#change since 2009
county_tmin1980 <- county_tmin %>%
  filter(year == 1980) %>%
  rename(spring1980 = springtmin,
         summer1980 = summertmin,
         fall1980 = falltmin,
         winter1980 = wintertmin)
county_tmin1989 <- county_tmin %>%
  filter(year == 1989) %>%
  rename(spring1989 = springtmin,
         summer1989 = summertmin,
         fall1989 = falltmin,
         winter1989 = wintertmin)
county_tmin1999 <- county_tmin %>%
  filter(year == 1999) %>%
  rename(spring1999 = springtmin,
         summer1999 = summertmin,
         fall1999 = falltmin,
         winter1999 = wintertmin)
county_tmin2009 <- county_tmin %>%
  filter(year == 2009) %>%
  rename(spring2009 = springtmin,
         summer2009 = summertmin,
         fall2009 = falltmin,
         winter2009 = wintertmin)
county_tmin2019 <- county_tmin %>%
  filter(year == 2019) %>%
  rename(spring2019 = springtmin,
         summer2019 = summertmin,
         fall2019 = falltmin,
         winter2019 = wintertmin)

county_tmin_summ <- inner_join(county_tmin2019,county_tmin2009,by="cfips")
county_tmin_summ <- inner_join(county_tmin_summ,county_tmin1999,by="cfips")
county_tmin_summ <- inner_join(county_tmin_summ,county_tmin1989,by="cfips")
county_tmin_summ <- inner_join(county_tmin_summ,county_tmin1980,by="cfips")
county_tmin_summ <- county_tmin_summ %>%
  mutate(sp2019tmin = spring2019,
         su2019tmin = summer2019,
         fa2019tmin = fall2019,
         wi2019tmin = winter2019,
         sptmind2009 = spring2019 - spring2009,
         sutmind2009 = summer2019 - summer2009,
         fatmind2009 = fall2019 - fall2009,
         witmind2009 = winter2019 - winter2009,
         sptmind1999 = spring2019 - spring1999,
         sutmind1999 = summer2019 - summer1999,
         fatmind1999 = fall2019 - fall1999,
         witmind1999 = winter2019 - winter1999,
         sptmind1989 = spring2019 - spring1989,
         sutmind1989 = summer2019 - summer1989,
         fatmind1989 = fall2019 - fall1989,
         witmind1989 = winter2019 - winter1989,
         sptmind1980 = spring2019 - spring1980,
         sutmind1980 = summer2019 - summer1980,
         fatmind1980 = fall2019 - fall1980,
         witmind1980 = winter2019 - winter1980) %>%
  select(cfips,
         sp2019tmin,
         su2019tmin,
         fa2019tmin,
         wi2019tmin,
         sptmind2009,
         sutmind2009,
         fatmind2009,
         witmind2009,
         sptmind1999,
         sutmind1999,
         fatmind1999,
         witmind1999,
         sptmind1989,
         sutmind1989,
         fatmind1989,
         witmind1989,
         sptmind1980,
         sutmind1980,
         fatmind1980,
         witmind1980)

###########################################
# Convert climate precipitation file
###########################################

#Need to filter for the correct years, define values by season, get the fips codes
#Which years?
#Spring: March of year - May of year
#Summer: June of year -- August of year
#Fall: Sep of year -- Nove of year
#Winter: Dec of year -- Feb of next year

county_prcp <- county_prcp %>%
  filter(year >= "1980") %>%
  mutate(year = as.numeric(year),
         winter23prcp = as.numeric(janval) + as.numeric(febval),
         springprcp = ((as.numeric(marval) + as.numeric(aprval) +
                          as.numeric(mayval)) / 3),
         summerprcp = ((as.numeric(junval) + as.numeric(julval) +
                          as.numeric(augval)) / 3),
         fallprcp = ((as.numeric(sepval) + as.numeric(octval) +
                        as.numeric(novval)) / 3  ),
         winter1prcp = as.numeric(decval))
```

```r
#Get correct state code
county_prcp <- inner_join(county_prcp,climstates,by="statecode")

#Format the cfips identifier
county_prcp <- county_prcp %>%
  mutate(cfips = paste(state,countycode,sep="")) %>%
  select(cfips, year, winter23prcp, springprcp,
         summerprcp, fallprcp, winter1prcp)

#Fix the "winter crossing year" problem
#Add "winter year" which is the year the Jan and Feb values should be recorded in
county_prcp <- county_prcp %>%
  mutate(wintyear = year-1)
#Create a separate file showing Jan/Feb by winter year
county_prcp_janfeb <- county_prcp %>%
  filter(wintyear >= 1980) %>%
  select(-year) %>%
  rename(janfebval = winter23prcp,
         year = wintyear) %>%
  select(cfips,year,janfebval)
#Link main file to Jan/Feb by winter year to get the right values.
county_prcp <- left_join(county_prcp,county_prcp_janfeb,by=c("cfips","year"))
#Now calculate winter values
county_prcp <- county_prcp %>%
  mutate(winterprcp = (janfebval + winter1prcp)/3) %>%
  select(cfips, year, springprcp, summerprcp, fallprcp, winterprcp)

#Will look at current temps, change since 1980, change since 1989, change since 1999,
#change since 2009
county_prcp1980 <- county_prcp %>%
  filter(year == 1980) %>%
  rename(spring1980 = springprcp,
         summer1980 = summerprcp,
         fall1980 = fallprcp,
         winter1980 = winterprcp)
county_prcp1989 <- county_prcp %>%
  filter(year == 1989) %>%
  rename(spring1989 = springprcp,
         summer1989 = summerprcp,
         fall1989 = fallprcp,
         winter1989 = winterprcp)
county_prcp1999 <- county_prcp %>%
  filter(year == 1999) %>%
  rename(spring1999 = springprcp,
         summer1999 = summerprcp,
         fall1999 = fallprcp,
         winter1999 = winterprcp)
county_prcp2009 <- county_prcp %>%
  filter(year == 2009) %>%
  rename(spring2009 = springprcp,
         summer2009 = summerprcp,
         fall2009 = fallprcp,
         winter2009 = winterprcp)
```

```r
county_prcp2019 <- county_prcp %>%
  filter(year == 2019) %>%
  rename(spring2019 = springprcp,
         summer2019 = summerprcp,
         fall2019 = fallprcp,
         winter2019 = winterprcp)

county_prcp_summ <- inner_join(county_prcp2019,county_prcp2009,by="cfips")
county_prcp_summ <- inner_join(county_prcp_summ,county_prcp1999,by="cfips")
county_prcp_summ <- inner_join(county_prcp_summ,county_prcp1989,by="cfips")
county_prcp_summ <- inner_join(county_prcp_summ,county_prcp1980,by="cfips")
county_prcp_summ <- county_prcp_summ %>%
  mutate(sp2019prcp = spring2019,
         su2019prcp = summer2019,
         fa2019prcp = fall2019,
         wi2019prcp = winter2019,
         spprcpd2009 = spring2019 - spring2009,
         suprcpd2009 = summer2019 - summer2009,
         faprcpd2009 = fall2019 - fall2009,
         wiprcpd2009 = winter2019 - winter2009,
         spprcpd1999 = spring2019 - spring1999,
         suprcpd1999 = summer2019 - summer1999,
         faprcpd1999 = fall2019 - fall1999,
         wiprcpd1999 = winter2019 - winter1999,
         spprcpd1989 = spring2019 - spring1989,
         suprcpd1989 = summer2019 - summer1989,
         faprcpd1989 = fall2019 - fall1989,
         wiprcpd1989 = winter2019 - winter1989,
         spprcpd1980 = spring2019 - spring1980,
         suprcpd1980 = summer2019 - summer1980,
         faprcpd1980 = fall2019 - fall1980,
         wiprcpd1980 = winter2019 - winter1980) %>%
  select(cfips,
         sp2019prcp,
         su2019prcp,
         fa2019prcp,
         wi2019prcp,
         spprcpd2009,
         suprcpd2009,
         faprcpd2009,
         wiprcpd2009,
         spprcpd1999,
         suprcpd1999,
         faprcpd1999,
         wiprcpd1999,
         spprcpd1989,
         suprcpd1989,
         faprcpd1989,
         wiprcpd1989,
         spprcpd1980,
         suprcpd1980,
         faprcpd1980,
         wiprcpd1980)
```

```
############################################
# Add climate data to county master
############################################
county_master <- inner_join(county_master,county_temp_summ,by="cfips")
county_master <- inner_join(county_master,county_tmax_summ,by="cfips")
county_master <- inner_join(county_master,county_tmin_summ,by="cfips")
county_master <- inner_join(county_master,county_prcp_summ,by="cfips")

#This eliminates the 6 county equivalents that don't have climate data
#All five Hawaii counties and the independent city of Lexington VA
```

As stated earlier, there were 3,141 counties in the **county__master** tibble. The NOAA climate files contained information for 3,135 counties – the six missing counties were all five of Hawaii's counties plus one independent city (Lexington) in Virginia. No alternative climate data sources for these counties could be found. For this reason, these six counties were excluded for further analysis, and no attempt was made to predict their climate change opinion.

3.1.11 Preprocess Climate Opinion Data

As discussed earlier, out of the 31 climate statements in the file from the Yale Program for Climate Communications, the one selected for prediction was "Global warming is caused mostly by human activities." In the interest of brevity, this statement will often be shortened to "Human Caused" when referenced going forward.

the following code added the esimated support percentages by county for the "Human Caused" statement to the **county-master** tibble.

```
########################################################
# Preprocess Yale Climate Communication opinion file
########################################################
str(YCOM_file)

#The Yale file includes opinion data at state, county, congressional district, and metro
#area levels.  We only want the county ones, so filter the others out.  Also filter out
#the District of Columbia.  Also, the GEOID field is the FIPS code, but stored as numeric.
#Convert to character, and adjust for missing leading zeros.
clim_opin <- YCOM_file %>%
               filter(GeoType=="County" & GEOID != 11001) %>%
               mutate(cfips = as.character(GEOID))

clim_opin$cfips <- ifelse(nchar(clim_opin$cfips)==4,
                            paste("0",clim_opin$cfips,sep=""),
                            clim_opin$cfips)

#Select only the 2 questions we want to consider
clim_opin <- clim_opin %>%
               select(cfips, human, personal)

#Add to county master file
county_master <- inner_join(county_master,clim_opin,by="cfips")
```

3.2 Data Exploration

3.2.1 "Human Caused" Climate Change Opinions

Here are some basic statistics along with a histogram of the support percentages for the "Human Caused" statement:

```
#Range of human-caused opinions
min(county_master$human)  #Minimum county support percentage
```

```
## [1] 38.6
```

```
mean(county_master$human) #Mean county support percentage
```

```
## [1] 51
```

```
max(county_master$human)  #Maximum county support percentage
```

```
## [1] 74.2
```

```
sd(county_master$human)   #Std Deviation of county support percentage
```

```
## [1] 5.21
```

```
quantile(county_master$human)  #25% quantiles
```

```
##   0%  25%  50%  75% 100% 
## 38.6 47.4 50.1 53.6 74.2
```

```
ggplot(aes(human),data=county_master,color="blue") +
  geom_histogram(binwidth=1,fill="blue",alpha=0.5) +
  xlab("'Human Caused' Support Percentage")
```

As these statistics show, most counties have a support percentage for "Human Caused" that falls within the 47-53% range. There is a more pronounced "tail" on the right side of the graph, with the maximum value being about 4.4 standard deviations above the mean, while the minimum value is only 2.4 standard deviations below the mean.

3.2.2 Climate Predictors

Given that climate is a key focus of this project, it makes sense to look at trends in climate predictors over time. The following set of graphs illustrate changes in the overall means across all counties for climate metrics using data from 1980,1989,1999,2009, and 2019. This is not helpful for determining the impact of climate change at a local level, but will show any trends or degree of volatility in these overall metrics.

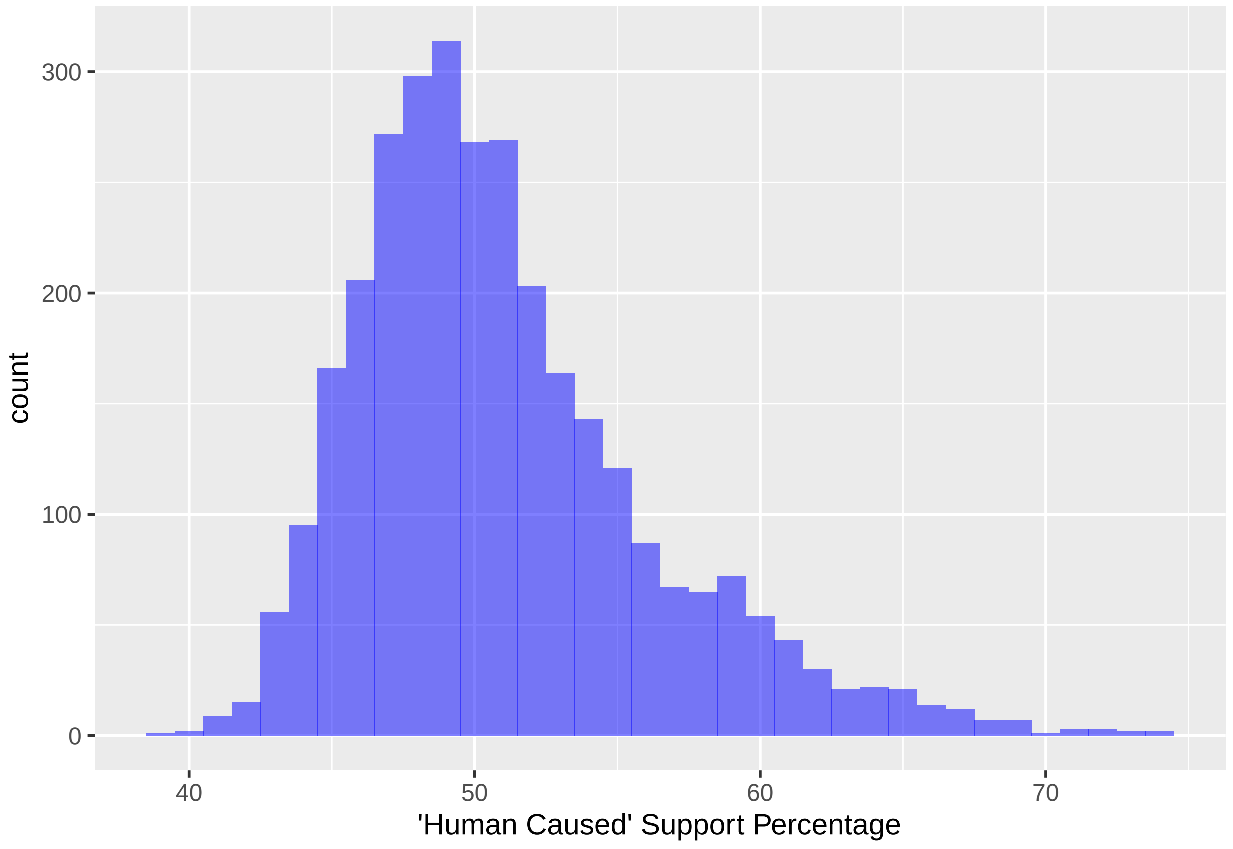
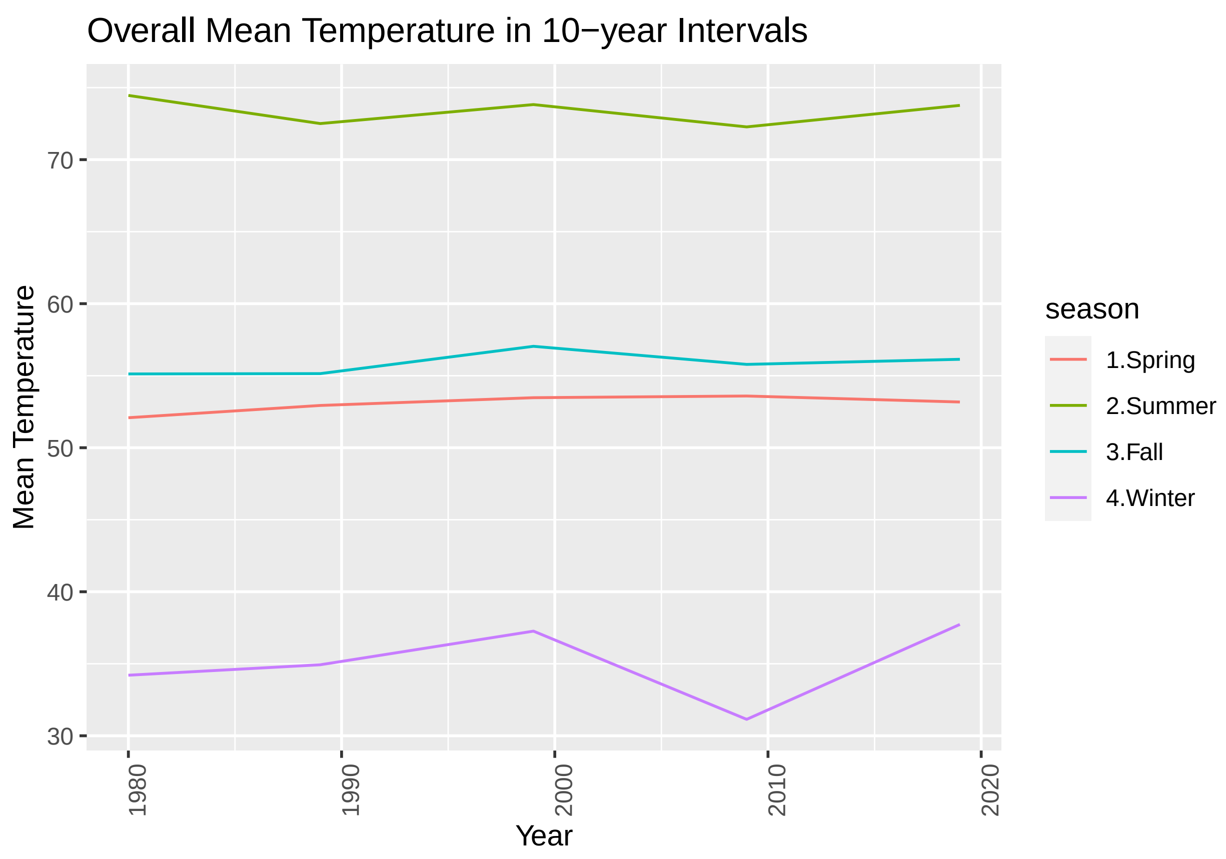
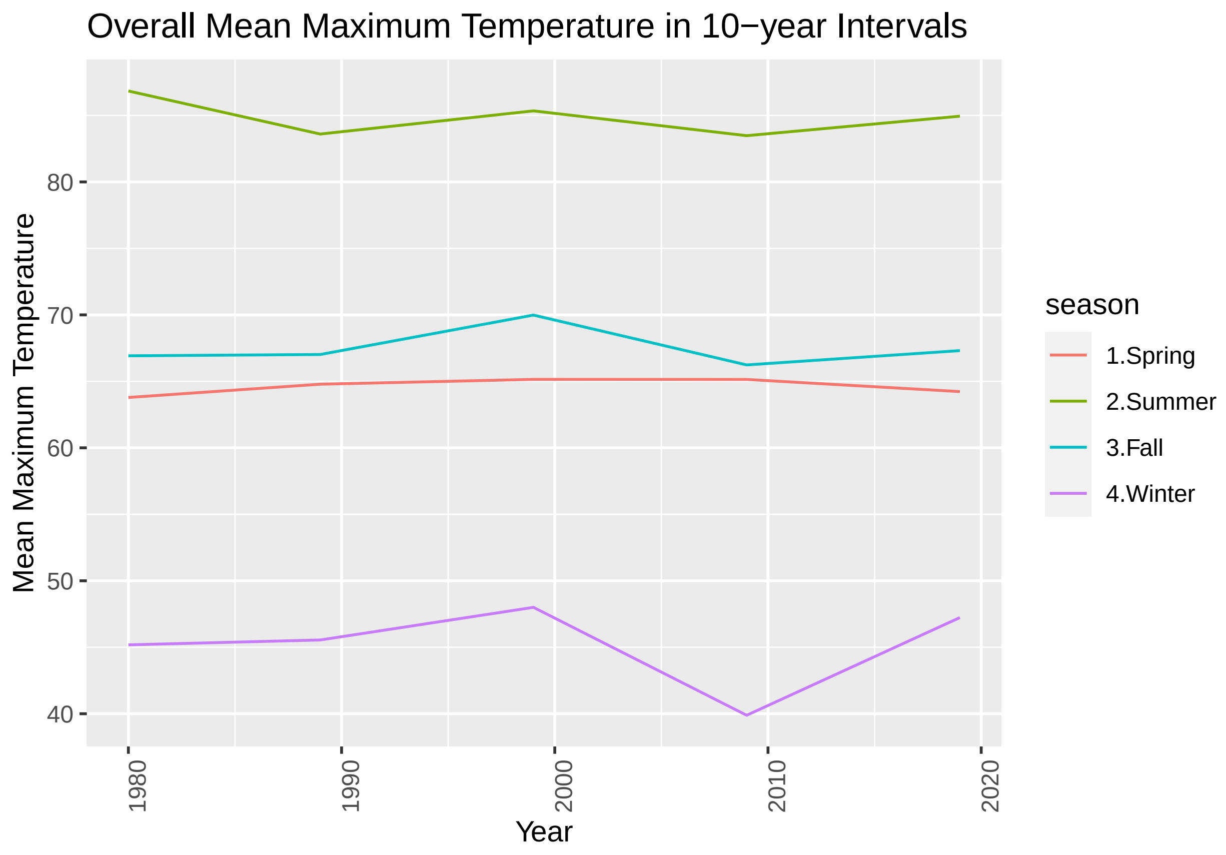
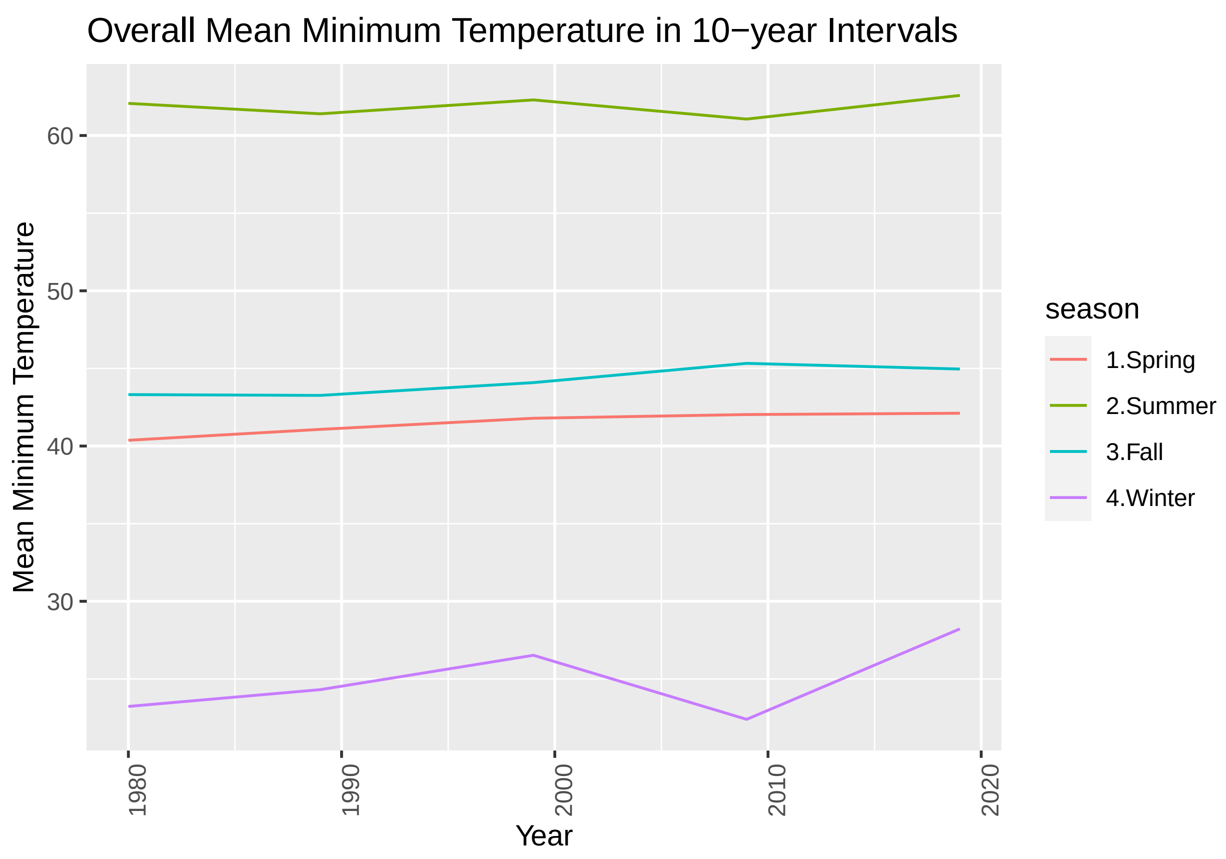
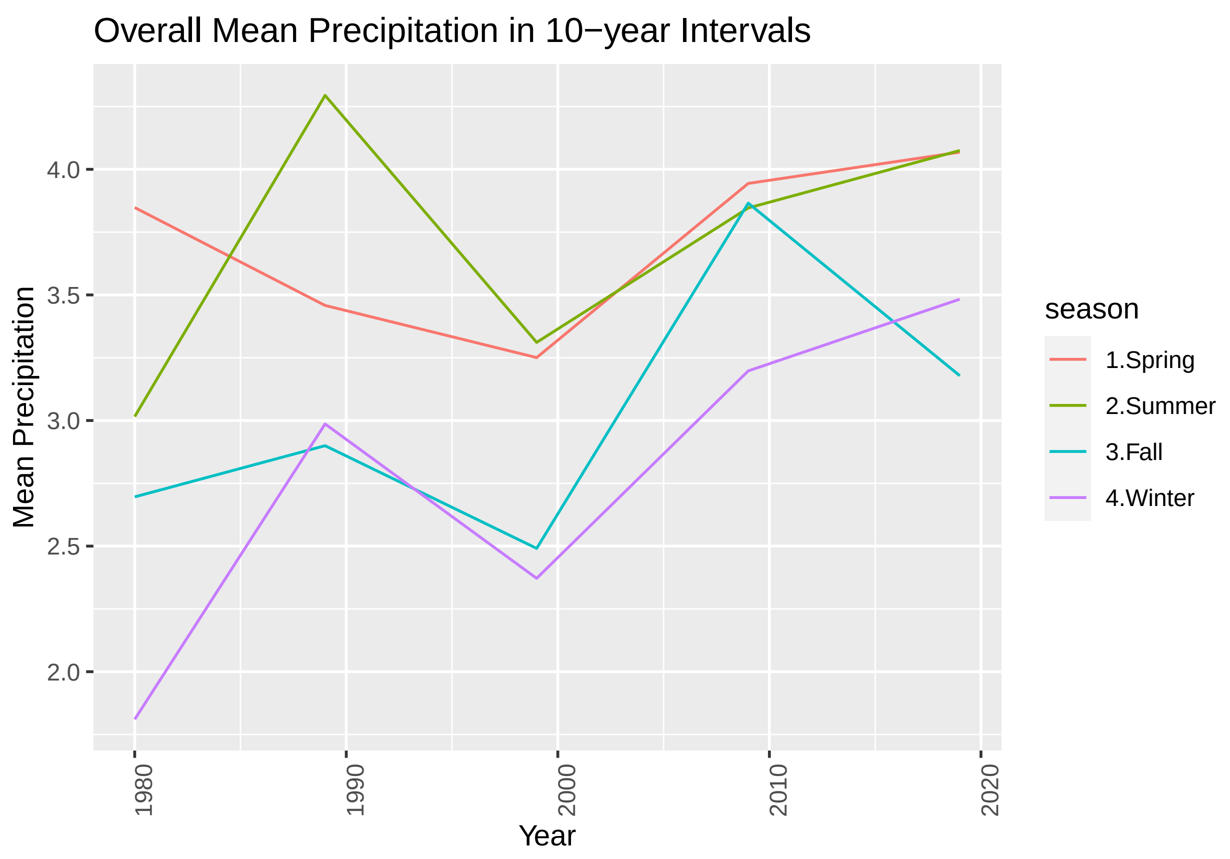


Figure 1: Climate Statement Histogram


Overall Mean Temperature in 10−year Intervals
Mean Temperature
70
60
50
40
30
1980
1990
2000
2010
2020
Year
season
1.Spring
2.Summer
3.Fall
4.Winter


Overall Mean Maximum Temperature in 10−year Intervals
Mean Maximum Temperature
80
70
60
50
40
1980
1990
2000
2010
2020
Year
season
1.Spring
2.Summer
3.Fall
4.Winter


Overall Mean Minimum Temperature in 10−year Intervals
Mean Minimum Temperature
60
50
40
30
1980
1990
2000
2010
2020
Year
season
1.Spring
2.Summer
3.Fall
4.Winter



For the three temperature-related predictors, there is relatively little change in the spring and fall, some moderate up-and-down in the summer, and some noticeable volatility in the winter, especially over the last twenty years. Precipitation is considerably more volatile across all seasons.

3.2.3 Binary Predictors

Three binary predictors were included in the **county__master** tibble. Here are statistics about the frequency of coastal counties, coal counties, and oil/gas counties, as well as the differences between their mean support for the "Human Caused" statement vs their non-equivalents.

Table 1: Binary Predictor Prevalence & Opinion Differences

Predictor	# of Counties	Proportion	Mean Support%	Diff from Overall
Coastal counties	250	0.080	56.2	5.24
Coal counties	151	0.048	47.8	-3.16
Oil/Gas counties	315	0.100	49.3	-1.73

There appears to be greater support for the "Human Caused" statement among coastal counties. There appears to be less support for the statement among coal-producing counties, although only 4.8% of counties are coal producers so the effect may be limited. The mean support from oil and gas producing counties is slightly less than the overall mean.

3.3 Predictor Analysis and Model Building

The process of building, analyzing, and refining the model for the "Human Caused" statement contains the following steps:

1. Divide the data into training, test, and validation sets.
2. Make an initial baseline prediction using the overall mean percentage of support for the "Human Caused" statement.
3. Examine population-related predictors, and build a model to predict support using the best candidates among them.
4. Examine economic-related predictors, and build a model to predict support using the best candidates among them.
5. Examine climate-related predictors, and build a model to predict support using the best candidates among them.
6. Combine the best population, economic, and climate-related predictors into a final model.

The **county_master** tibble contains 95 potential predictors, which was judged as too many given the time constraints of the project. Two criteria were used to reduce the number of predictors to a more reasonable number of stronger candidates:

1. Correlation scores were calculated between the "Human Caused" support percentage and each predictor, and any predictor which did not have a correlation <= -0.2 or >= 0.2 was excluded.
2. The **findCorrelation** function was used to identify predictors that were highly correlated with other predictors, using a cutoff value of 0.8 or higher. For each pair of such predictors, one of the pair was removed.

At each step of model building, the following three types of algorithms were considered:

1. Linear regression (as a simple and easy to understand method)
2. K-nearest neighbor (since the data set is relatively small, and since there is likely some degree of commonality among counties with similar characteristics)
3. Random forest (can handle larger numbers of predictors, and includes cross-validation)

Given that the support percentage for "Human Caused" is a continuous outcome, Root Mean Square Error (RMSE) was selected as the loss function to evaluate each algorithm's success.

3.3.1 Data Partitioning

Ten percent of the 3,135 counties were randomly partitioned into a validation set to check the performance of the final model. The remaining 90 percent were split into a training set (80%) for building models and a test set (20%) for evaluating models. A loss function was created to calculate RMSE for model evaluation.

```r
#Partitions for "global warming is caused mostly by human activity"
set.seed(321, sample.kind="Rounding")
county_val_index <- createDataPartition(
  y = county_master$human, times = 1, p = 0.1, list = FALSE)
human_model <- county_master[-county_val_index,]
human_val <- county_master[county_val_index,]
county_test_index <- createDataPartition(
  y = human_model$human, times = 1, p = 0.2, list = FALSE)
human_train <- human_model[-county_test_index,]
```

```r
human_test <- human_model[county_test_index,]


#Define loss function (RMSE)
RMSE_f <- function(predicted,actual){
  RMSE <- sqrt(mean((actual - predicted)^2))
}
```

3.3.2 Baseline prediction

The overall mean support for the "Human Caused" statement across all counties was used to make a baseline prediction. This provided a benchmark for measuring the effectiveness of subsequent models.

```r
#Baseline prediction
pred1 <- mean(human_train$human)
pred1_RMSE <- RMSE_f(pred1,human_test$human)
human_results <- data.frame(prednum = 1, predtype = "Overall Mean",
                            predmodel = "Overall Mean", RMSE = pred1_RMSE,
                            Pct_Improve = 0)
knitr::kable(human_results,caption="HUman Caused Model Results",
             col.names=c("Pred#","Type","Algorithm","RMSE","Overall %Improved"))
```

Table 2: HUman Caused Model Results

Pred#	Type	Algorithm	RMSE	Overall %Improved
1	Overall Mean	Overall Mean	4.86	0

3.3.3 Population Predictors

The **county-master** tibble contains the following potential population predictors:

- 2019 population
- Natural population change, 2010-2019
- Population change due to domestic immigration, 2010-2019
- Population change due to international immigration, 2010-2019
- Whether or not the county is a coastal county

The following code calculates the correlation matrix between these predictors and the "Human Caused" support percentage.

```r
#Look at correlation among population-related predictors
human_pop <- human_train %>%
  select(human,pop2019,popchgnat,popchgdimm,popchgiimm,coast)
pop_cm <- cor(human_pop)
pop_cm
```

```
##             human pop2019 popchgnat popchgdimm popchgiimm  coast
## human       1.000   0.366     0.317     -0.143      0.344  0.304
## pop2019     0.366   1.000     0.940     -0.451      0.832  0.218
## popchgnat   0.317   0.940     1.000     -0.455      0.819  0.167
## popchgdimm -0.143  -0.451    -0.455      1.000     -0.534 -0.112
## popchgiimm  0.344   0.832     0.819     -0.534      1.000  0.234
## coast       0.304   0.218     0.167     -0.112      0.234  1.000
```

Based on the guidelines established for analysis, the domestic immigration predictor was removed from consideration due to its low correlation with the "Human Caused" support percentage.

Next, the findCorrelation function was run to identify pairs of predictors that are highly correlated with each other, and identify individual predictors that could be removed on that basis.

```
toremove <- findCorrelation(pop_cm,cutoff=0.8)
toremove
```

```
## [1] 2 5
```

The pairwise correlation check flagged the second and fifth columns in the matrix (2019 population and change due to international immigration) for removal. This left three population predictors:

- Natural population change, 2010-2019
- Population change due to international immigration, 2010-2019
- Whether or not the county is a coastal county

The three candidate algorithms – linear regression, K nearest neighbor, and random forest – were then used to build models using these three population predictors in the training set. The resulting models were then used to predict support percentages in the test set.

```
#Prediction 2: Population predictors only
#Linear regression
linreg <- train(human ~ popchgnat + popchgiimm + coast, data=human_train, method = "lm")
pred2lin <- predict(linreg,human_test)
pred2lin_RMSE <- RMSE_f(pred2lin,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 2, predtype = "Population",
                   predmodel = "Linear", RMSE = pred2lin_RMSE,
                   Pct_Improve = (pred1_RMSE - pred2lin_RMSE)/pred1_RMSE*100))

#Knn
set.seed(543, sample.kind="Rounding")
knn <- train(human ~ popchgnat + popchgiimm + coast, data=human_train, method = "knn",
             tuneGrid = data.frame(k = seq(51,81,2)))
pred2knn <- predict(knn,human_test)
pred2knn_RMSE <- RMSE_f(pred2knn,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 2, predtype = "Population",
                   predmodel = "Knn", RMSE = pred2knn_RMSE,
                   Pct_Improve = (pred1_RMSE - pred2knn_RMSE)/pred1_RMSE*100))

#random forest
set.seed(654, sample.kind="Rounding")
ranfor <- train(human ~ popchgnat + popchgiimm + coast, data=human_train, method = "rf",
                importance = TRUE)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
pred2rf <- predict(ranfor,human_test)
pred2rf_RMSE <- RMSE_f(pred2rf,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                  prednum = 2, predtype = "Population",
                  predmodel = "RF", RMSE = pred2rf_RMSE,
                  Pct_Improve = (pred1_RMSE - pred2rf_RMSE)/pred1_RMSE*100))
rfimp_hu2 <- varImp(ranfor)

knitr::kable(human_results,caption="Human Caused Model Results",
              col.names=c("Pred#","Type","Algorithm","RMSE","Overall %Improved"))
```

Table 3: Human Caused Model Results

Pred#	Type	Algorithm	RMSE	Overall %Improved
1	Overall Mean	Overall Mean	4.86	0.00
2	Population	Linear	4.58	5.84
2	Population	Knn	4.00	17.79
2	Population	RF	4.31	11.40

As the results table shows, the K nearest neighbors algorithm was the best performer with a modest 17.8% improvement over the baseline. The linear regession algorithm was the worst performer, less than 6% better than the baseline.

3.3.4 Economic Predictors

The **county-master** tibble contains the following potential economic predictors for each county:

- Whether or not the county produces coal
- Whether or not the county produces oil/natural gas
- 2019 current-dollar GDP
- Current-dollar GDP change, 2010-2019
- 2019 per capita personal income
- Per capita personal income change, 2010-2019

The matrix below shows the degree of correlation between these predictors and the "Human Caused" support percentage.

```
human_econ <- human_train %>%
  select(human,coal,oilgas,gdp2019,gdpchg,inc2019,incchg)
econ_cm <- cor(human_econ)
econ_cm
```

```
##          human    coal  oilgas gdp2019  gdpchg inc2019  incchg
## human    1.000 -0.1535 -0.1203  0.3702  0.3469  0.3846  0.2822
## coal    -0.153  1.0000  0.2292 -0.0335 -0.0339 -0.0603 -0.0682
## oilgas  -0.120  0.2292  1.0000 -0.0364 -0.0311  0.0101  0.0539
## gdp2019  0.370 -0.0335 -0.0364  1.0000  0.9832  0.3666  0.2889
## gdpchg   0.347 -0.0339 -0.0311  0.9832  1.0000  0.3772  0.3192
## inc2019  0.385 -0.0603  0.0101  0.3666  0.3772  1.0000  0.8698
## incchg   0.282 -0.0682  0.0539  0.2889  0.3192  0.8698  1.0000
```

Based on the guidelines established for analysis, the coal and oil/natural gas predictors were removed from consideration due to low correlation with the "Human Caused" support percentage.

Next, the findCorrelation function was run to identify pairs of predictors that are highly correlated with each other, and identify individual predictors that could be removed on that basis.

```
toremove <- findCorrelation(econ_cm,cutoff=0.8)
toremove
```

```
## [1] 5 6
```

The pairwise correlation check flags columns 5 and 6 in the matrix (GDP change and 2019 income). This left two economic predictors:

- 2019 current-dollar GDP
- Per capita personal income change, 2010-2019

The three candidate algorithms – linear regression, K nearest neighbor, and random forest – were then used to build models using these two economic predictors in the training set. The resulting models were then used to predict support percentages in the test set.

```
#Prediction 3: Economic predictors only
#Linear regression
linreg <- train(human ~ gdp2019 + incchg, data=human_train, method = "lm")
pred3lin <- predict(linreg,human_test)
pred3lin_RMSE <- RMSE_f(pred3lin,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 3, predtype = "Economic",
                   predmodel = "Linear", RMSE = pred3lin_RMSE,
                   Pct_Improve = (pred1_RMSE - pred3lin_RMSE)/pred1_RMSE*100))

#Knn
set.seed(543, sample.kind="Rounding")
knn <- train(human ~ gdp2019 + incchg, data=human_train, method = "knn",
             tuneGrid = data.frame(k = seq(81,111,2)))
pred3knn <- predict(knn,human_test)
pred3knn_RMSE <- RMSE_f(pred3knn,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 3, predtype = "Economic",
                   predmodel = "Knn", RMSE = pred3knn_RMSE,
                   Pct_Improve = (pred1_RMSE - pred3knn_RMSE)/pred1_RMSE*100))

#random forest
set.seed(654, sample.kind="Rounding")
ranfor <- train(human ~ gdp2019 + incchg, data=human_train, method = "rf",
                importance = TRUE)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
pred3rf <- predict(ranfor,human_test)
pred3rf_RMSE <- RMSE_f(pred3rf,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                    prednum = 3, predtype = "Economic",
                    predmodel = "RF", RMSE = pred3rf_RMSE,
                    Pct_Improve = (pred1_RMSE - pred3rf_RMSE)/pred1_RMSE*100))
rfimp_hu3 <- varImp(ranfor)

knitr::kable(human_results,caption="Human Caused Model Results",
               col.names=c("Pred#","Type","Algorithm","RMSE","Overall %Improved"))
```

Table 4: Human Caused Model Results

Pred#	Type	Algorithm	RMSE	Overall %Improved
1	Overall Mean	Overall Mean	4.86	0.00
2	Population	Linear	4.58	5.84
2	Population	Knn	4.00	17.79
2	Population	RF	4.31	11.40
3	Economic	Linear	4.52	7.00
3	Economic	Knn	4.20	13.68
3	Economic	RF	4.46	8.22

The models based on economic predictors were less successful than the population-based ones. The K nearest neighbors algorithm was again the best performer, but with only a 13.7% improvement over baseline. Neither the linear regression model nor the random forest model were particularly effective (in the case of random forest, this is likely due to a shortage of predictors to partition in trees).

3.3.5 Climate predictors

The **county-master** tibble contains 80 potential climate predictors:

- Four types of climate predictors (mean temperature, mean maximum temperature, mean minimum temperature, precipitation), multiplied by…
- Four seasons, multiplied by…
- Five time windows (2019, change since 2009, change since 1999, change since 1989, change since 1980).

As with other classes of predictors, correlation with the support percentage for “Human Caused” will be used to reduce the number of climate predictors. To facilitate analysis and display, climate correlation was examined in subgroups by type of predictor and by season within type. In addition, only one predictor (the best performer) was selected per type/season combination. For example, if there were three mean temperature/summer predictors that met the correlation cutoff, only the one with the highest correlation would be selected as a predictor.

The first type examined were mean temperature predictors:

```
###############################
# Climate predictors
###############################

#- Mean temperatures
human_temp_cm <- human_train %>%
  filter(is.na(sp2019temp) == FALSE) %>%
  select(human,sp2019temp,su2019temp,fa2019temp,wi2019temp,
         sptempd2009,sutempd2009,fatempd2009,witempd2009,
         sptempd1999,sutempd1999,fatempd1999,witempd1999,
         sptempd1989,sutempd1989,fatempd1989,witempd1989,
         sptempd1980,sutempd1980,fatempd1980,witempd1980)

#Look at correlation for mean temperature predictors by season
#Spring
human_tempsp_cm <- human_temp_cm %>%
  select(human,sp2019temp,sptempd2009,sptempd1999,sptempd1989,sptempd1980)
cmat_tempsp <- cor(human_tempsp_cm)

#Summer
human_tempsu_cm <- human_temp_cm %>%
  select(human,su2019temp,sutempd2009,sutempd1999,sutempd1989,sutempd1980)
cmat_tempsu <- cor(human_tempsu_cm)

#Fall
human_tempfa_cm <- human_temp_cm %>%
  select(human,fa2019temp,fatempd2009,fatempd1999,fatempd1989,fatempd1980)
cmat_tempfa <- cor(human_tempfa_cm)

#Winter
human_tempwi_cm <- human_temp_cm %>%
  select(human,wi2019temp,witempd2009,witempd1999,witempd1989,witempd1980)
cmat_tempwi <- cor(human_tempwi_cm)

cmat_tempsp
```

```
##               human sp2019temp sptempd2009 sptempd1999 sptempd1989 sptempd1980
## human        1.0000     -0.162      0.0264      0.0438      0.0467     -0.0619
## sp2019temp  -0.1617      1.000      0.3794      0.4699      0.3232      0.6287
## sptempd2009  0.0264      0.379      1.0000      0.8174      0.7510      0.6041
## sptempd1999  0.0438      0.470      0.8174      1.0000      0.6621      0.8579
## sptempd1989  0.0467      0.323      0.7510      0.6621      1.0000      0.6067
## sptempd1980 -0.0619      0.629      0.6041      0.8579      0.6067      1.0000
```

```
cmat_tempsu
```

```
##               human su2019temp sutempd2009 sutempd1999 sutempd1989 sutempd1980
## human        1.0000    -0.2050       0.139      0.3161      0.0308      0.3874
## su2019temp  -0.2050     1.0000      -0.455     -0.0603      0.5078     -0.3891
## sutempd2009  0.1386    -0.4550       1.000     -0.1032     -0.3909      0.1504
## sutempd1999  0.3161    -0.0603      -0.103      1.0000      0.3573      0.5864
## sutempd1989  0.0308     0.5078      -0.391      0.3573      1.0000     -0.0118
## sutempd1980  0.3874    -0.3891       0.150      0.5864     -0.0118      1.0000
```

```
cmat_tempfa
```

```
##                human fa2019temp fatempd2009 fatempd1999 fatempd1989 fatempd1980
## human        1.00000     -0.142      -0.118     0.00144     -0.0413      0.0378
## fa2019temp  -0.14198      1.000       0.678     0.64523      0.5418      0.5889
## fatempd2009 -0.11803      0.678       1.000     0.80287      0.7444      0.6724
## fatempd1999  0.00144      0.645       0.803     1.00000      0.8651      0.8331
## fatempd1989 -0.04130      0.542       0.744     0.86507      1.0000      0.8472
## fatempd1980  0.03781      0.589       0.672     0.83305      0.8472      1.0000
```

```
cmat_tempwi
```

```
##               human wi2019temp witempd2009 witempd1999 witempd1989 witempd1980
## human        1.0000     -0.109      -0.380      0.0527       0.102      -0.055
## wi2019temp  -0.1088      1.000       0.435      0.4749       0.268       0.527
## witempd2009 -0.3803      0.435       1.000      0.2434       0.158       0.478
## witempd1999  0.0527      0.475       0.243      1.0000       0.717       0.875
## witempd1989  0.1018      0.268       0.158      0.7175       1.000       0.720
## witempd1980 -0.0550      0.527       0.478      0.8755       0.720       1.000
```

```
#Using a 0.2 abs val correlation threshold:
#Spring: had no significance
#Summer: 1980 0.389, 1999 0.310, 2019 -0.229
#Fall: had no significance
#Winter: 2009 -0.392
#So do summer since 1980, winter since 2009
```

Based on the guidelines established for analysis, none of the spring or fall mean temperatures were correlated enough to consider further. Three summer values and one winter value did meet the correlation cutoff. As previously discussed, only one predictor per season/type combination was selected. This left an initial list of two mean temperature predictors:

- Summer mean temperature change, 1980-2019
- Winter mean temperature change, 2009-2019

The next type examined were mean maximum temperatures:

```
#-Max temperatures
human_tmax_cm <- human_train %>%
  filter(is.na(sp2019tmax) == FALSE) %>%
  select(human,sp2019tmax,su2019tmax,fa2019tmax,wi2019tmax,
         sptmaxd2009,sutmaxd2009,fatmaxd2009,witmaxd2009,
         sptmaxd1999,sutmaxd1999,fatmaxd1999,witmaxd1999,
         sptmaxd1989,sutmaxd1989,fatmaxd1989,witmaxd1989,
         sptmaxd1980,sutmaxd1980,fatmaxd1980,witmaxd1980)

#Look at correlation for mean max temperature predictors by season
#Spring
human_tmaxsp_cm <- human_tmax_cm %>%
  select(human,sp2019tmax,sptmaxd2009,sptmaxd1999,sptmaxd1989,sptmaxd1980)
cmat_tmaxsp <- cor(human_tmaxsp_cm)

#Summer
human_tmaxsu_cm <- human_tmax_cm %>%
  select(human,su2019tmax,sutmaxd2009,sutmaxd1999,sutmaxd1989,sutmaxd1980)
cmat_tmaxsu <- cor(human_tmaxsu_cm)
```

```
#Fall
human_tmaxfa_cm <- human_tmax_cm %>%
  select(human,fa2019tmax,fatmaxd2009,fatmaxd1999,fatmaxd1989,fatmaxd1980)
cmat_tmaxfa <- cor(human_tmaxfa_cm)

#Winter
human_tmaxwi_cm <- human_tmax_cm %>%
  select(human,wi2019tmax,witmaxd2009,witmaxd1999,witmaxd1989,witmaxd1980)
cmat_tmaxwi <- cor(human_tmaxwi_cm)

cmat_tmaxsp
```

```
##                 human sp2019tmax sptmaxd2009 sptmaxd1999 sptmaxd1989 sptmaxd1980
## human          1.0000     -0.184     -0.0403      0.0219      0.0399     -0.0461
## sp2019tmax  -0.1841      1.000      0.4202      0.4539      0.2657      0.5745
## sptmaxd2009 -0.0403      0.420      1.0000      0.7872      0.7696      0.6153
## sptmaxd1999  0.0219      0.454      0.7872      1.0000      0.5971      0.8229
## sptmaxd1989  0.0399      0.266      0.7696      0.5971      1.0000      0.5692
## sptmaxd1980 -0.0461      0.575      0.6153      0.8229      0.5692      1.0000
```

```
cmat_tmaxsu
```

```
##                 human su2019tmax sutmaxd2009 sutmaxd1999 sutmaxd1989 sutmaxd1980
## human          1.0000    -0.2073     0.18350     0.30681      0.0117       0.386
## su2019tmax  -0.2073     1.0000    -0.46610     0.03384      0.4618      -0.352
## sutmaxd2009  0.1835    -0.4661     1.00000    -0.00403     -0.2229       0.294
## sutmaxd1999  0.3068     0.0338    -0.00403     1.00000      0.2696       0.536
## sutmaxd1989  0.0117     0.4618    -0.22291     0.26960      1.0000       0.121
## sutmaxd1980  0.3860    -0.3521     0.29449     0.53647      0.1213       1.000
```

```
cmat_tmaxfa
```

```
##                 human fa2019tmax fatmaxd2009 fatmaxd1999 fatmaxd1989 fatmaxd1980
## human          1.0000     -0.183      -0.190       0.071     -0.0756      0.0244
## fa2019tmax  -0.1827      1.000       0.723       0.599      0.5909      0.5375
## fatmaxd2009 -0.1897      0.723       1.000       0.741      0.8246      0.6292
## fatmaxd1999  0.0710      0.599       0.741       1.000      0.8563      0.7996
## fatmaxd1989 -0.0756      0.591       0.825       0.856      1.0000      0.8528
## fatmaxd1980  0.0244      0.538       0.629       0.800      0.8528      1.0000
```

```
cmat_tmaxwi
```

```
##                  human wi2019tmax witmaxd2009 witmaxd1999 witmaxd1989 witmaxd1980
## human          1.00000     -0.125     -0.3780      0.1377       0.119    -0.00304
## wi2019tmax  -0.12535      1.000      0.4180      0.3174       0.306     0.45797
## witmaxd2009 -0.37803      0.418      1.0000      0.0326       0.130     0.37475
## witmaxd1999  0.13769      0.317      0.0326      1.0000       0.687     0.78527
## witmaxd1989  0.11896      0.306      0.1303      0.6873       1.000     0.66860
## witmaxd1980 -0.00304      0.458      0.3748      0.7853       0.669     1.00000
```

```r
#Using a 0.2 abs val correlation threshold:
#Spring: 2019 -.207
#Summer: 1980 0.387
#Fall: 2019 -0.205
#Winter: 2009 -0.389
```

At least one mean maximum temperature value in each of the seasons met the cutoff. After selecting the season value with the highest correlation, the initial list of mean maximum temperature predictors is:

- Spring 2019 mean maximum temperature
- Summer mean maximum temperature change, 1980-2019
- Fall 2019 mean maximum temperature
- Winter mean maximum temperature change, 2009-2019

Note that the summer and winter predictors selected for mean maximum temperature are the same as those selected for mean temperature.

The next type examined were mean minimum temperatures:

```r
#-Min temperatures
human_tmin_cm <- human_train %>%
  filter(is.na(sp2019tmin) == FALSE) %>%
  select(human,sp2019tmin,su2019tmin,fa2019tmin,wi2019tmin,
         sptmind2009,sutmind2009,fatmind2009,witmind2009,
         sptmind1999,sutmind1999,fatmind1999,witmind1999,
         sptmind1989,sutmind1989,fatmind1989,witmind1989,
         sptmind1980,sutmind1980,fatmind1980,witmind1980)

#Look at correlation for mean max temperature predictors by season
#Spring
human_tminsp_cm <- human_tmin_cm %>%
  select(human,sp2019tmin,sptmind2009,sptmind1999,sptmind1989,sptmind1980)
cmat_tminsp <- cor(human_tminsp_cm)

#Summer
human_tminsu_cm <- human_tmin_cm %>%
  select(human,su2019tmin,sutmind2009,sutmind1999,sutmind1989,sutmind1980)
cmat_tminsu <- cor(human_tminsu_cm)

#Fall
human_tminfa_cm <- human_tmin_cm %>%
  select(human,fa2019tmin,fatmind2009,fatmind1999,fatmind1989,fatmind1980)
cmat_tminfa <- cor(human_tminfa_cm)

#Winter
human_tminwi_cm <- human_tmin_cm %>%
  select(human,wi2019tmin,witmind2009,witmind1999,witmind1989,witmind1980)
cmat_tminwi <- cor(human_tminwi_cm)

#Using a 0.2 abs val correlation threshold:
#Spring: No significance
#Summer: 1980 0.323
#Fall: No significance
#Winter: 2009 -0.363
```

As the correlation matrices show, no spring or fall values were significant enough to include. The two selected predictors for mean minimum temperature were:

- Summer mean minimum temperature change, 1980-2019
- Winter mean minimum temperature change, 2009-2019

Once again, the summer and winter predictors selected for mean minimum temperature were the same as those selected for the other two temperature types.

The last type examined was precipitation:

```
#-Precipitation
human_prcp_cm <- human_train %>%
  filter(is.na(sp2019prcp) == FALSE) %>%
  select(human,sp2019prcp,su2019prcp,fa2019prcp,wi2019prcp,
         spprcpd2009,suprcpd2009,faprcpd2009,wiprcpd2009,
         spprcpd1999,suprcpd1999,faprcpd1999,wiprcpd1999,
         spprcpd1989,suprcpd1989,faprcpd1989,wiprcpd1989,
         spprcpd1980,suprcpd1980,faprcpd1980,wiprcpd1980)

#Look at correlation for precipitation predictors by season
#Spring
human_prcpsp_cm <- human_prcp_cm %>%
  select(human,sp2019prcp,spprcpd2009,spprcpd1999,spprcpd1989,spprcpd1980)
cmat_prcpsp <- cor(human_prcpsp_cm)

#Summer
human_prcpsu_cm <- human_prcp_cm %>%
  select(human,su2019prcp,suprcpd2009,suprcpd1999,suprcpd1989,suprcpd1980)
cmat_prcpsu <- cor(human_prcpsu_cm)

#Fall
human_prcpfa_cm <- human_prcp_cm %>%
  select(human,fa2019prcp,faprcpd2009,faprcpd1999,faprcpd1989,faprcpd1980)
cmat_prcpfa <- cor(human_prcpfa_cm)

#Winter
human_prcpwi_cm <- human_prcp_cm %>%
  select(human,wi2019prcp,wiprcpd2009,wiprcpd1999,wiprcpd1989,wiprcpd1980)
cmat_prcpwi <- cor(human_prcpwi_cm)

#Using a 0.2 abs val correlation threshold:
#Spring: No significance
#Summer: No significance
#Fall: 1999 -0.238
#Winter: 1980 -0.266
```

As the correlation matrices show, no spring or summer values were significant enough to include. The two selected predictors for precipitation were:

- Fall mean precipitation change, 1999-2019
- Winter mean precipitation change, 1980-2019

With an initial pass of all types completed, the reduced list of climate predictors was as follows:

- Summer mean temperature change since 1980
- Winter mean temperature change since 2009
- Spring mean maximum temperature in 2019
- Summer mean maximum temperature change since 1980
- Fall mean maximum temperature in 2019
- Winter mean maximum temperature change since 2009
- Summer mean minimum temperature change since 1980
- Winter mean minimum temperature change since 2009
- Fall mean precipitation change since 1999
- Winter mean precipitation change since 1980

This list was then checked for pairwise correlation:

```
human_clim_cm <- human_train %>%
  filter(is.na(su2019temp) == FALSE) %>%
  select(human,sutempd1980,witempd2009,sp2019tmax,sutmaxd1980,fa2019tmax,
         witmaxd2009,sutmind1980,witmind2009,faprcpd1999,wiprcpd1980)
clim_cm <- cor(human_clim_cm)
clim_cm
```

```
##              human sutempd1980 witempd2009 sp2019tmax sutmaxd1980 fa2019tmax
## human        1.000       0.387      -0.380    -0.1841       0.386     -0.183
## sutempd1980  0.387       1.000      -0.532    -0.2008       0.969     -0.243
## witempd2009 -0.380      -0.532       1.000     0.5142      -0.593      0.526
## sp2019tmax  -0.184      -0.201       0.514     1.0000      -0.246      0.981
## sutmaxd1980  0.386       0.969      -0.593    -0.2456       1.000     -0.282
## fa2019tmax  -0.183      -0.243       0.526     0.9815      -0.282      1.000
## witmaxd2009 -0.378      -0.578       0.965     0.4939      -0.617      0.522
## sutmind1980  0.323       0.886      -0.326    -0.0816       0.745     -0.128
## witmind2009 -0.353      -0.440       0.958     0.4963      -0.521      0.490
## faprcpd1999 -0.245      -0.384       0.128    -0.1867      -0.345     -0.156
## wiprcpd1980 -0.268      -0.175       0.447     0.4952      -0.246      0.464
##             witmaxd2009 sutmind1980 witmind2009 faprcpd1999 wiprcpd1980
## human            -0.378      0.3228      -0.353     -0.2455     -0.2680
## sutempd1980      -0.578      0.8859      -0.440     -0.3842     -0.1746
## witempd2009       0.965     -0.3260       0.958      0.1280      0.4471
## sp2019tmax        0.494     -0.0816       0.496     -0.1867      0.4952
## sutmaxd1980      -0.617      0.7445      -0.521     -0.3449     -0.2456
## fa2019tmax        0.522     -0.1280       0.490     -0.1559      0.4637
## witmaxd2009       1.000     -0.4069       0.850      0.1321      0.3509
## sutmind1980      -0.407      1.0000      -0.213     -0.3923     -0.0118
## witmind2009       0.850     -0.2126       1.000      0.1134      0.5166
## faprcpd1999       0.132     -0.3923       0.113      1.0000     -0.0366
## wiprcpd1980       0.351     -0.0118       0.517     -0.0366      1.0000
```

```
toremove <- findCorrelation(clim_cm,cutoff=0.8)
toremove
```

```
## [1] 3 7 5 2 6
```

Pairwise correlation flags columns 3,7,5,2, and 6 for removal. With these removed, five climate predictors remain:

- Spring mean maximum temperature in 2019
- Summer mean minimum temperature change since 1980
- Winter mean minimum temperature change since 2009
- Fall mean precipitation change since 1999
- Winter mean precipitation change since 1980

The three candidate algorithms – linear regression, K nearest neighbor, and random forest – were then used to build models using these five climate predictors in the training set. The resulting models were then used to predict support percentages in the test set.

```r
#Prediction 5: All climate predictors

#Linear regression
linreg <- train(human ~ sp2019tmax + sutmind1980 + witmind2009 +
                faprcpd1999 + wiprcpd1980, data=human_train, method = "lm")
pred5lin <- predict(linreg,human_test)
pred5lin_RMSE <- RMSE_f(pred5lin,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 5, predtype = "Climate",
                   predmodel = "Linear", RMSE = pred5lin_RMSE,
                   Pct_Improve = (pred1_RMSE - pred5lin_RMSE)/pred1_RMSE*100))

#Knn
set.seed(543, sample.kind="Rounding")
knn <- train(human ~ sp2019tmax + sutmind1980 + witmind2009 +
             faprcpd1999 + wiprcpd1980, data=human_train, method = "knn",
             tuneGrid = data.frame(k = seq(3,53,2)))
pred5knn <- predict(knn,human_test)
pred5knn_RMSE <- RMSE_f(pred5knn,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 5, predtype = "Climate",
                   predmodel = "Knn", RMSE = pred5knn_RMSE,
                   Pct_Improve = (pred1_RMSE - pred5knn_RMSE)/pred1_RMSE*100))


#random forest
set.seed(654, sample.kind="Rounding")
ranfor <- train(human ~ sp2019tmax + sutmind1980 + witmind2009 +
                faprcpd1999 + wiprcpd1980, data=human_train, method = "rf",
                importance = TRUE)
pred5rf <- predict(ranfor,human_test)
pred5rf_RMSE <- RMSE_f(pred5rf,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                   prednum = 5, predtype = "Climate",
                   predmodel = "RF", RMSE = pred5rf_RMSE,
                   Pct_Improve = (pred1_RMSE - pred5rf_RMSE)/pred1_RMSE*100))
rfimp_hu5 <- varImp(ranfor)

knitr::kable(human_results,caption="Human Caused Model Results",
               col.names=c("Pred#","Type","Algorithm","RMSE","Overall %Improved"))
```

Table 5: Human Caused Model Results

Pred#	Type	Algorithm	RMSE	Overall %Improved
1	Overall Mean	Overall Mean	4.86	0.00
2	Population	Linear	4.58	5.84
2	Population	Knn	4.00	17.79
2	Population	RF	4.31	11.40
3	Economic	Linear	4.52	7.00
3	Economic	Knn	4.20	13.68
3	Economic	RF	4.46	8.22
5	Climate	Linear	4.35	10.60
5	Climate	Knn	3.87	20.48
5	Climate	RF	3.79	22.11

The climate predictors alone were significantly more accurate than the population or economic predictors alone. Even the habitual worst performer, the linear regression model, broke the 10% improvement threshold. The random forest model edge out the K nearest neighbors model with a 22.1% improvement.

3.3.6 All Predictors

With analysis of the individual classes of predictors complete, a final model was now built using the three population predictors, the two economic predictors, and the five climate predictors. The full list is:

- Natural population change, 2010-2019
- Population change due to international immigration, 2010-2019
- Whether or not the county is a coastal county
- 2019 current-dollar GDP
- Per capita personal income change, 2010-2019
- Spring mean maximum temperature in 2019
- Summer mean minimum temperature change since 1980
- Winter mean minimum temperature change since 2009
- Fall mean precipitation change since 1999
- Winter mean precipitation change since 1980

The three candidate algorithms – linear regression, K nearest neighbor, and random forest – were then used to build models using all ten predictors in the training set. The resulting models were then used to predict support percentages in the test set.

```r
##############################
# All predictors
##############################

#Prediction 6: All predictors

#Linear regression
linreg <- train(human ~ popchgnat + popchgiimm + coast + gdp2019 + incchg +
                sp2019tmax + sutmind1980 + witmind2009 +
                faprcpd1999 + wiprcpd1980, data=human_train, method = "lm")
pred6lin <- predict(linreg,human_test)
pred6lin_RMSE <- RMSE_f(pred6lin,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                    prednum = 6, predtype = "All",
                    predmodel = "Linear", RMSE = pred6lin_RMSE,
                    Pct_Improve = (pred1_RMSE - pred6lin_RMSE)/pred1_RMSE*100))

#Knn
set.seed(543, sample.kind="Rounding")
knn <- train(human ~ popchgnat + popchgiimm + coast + gdp2019 + incchg +
             sp2019tmax + sutmind1980 + witmind2009 +
             faprcpd1999 + wiprcpd1980, data=human_train, method = "knn",
             tuneGrid = data.frame(k = seq(91,121,2)))
pred6knn <- predict(knn,human_test)
pred6knn_RMSE <- RMSE_f(pred6knn,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                    prednum = 6, predtype = "All",
                    predmodel = "Knn", RMSE = pred6knn_RMSE,
                    Pct_Improve = (pred1_RMSE - pred6knn_RMSE)/pred1_RMSE*100))

#random forest
set.seed(654, sample.kind="Rounding")
ranforh <- train(human ~ popchgnat + popchgiimm + coast + gdp2019 + incchg +
                sp2019tmax + sutmind1980 + witmind2009 +
                faprcpd1999 + wiprcpd1980, data=human_train, method = "rf",
                importance = TRUE)
pred6rf <- predict(ranforh,human_test)
pred6rf_RMSE <- RMSE_f(pred6rf,human_test$human)
human_results <- bind_rows(human_results, data.frame(
                    prednum = 6, predtype = "All",
                    predmodel = "RF", RMSE = pred6rf_RMSE,
                    Pct_Improve = (pred1_RMSE - pred6rf_RMSE)/pred1_RMSE*100))
rfimp_hu6 <- varImp(ranforh)

knitr::kable(human_results,caption="Human Caused Model Results",
               col.names=c("Pred#","Type","Algorithm","RMSE","Overall %Improved"))
```

Table 6: Human Caused Model Results

Pred#	Type	Algorithm	RMSE	Overall %Improved
1	Overall Mean	Overall Mean	4.86	0.00
2	Population	Linear	4.58	5.84
2	Population	Knn	4.00	17.79
2	Population	RF	4.31	11.40
3	Economic	Linear	4.52	7.00
3	Economic	Knn	4.20	13.68
3	Economic	RF	4.46	8.22
5	Climate	Linear	4.35	10.60
5	Climate	Knn	3.87	20.48
5	Climate	RF	3.79	22.11
6	All	Linear	4.12	15.26
6	All	Knn	4.20	13.68
6	All	RF	3.20	34.19

The use of all predictors led to some interesting results. The random forest model turned in its strongest performance yet, dropping RMSE to 3.2, a 34.2% improvement over the baseline. The linear regression model actual outperformed the k nearest neighbor model.

Based on these results, the random forest model with all ten predictors was selected for the final validation run. While a downside of random forest models is their lack of interpretation ability, some insights can be gleaned by looking at variable importance in the model.

```r
print(rfimp_hu6)
```

```
## rf variable importance
##
##              Overall
## popchgiimm   100.00
## faprcpd1999   83.90
## witmind2009   60.12
## sutmind1980   54.32
## sp2019tmax    43.18
## gdp2019       42.25
## wiprcpd1980   42.02
## popchgnat     29.33
## incchg         7.87
## coast          0.00
```

Interestingly, international immigration was the most important variable, suggesting a significant difference in support percentage between counties with lots of international immigration and counties with little. The next four most important variables were climate-related, which seems to coincide with the relatively stronger predictive value of climate predictors vs. population or economic ones.

The significant improvement in random forest performance with more predictors, compared to the decline in performance of the k nearest neighbor model, likely speaks to the strength of random forest in larger predictor sets.

4. Results

Based on the results of the various trial models, the random forest model with all predictors was selected to run against the validation set.

```r
#########################################
# Final validation run
# Use all predictors, random forest model
#########################################

#Human-caused global warming
final_human <- predict(ranforh,human_val)
final_human_RMSE <- RMSE_f(final_human,human_val$human)
print("Validation RMSE (global warming is human-caused):")
```

```
## [1] "Validation RMSE (global warming is human-caused):"
```

```r
final_human_RMSE
```

```
## [1] 3.32
```

```r
print("Percent Improvement over Baseline:")
```

```
## [1] "Percent Improvement over Baseline:"
```

```r
(pred1_RMSE - final_human_RMSE) / pred1_RMSE * 100
```

```
## [1] 31.8
```

The resulting RMSE of 3.316 is similar to this model's performance against the test data set, and represents a 31.8% improvement over the baseline.

To understand the performance further, it is helpful to examine the magnitude of RMSEs in the validation set run, rounded down to the nearest unit, and to compare the RMSE to the standard deviation of support percentage in the validation set.

```r
nrow(human_val)
```

```
## [1] 316
```

```r
valsd <- sd(human_val$human)

valresults <- data.frame(cfips = human_val$cfips,
                         sname = human_val$sname,
                         cname = human_val$cname,
                         human = human_val$human,
                         final_human = final_human)
valresults <- valresults %>%
  mutate(rfdiff = final_human - human,
         rfabs = abs(rfdiff),
         rffloor = floor(rfabs))
valresults %>% group_by (rffloor) %>% summarize(count=n())
```

```
## # A tibble: 12 x 2
##    rffloor count
##      <dbl> <int>
##  1       0    89
##  2       1    66
##  3       2    59
##  4       3    41
##  5       4    20
##  6       5    17
##  7       6    10
##  8       7     7
##  9       8     4
## 10       9     1
## 11      10     1
## 12      14     1
```

```r
valbigerr <- valresults %>% filter(rfabs > valsd)
```

As these results show, the "Human-caused" support percentages in 280 of the 316 validation counties (88.6%) were predicted with an RMSE of less than 5.38, the standard deviation of the support percentage in the validation set.

```r
vr1 <- valresults %>% filter(rfdiff >= valsd)
nrow(vr1)
```

```
## [1] 16
```

```r
vr2 <- valresults %>% filter(rfdiff <= -valsd)
nrow(vr2)
```

```
## [1] 20
```

Of the 36 counties with high RMSEs, there were 16 over-predictions and 20 under- predictions.

The top 10 over- and under-estimated counties are as follows:

```r
top10over <- vr1 %>% select(-rfabs,-rffloor) %>%
               top_n(10,wt=rfdiff) %>% arrange(desc(rfdiff))
top10under <- vr2 %>% select(-rfabs,-rffloor) %>%
               top_n(10,wt=abs(rfdiff)) %>% arrange(rfdiff)
knitr::kable(top10over,caption="Worst Overestimated Predictions")
```

Table 7: Worst Overestimated Predictions

cfips	sname	cname	human	final_human	rfdiff
42071	PENNSYLVANIA	LANCASTER COUNTY	53.7	61.8	8.11
16041	IDAHO	FRANKLIN COUNTY	42.0	50.0	7.96
48235	TEXAS	IRION COUNTY	43.9	51.5	7.58
40025	OKLAHOMA	CIMARRON COUNTY	41.8	49.1	7.32
31153	NEBRASKA	SARPY COUNTY	48.1	55.2	7.09
16027	IDAHO	CANYON COUNTY	47.2	54.1	6.94
12071	FLORIDA	LEE COUNTY	50.7	57.3	6.64
48433	TEXAS	STONEWALL COUNTY	42.7	49.3	6.62
30025	MONTANA	FALLON COUNTY	43.1	49.6	6.53
13231	GEORGIA	PIKE COUNTY	41.4	47.9	6.51

```r
knitr::kable(top10under,caption="Worst Underestimated Predictions")
```

Table 8: Worst Underestimated Predictions

cfips	sname	cname	human	final_human	rfdiff
48377	TEXAS	PRESIDIO COUNTY	69.1	54.3	-14.77
26161	MICHIGAN	WASHTENAW COUNTY	68.7	58.2	-10.50
55078	WISCONSIN	MENOMINEE COUNTY	62.3	52.6	-9.70
08051	COLORADO	GUNNISON COUNTY	61.5	52.8	-8.68
48465	TEXAS	VAL VERDE COUNTY	62.8	54.3	-8.56
17019	ILLINOIS	CHAMPAIGN COUNTY	64.5	56.4	-8.07
41003	OREGON	BENTON COUNTY	65.1	57.3	-7.84

cfips	sname	cname	human	final_human	rfdiff
51760	VIRGINIA	RICHMOND CITY	65.3	57.8	-7.51
55003	WISCONSIN	ASHLAND COUNTY	58.9	51.9	-7.09
13037	GEORGIA	CALHOUN COUNTY	54.7	47.7	-6.93

5. Conclusions

5.1 Key points

The final model showed a modest level of success in predicting support percentage for the statement "Global warming is caused mostly by human activities". The validation set RMSE of 3.316 is an improvement of 31.8% from the baseline prediction using the overall mean support in the training set. In addition, 88.6% of the validation counties had predictions with RMSEs that were less than the standard deviation of the support percentage in the validation set, which indicates some reasonable degree of accuracy.

Among the three classes of predictors, climate predictors had by the far the largest impact on support percentage. Going back to the question that initiated this project, it appears that changes in climate metrics do have some impact on the support percentage for the "Human Caused" statement.

5.2 Areas for Improvement

One key population predictor not included in this project was population density, due to a difficulty in locating downloadable data sets with this information. If population density could be included, this would be a useful one to evaluate, since it would more starkly delineate any urban/suburban/rural differences in support.

There were six counties (five from Hawaii, one from Virginia) that were not predicted because climate records for them could not be located. If this data was found, the model could cover the full 50 states.

When evaluating potential predictors, this project considered changes in population and economic indicators in absolute terms. It might be interesting to look at these in terms of percentage of change, which might provide better performance for smaller-population counties.

5.3 Future Work

Political preference data by county was out of scope for this project. However, anecdotal data from the media suggests that there is some degree of linkage between climate change opinions and Democrat vs. Republican identification. It would be interesting to compare the predictive power of political preference to climate predictors.

The opinion data from the Yale Program on Climate Communications includes support percentages for 31 climate change statements. Only one was addressed by this project – "Global warming is caused mostly by human activity." Future project could perform similar analysis on some of the other questions, and the similarities and differences in predictor strength might be illuminating. Were this to be done, the R code could make more use of nested functions to allow for easy scalability to multiple questions.

Appendices

Appendix 1: Specific Paths to Data Sources

The NOAA climate data files were found on the following site: ftp://ftp.ncdc.noaa.gov/pub/data/cirs/climdiv/. The specific files I used are as follows: Temperature file: **climdiv-tmpccy-v1.0.0-20201205** Max temperature file: **climdiv-tmaxcy-v1.0.0-20201205** Min temperature file: **climdiv-tmincy-v1.0.0-20201205** Precipitation file: **climdiv-pcpncy-v1.0.0-20201205** I also made use of the **county-readme.txt** file, first to understand the data, then to interpret the state coding.

The U.S. Census Bureau 2019 county population estimates and changes since 2010 were obtained from the following site: https://www.census.gov/data/datasets/time-series/demo/popest/2010s-counties-total.html#par_textimage_70769902. The specific file used was **co-est2019-alldata.csv**.

The list of coastal counties was obtained from the U.S. Census Bureau at the following site: www2.census.gov/library/stories/2018/08/. The specific file used was **coastline-counties-list.xlsx**.

GDP and personal income data were obtained from the Bureau of Economic Analysis, part of the U.S. Department of Commerce, at the following site: https://apps.bea.gov/regional/downloadzip.cfm. The GDP file used (selected from the drop-down) was **CAGDP1: GDP Summary by County and MSA**. The personal income file used (selected from the drop-down) was **CAINC1: Annual Personal Income by County**.

Coal, oil and natural gas data by county was obtained from the Energy Information Administration, part of the U.S. Department of Energy. Coal data was found at the following site: https://www.eia.gov/coal/annual/. I downloaded **Table 2, Coal Production and Number of Mines by State, County, and Mine Type**, in Excel format. Oil/natural gas data was found at the following site: https://www.eia.gov/petroleum/drilling/. I downloaded **Report data (aggregated by region)**.

Because many of these data sources used different identifier codes for counties and states, it was necessary to convert them all to a consistent standard. To do so, I downloaded the following: From https://worldpopulationreview.com/states/state-abbreviations, a list of US state names and abbreviations. From the Census Bureau at https://www.census.gov/geographies/reference-files/2019/demo/popest/2019-fips.html, a list of the 2019 CFIPS county identifiers.

Appendix 2

The text below, copied directly from the NOAA site, describes the collection methods and the file format for the four NOAA county climate-level files used in this project. This is the same information present in the **county-readme.txt** file discussed in Appendix 1, and can also be found in the project's github subdirectory.

"This documentation describes the record format for the county files on /pub/data/cirs/climdiv that have the filenames:

climdiv-pcpncy-vx.y.z-YYYYMMDD climdiv-tmaxcy-vx.y.z-YYYYMMDD climdiv-tmincy-vx.y.z-YYYYMMDD climdiv-tmpccy-vx.y.z-YYYYMMDD

```
                         nClimDiv
                          COUNTY
                 TEMPERATURE-PRECIPITATION

                       OCTOBER 2018
```

The major parameters in this file are sequential climatic county monthly maximum, minimum and average temperature (deg. F. to 10ths) and precipitation (inches to 100ths). Period of record is 1895 through latest month available, updated monthly.

Values from the most recent two calendar years will be updated on a monthly basis. Period of record updates will occur when the underlying data set undergoes a version change.

METHODOLOGY:

County values in nClimDiv were derived from area-weighted averages of grid-point estimates interpolated from station data. A nominal grid resolution of 5 km was used to ensure that all divisions had sufficient spatial sampling (only four small divisions had less than 100 points) and because the impact of elevation on precipitation is minimal below 5 km. Station data were gridded via climatologically aided interpolation to minimize biases from topographic and network variability.

The Global Historical Climatology Network (GHCN) Daily dataset is the source of station data for nClimDiv. GHCN-Daily contains several major observing networks in North America, five of which are used here. The primary network is the National Weather Service (NWS) Cooperative Observing (COOP) program, which consists of stations operated by volunteers as well as by agencies such as the Federal Aviation Administration. To improve coverage in western states and along international borders, nClimDiv also includes the National Interagency Fire Center (NIFC) Remote Automatic Weather Station (RAWS) network, the USDA Snow Telemetry (SNOTEL) network, the Environment Canada (EC) network (south of 52°N), and part of Mexicos Servicio Meteorologico Nacional (SMN) network (north of 24°N). Note that nClimDiv does not incorporate precipitation data from RAWS because that networks tipping-bucket gauges are unheated, leading to suspect cold-weather data.

All GHCN-Daily stations are routinely processed through a suite of logical, serial, and spatial quality assurance reviews to identify erroneous observations. For nClimDiv, all such data were set to missing before computing monthly values, which in turn were subjected to additional serial and spatial checks to eliminate residual outliers. Stations having at least 10 years of valid monthly data since 1950 were used in nClimDiv.

For temperature, bias adjustments were computed to account for historical changes in observation time, station location, temperature instrumentation, and siting conditions. Changes in observation time are only problematic for the COOP network whereas changes in station location and instrumentation occur in almost all surface networks. As in the U.S. Historical Climatology Network version 2.5, the method of Karl et al. (1986) was applied to remove the observation time bias from the COOP network, and the pairwise method of Menne and Williams (2009) was used to address changes in station location and instrumentation in all networks. Because the pairwise method also largely accounts for local, unrepresentative trends that arise from changes in siting conditions, nClimDiv contains no separate adjustment in that regard.

For additional information on how nClimDiv is constructed, please see: http://journals.ametsoc.org/doi/abs/10.1175/JAMC-D-13-0248.1

STATE CODE TABLE: Range of values of 01-48.

01	Alabama	28	New Jersey
02	Arizona	29	New Mexico
03	Arkansas	30	New York
04	California	31	North Carolina
05	Colorado	32	North Dakota
06	Connecticut	33	Ohio
07	Delaware	34	Oklahoma
08	Florida	35	Oregon
09	Georgia	36	Pennsylvania
10	Idaho	37	Rhode Island
11	Illinois	38	South Carolina
12	Indiana	39	South Dakota
13	Iowa	40	Tennessee
14	Kansas	41	Texas
15	Kentucky	42	Utah
16	Louisiana	43	Vermont

17 Maine	44 Virginia
18 Maryland	45 Washington
19 Massachusetts	46 West Virginia
20 Michigan	47 Wisconsin
21 Minnesota	48 Wyoming
22 Mississippi	
23 Missouri	
24 Montana	
25 Nebraska	
26 Nevada	
27 New Hampshire	

FILE FORMAT:

IMPORTANT NOTE:

The format of the county data is slightly different than the other data files. To accomadate the 2 digit state code and the 3 digit county FIPS code, the first field contains 11 columns. The other data files still contain 10 columns.

Element Record Name Position Element Description

STATE-CODE 1-2 STATE-CODE as indicated in State Code Table as described in FILE 1. Range of values is 01-48.

DIVISION-NUMBER 3-5 COUNTY FIPS - Range of values 001-999.

ELEMENT CODE 6-7 01 = Precipitation 02 = Average Temperature 27 = Maximum Temperature 28 = Minimum Temperature

YEAR 8-11 This is the year of record. Range is 1895 to current year processed.

(all data values are right justified):

JAN-VALUE 12-18

```
Monthly Divisional Temperature format (f7.2)
Range of values -50.00 to 140.00 degrees Fahrenheit.
Decimals retain a position in the 7-character
field.  Missing values in the latest year are
indicated by -99.99.

Monthly Divisional Precipitation format (f7.2)
Range of values 00.00 to 99.99.  Decimal point
retains a position in the 7-character field.
Missing values in the latest year are indicated
by -9.99.
```

FEB-VALUE 19-25

MAR-VALUE 26-32

APR-VALUE 33-39

MAY-VALUE 40-46

JUNE-VALUE 47-53

JULY-VALUE 54-60

AUG-VALUE 61-67

SEPT-VALUE 68-74

OCT-VALUE 75-81

NOV-VALUE 82-88

DEC-VALUE 89-95 "