In [1]:
```
if(!require('pacman')) {
  install.packages('pacman')
}
pacman::p_load(tidyverse, skimr, RCurl,
               data.table,bit64,stringr,readxl,tidyr,haven,purrr,
               splitstackshape,bestglm, glmnet, leaps, car, pROC,
              randomForest,  rattle, pROC, usmap, xtable, ggcorrplot,
              fastDummies, caret, janitor, here)
```

```
Loading required package: pacman

Warning message:
“package ‘pacman’ was built under R version 4.0.2”
Installing package into ‘/Users/sahluwalia/Library/R/4.0/library’
(as ‘lib’ is unspecified)

also installing the dependency ‘glmnet’



  There is a binary version available but the source version is later:
       binary source needs_compilation
glmnet  4.1-3  4.1-6              TRUE


The downloaded binary packages are in
	/var/folders/rx/qkbp8l317c31rs2zkvc_p4b80000gn/T//Rtmp26icVx/downloaded_packages
installing the source package ‘glmnet’


Warning message in utils::install.packages(package, ...):
“installation of package ‘glmnet’ had non-zero exit status”

bestglm installed

Warning message:
“package ‘bestglm’ was built under R version 4.0.2”
Installing package into ‘/Users/sahluwalia/Library/R/4.0/library’
(as ‘lib’ is unspecified)

  There is a binary version available but the source version is later:
       binary source needs_compilation
glmnet  4.1-3  4.1-6              TRUE

installing the source package ‘glmnet’


Warning message in utils::install.packages(package, ...):
“installation of package ‘glmnet’ had non-zero exit status”
Warning message in p_install(package, character.only = TRUE, ...):
“”
Warning message in library(package, lib.loc = lib.loc, character.only = TRUE, logical.re
turn = TRUE, :
“there is no package called ‘glmnet’”
Installing package into ‘/Users/sahluwalia/Library/R/4.0/library’
(as ‘lib’ is unspecified)

Warning message:
“package ‘randomForest’ is not available (for R version 4.0.1)”
Warning message:
“'BiocManager' not available.  Could not check Bioconductor.

Please use `install.packages('BiocManager')` and then retry.”
Warning message in p_install(package, character.only = TRUE, ...):
```

```
""
Warning message in library(package, lib.loc = lib.loc, character.only = TRUE, logical.re
turn = TRUE, :
"there is no package called 'randomForest'"
Warning message in pacman::p_load(tidyverse, skimr, RCurl, data.table, bit64, stringr, :
"Failed to install/load:
bestglm, glmnet, randomForest"
```

Question

Can we predict who will donate money to a political campaign in 2020? While social scientists have invested notable time and resources into understanding how to mobilize voters and how to change political attitudes[1] there are few studies reviewing how to optimally target voters. In this short exercise I attempt to predict federal campaign contribution behavior in the state of North Carolina.

I implement a LASSO logistic regression as a baseline model. The response variable variable is a binary indicator for whether an individual donated money to a federal political campaign in 2020. The data for this short study amalgamates federal campaign individual contribution data from 2012 - 2019, North Carolina voting registration records, governmental socio-demographic county data, and 2016 U.S. presidential election results.

1. Gerber, Alan S., and Donald P. Green. 2000. "The Effects of Canvassing, Phone Calls, and Direct Mail on Voter Turnout: A Field Experiment." The American Political Science Review 94 (3): 653-663.

This study merges data from four sources:

1. First, we utilize data from the North Carolina Voter Registration, which contains comprehensive administrative data on every individual registered to vote in the state of North Carolina. The data includes information on all voters including name, sex, date of birth, date of registration, and registered political party. I also impute the race of each voter using each voter's name and address.
2. Second, we access all individual-level campaign contributions to federal elections (2012-2020) from individuals living in North Carolina. This data comes from the North Carolina State Board of Elections, which is available for public download. This data includes individual-level. The raw data includes the name of the contributor, the contributor's zip code, the contributor's employer, the contributor's occupation, the date of the contribution, the donation amount, and which candidate or political organization received the donation.
3. Third, I draw on socio-demographic data from the American Community Survey (ACS) 2015-2019 (5-year estimates). . The ACS is an ongoing governmental survey that provides important information on a yearly basis about the United States. The survey results help determine how more than $675 billion in federal and state funds are distributed each year. For this analysis, I downloaded county-level data on the total population, median income, percent white, percent Black, percent Hispanic, percent noncitizen, and percent college educated.
4. Finally, I use county-level election results from the 2016 U.S. Presidential Election. The data come from the MIT Election Lab.

Transformations and data-preprocessing

Brief comments on the data pre-processing:

1. The final data is a merge of the aforementioned sources, with additional covariates representing donations from 2012 - 2019, as well as a new donor status - indicating whether or not the registered vo
2. After performing some basic transformations on the raw data for each year of individual contributions, I grouped each donation by first name, last name, and zip code and then aggregate the total donation amount per year. After joining the donation data with the North Carolina voter registrations, the resulting dataframe is a unique row for each individual and their complete donation and demographic information.
3. Because the North Carolina data includes an individual's first name, last name, and zip code, I cannot be certain that duplicates in the North Carolina voter file are associated with a unique donation history. As a result, I resort to dropping all duplicate records from our data. Duplicates were defined as multiple records with the same first name, last name, age, race, donations from 2012 - 2020, inclusive, and living in the same zip code.
4. Any voter registration record that did not have any donation records was filled with zeros for the donation amount over time. To handle NA values for the other numeric or integer variables, I replaced them with the mean value for that variable. To handle NA values for categorical variables, I replaced them with the mode value for that variable.

HUD zip code and county data (for crosswalk to add counties)

https://www.huduser.gov/portal/datasets/usps_crosswalk.html

In [2]:
```
zip_to_county_data <-
  read_excel("../data/ZIP_COUNTY_122020.xlsx") %>%
  select(1, 2) %>%
  # limit to unique zip codes (bc several zip codes lie in multiple counties)
  distinct_at(vars(ZIP),.keep_all = T)

zip_to_county_data$COUNTY<-as.character(zip_to_county_data$COUNTY)
```

2016 County-Level U.S. Presidential Election Data

In [3]:
```
election_pres_raw <-
  read.csv("../data/countypres_2000-2020.csv")


election_2016_presidential <- election_pres_raw %>%
  # limit election data to only 2016
  filter(year == 2016, candidate %in% c("HILLARY CLINTON", "DONALD TRUMP")) %>%
  group_by(county_fips, candidate, totalvotes) %>%
  summarize(candidate_total = sum(candidatevotes))

election_2016_presidential_clean <- election_2016_presidential %>%
  mutate(county_fips = as.character(county_fips),
         vote_share = candidate_total / totalvotes,
         county_fips = case_when(nchar(county_fips) == 4 ~ paste0("0", county_fips),TRUE
  ) %>%
  filter(candidate == "DONALD TRUMP") %>% # for  now, limit to just TRUMP
  ungroup() %>%
  select(county_fips,totalvotes,trump_voteshare2016 = "vote_share")
```

```
`summarise()` has grouped output by 'county_fips', 'candidate'. You can override using t
he `.groups` argument.
```

In [4]:
```
head(election_2016_presidential_clean)
```

A tibble: 6 × 3

county_fips	totalvotes	trump_voteshare2016

<chr>	<int>	<dbl>
01001	24973	0.7276659
01003	95215	0.7654571
01005	10469	0.5209667
01007	8819	0.7640322
01009	25588	0.8933484
01011	4710	0.2420382

ACS 2015-2019 5-year survey

In [5]:

```
ACS2019_raw <-read.csv("../data/acs_clean_final_pivoted.csv")  %>%
  separate(NAME,c("County", "State") ,",")
ACS2019_raw$GEOID <-as.character(ACS2019_raw$GEOID)

final_acs_zip <- ACS2019_raw %>%
  # add column for county fips code
  left_join(zip_to_county_data,by = c("GEOID" = "COUNTY"))
```

In [6]:

```
head(final_acs_zip)
```

	X	GEOID	County	State	avg_renter_household_size	med_gross_rent	med_home_value	median_
	<int>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<c
1	0	37001	Alamance County	North Carolina	2.29	875	168900	
2	0	37001	Alamance County	North Carolina	2.29	875	168900	
3	0	37001	Alamance County	North Carolina	2.29	875	168900	
4	0	37001	Alamance County	North Carolina	2.29	875	168900	
5	0	37001	Alamance County	North Carolina	2.29	875	168900	
6	0	37001	Alamance County	North Carolina	2.29	875	168900	

North Carolina Contributions

In [7]:

```
nc_contributions <- read.csv("../data/nc_contributions_clean.csv")

nc_contributions_final <- nc_contributions %>%
  # Select only columns relevant for analysis
  # data wrangling
  group_by(year_of_donation, first_name, last_name, Zip.Code, employer, Profession.Job.T
  summarize(donation_amount = sum(Amount))
```

```
`summarise()` has grouped output by 'year_of_donation', 'first_name', 'last_name', 'Zip.
Code', 'employer'. You can override using the `.groups`
argument.
```

In [8]:

```
nc_contributions_final$year_of_donation  <- as.character(nc_contributions_final$year_of_
```

In [9]:

```
nc_contributions_ready <- nc_contributions_final %>% filter(first_name != "" ) %>%
#   # create variables for donation amount per year_of_donation
```

```
  mutate(
          donation_amount_2012 = case_when(year_of_donation == "2012" ~ donation_amount),
          donation_amount_2013 = case_when(year_of_donation == "2013" ~ donation_amount),
          donation_amount_2014 = case_when(year_of_donation == "2014" ~ donation_amount),
          donation_amount_2015 = case_when(year_of_donation == "2015" ~ donation_amount),
          donation_amount_2016 = case_when(year_of_donation == "2016" ~ donation_amount),
          donation_amount_2017 = case_when(year_of_donation == "2017" ~ donation_amount),
          donation_amount_2018 = case_when(year_of_donation == "2018" ~ donation_amount),
          donation_amount_2019 = case_when(year_of_donation == "2019" ~ donation_amount),
          donation_amount_2020 = case_when(year_of_donation == "2020" ~ donation_amount)
) %>%

  group_by(first_name, last_name, Zip.Code) %>%
  # sum total donation amount per year (so each donor has a single row)
  summarize(
            donation_amount_2012 = sum(donation_amount_2012),
            donation_amount_2013 = sum(donation_amount_2013),
            donation_amount_2014 = sum(donation_amount_2014),
            donation_amount_2015 = sum(donation_amount_2015),
            donation_amount_2016 = sum(donation_amount_2016),
            donation_amount_2017 = sum(donation_amount_2017),
            donation_amount_2018 = sum(donation_amount_2018),
            donation_amount_2019 = sum(donation_amount_2019),
            donation_amount_2020 = sum(donation_amount_2020),
            Profession.Job.Title = Profession.Job.Title,
            employer = employer
  ) %>%
  ungroup() %>%
  rowid_to_column("ID")

# in hindsight could have just done this above
nc_contributions_ready[c("donation_amount_2012",
                          "donation_amount_2013",
                            "donation_amount_2014",
                            "donation_amount_2015",
                            "donation_amount_2016",
                            "donation_amount_2017",
                            "donation_amount_2018",
                            "donation_amount_2019",
                         "donation_amount_2020")][is.na(nc_contributions_ready[c(
                            "donation_amount_2012",
                          "donation_amount_2013",
                            "donation_amount_2014",
                            "donation_amount_2015",
                            "donation_amount_2016",
                            "donation_amount_2017",
                            "donation_amount_2018",
                            "donation_amount_2019",
                         "donation_amount_2020"
  ))] <- 0
```

```
`summarise()` has grouped output by 'first_name', 'last_name', 'Zip.Code'. You can overr
ide using the `.groups` argument.
```

In [10]:

```
# Used to compute Mode
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Helper function to calculate mode employer and occupation per donor

# Calculate mode of occupation per donor - so each donor has one occupation/employer
occupation <- nc_contributions_ready %>%
mutate(Profession.Job.Title = case_when(Profession.Job.Title == "" ~ "NONE",TRUE ~ Profe
```

```
cSplit("Profession.Job.Title", sep= " | ") %>%
pivot_longer(cols = starts_with("Profession.Job.Title"),
             names_to = "occupation", values_to = "occupation_name") %>%
filter(!is.na(occupation_name)) %>%
group_by(ID,first_name,last_name,employer) %>%
mutate(occupation_name = str_remove(occupation_name,"\\| "),
       occupation_name = str_remove(occupation_name," \\|"),
       occupation_mode = getmode(occupation_name)) %>%
distinct(ID,first_name,last_name,employer,occupation_mode) %>%
ungroup() %>%
mutate(occupation_mode = case_when(occupation_mode == "" ~ "NONE",TRUE ~ occupation_mode
select(ID, occupation_mode)
```

In [11]:

```
# Calculate mode of employer per donor
employer <- nc_contributions_ready %>%
    mutate(employer = case_when(employer == "" ~ "NONE",TRUE ~ employer)) %>%
    pivot_longer(cols = starts_with("employer"),names_to = "employer", values_to = "empl
    filter(!is.na(employer_name)) %>%
    group_by(ID,first_name,last_name, Profession.Job.Title) %>%
    mutate(employer_name = str_remove(employer_name,"\\| "),
           employer_name = str_remove(employer_name," \\|"),
           employer_mode = getmode(employer_name)) %>%
    distinct(ID,first_name, last_name, Profession.Job.Title, employer_mode) %>%
    ungroup() %>%
    mutate(employer_mode = case_when(employer_mode == "" ~ "NONE",TRUE ~ employer_mode))
select(ID, employer_mode)

# join together employer and occupation
employer_occupation <- occupation %>%
left_join(employer, by ="ID")
```

In [12]:

```
# join employer_occupation to fec
data_full <- nc_contributions_ready %>%
left_join(employer_occupation, by = "ID") %>%
select(-employer,-Profession.Job.Title)
```

In [13]:

```
head(data_full)
```

ID	first_name	last_name	Zip.Code	donation_amount_2012	donation_amount_2013	donation_amount_2(
<int>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<d
1	a	allen	28785	0	0	
2	a	amero	28607	0	0	
3	a	barnes	27896	0	0	
4	a	barnes	27896	0	0	
5	a	biggerstaff	28168	0	0	
6	a	biggerstaff	28168	0	0	

North Carolina Voter Registration

In [14]:

```
nc_voter_registation <- read.csv("../data/final_voter_registrations_clean.csv")
nc_voter_registation$zip_code_final <- as.character(nc_voter_registation$zip_code_final)
```

In [15]:

```
head(nc_voter_registation)
```

A data.frame: 6 × 11

	X	race_code	party_cd	first_name	last_name	gender_code	ethnic_code	birth_age	registr_dt	pre
	<int>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<chr>	
1	0	W	UNA	ruth	aabel	F	NL	87	1984-10-01	
2	1	W	REP	timothy	aarmstrong	M	UN	56	2020-10-31	
3	2	W	UNA	christina	aaron	F	UN	46	1996-03-26	
4	3	W	UNA	claudia	aaron	F	NL	77	1989-08-15	
5	4	W	DEM	james	aaron	M	UN	74	2012-03-07	
6	5	B	DEM	kimberly	aaron	F	NL	56	2020-06-01	

In [16]:
```
# Join ACS with Presidential data
election_acs_join <- election_2016_presidential_clean %>%
  left_join(final_acs_zip,by=c("county_fips" = "GEOID"))
```

In [17]:
```
# Join voter file with donation data
voter_file_contribution_join <- nc_voter_registation %>%
  left_join(data_full, by = c("zip_code_final" = "Zip.Code","first_name","last_name"))
```

In [18]:
```
final_model_data <- voter_file_contribution_join %>%
  left_join(election_acs_join, by= c("zip_code_final" = "ZIP"))
```

In [19]:
```
# initial number of voters in the sample that were matched
dim(final_model_data)
```

7702951 · 47

In [20]:
```
# this approach for matching records across datasets was not the most efficient
# the most satisfactory results
final_model_data <- subset(final_model_data, (!is.na(final_model_data[,13:21])))
```

In [21]:
```
final_model_data <- final_model_data %>% distinct(
    first_name, last_name, race_code, birth_age, donation_amount_2012,
    donation_amount_2013, donation_amount_2014, donation_amount_2015,
    donation_amount_2016, donation_amount_2017, donation_amount_2018,
    donation_amount_2019, zip_code_final, donation_amount_2020, .keep_all= TRUE)
```

In [22]:
```
# I am being aggressive with removing duplicates (compare to the above, prior to selecti
dim(final_model_data)
```

69845 · 47

In [23]:
```
# Data preprocessing

today <- as.Date(Sys.Date(), format = "%Y-%m-%d")

cleaned_up_final_model_data <- final_model_data %>%
  mutate(
         # Calculate total donation amount per donor
         donation_total = rowSums(final_model_data[,13:21]),
         # Calculate total donation amount through 2019 per donor
         donation_through_2019 = rowSums(final_model_data[,13:20]),
         # calculate donation sums by presidency
```

```
          donation_2012_through_2015 = rowSums(final_model_data[,13:16]),
          donation_2016_through_2019 = rowSums(final_model_data[,17:20]),
          # cast variables to factor
          gender = as.factor(gender_code),
          political_party = as.factor(party_cd),
          county = as.factor(County),
          political_party = as.factor(party_cd),
          # calculate days since date of voter registration
          date_of_registration = as.Date(registr_dt, format = "%Y-%m-%d"),
          # binary indicator if an individual donated money in 2016
          donation_2016_binary = as.factor(case_when(donation_amount_2016 > 0 ~ as.intege
          # binary indicator if an individual donated money in 2020
          donation_2020_binary = as.factor(case_when(donation_amount_2020 > 0 ~ as.intege
          # binary indicator if an individual donated money for the first time in 2020
          new_donor = case_when(donation_through_2019 == 0 & donation_amount_2020 > 1 ~ a
          # fill NA values for employer and occupation with "NONE"
          employer_mode = as.character(employer_mode),
          occupation_mode = as.character(occupation_mode),
          employer_mode = case_when(is.na(employer_mode) ~ "NONE", TRUE ~ employer_mode),
          occupation_mode = case_when(is.na(occupation_mode) ~ "NONE", TRUE ~ occupation_
          employer_mode = as.factor(employer_mode),
          occupation_mode = as.factor(occupation_mode),
          # reduce the number of categories for occupation and employer factors
          occupation_mode = fct_lump_min(occupation_mode,220),
          employer_mode = fct_lump_min(employer_mode,103)) %>%
  # drop variables not needed for analysis
  select(-X.x,
          -date_of_registration, -total_pop, -registr_dt,
           -State, -ID, -county_fips, -X.y, total_pop,
          -precinct_abbrv,-County)
```

In [24]:

```
# sanity check
cleaned_up_final_model_data[, 13:21][is.na(cleaned_up_final_model_data[, 13:21])] <- 0
### for loop to replace NAs
for (i in 1:ncol(cleaned_up_final_model_data)) {
  if (class(cleaned_up_final_model_data[[i]]) == "numeric") {
    # replace NA with mean
    cleaned_up_final_model_data[[i]] <-
      case_when(is.na(cleaned_up_final_model_data[[i]]) ~
             mean(cleaned_up_final_model_data[[i]], na.rm = TRUE), TRUE ~
             cleaned_up_final_model_data[[i]])
  } else if (class(cleaned_up_final_model_data[[i]]) == "factor"){
    # replace NA with mode
    cleaned_up_final_model_data[[i]] <-
      case_when(is.na(cleaned_up_final_model_data[[i]]) ~
             getmode(cleaned_up_final_model_data[[i]]), TRUE ~
             cleaned_up_final_model_data[[i]])

  } else if (class(cleaned_up_final_model_data[[i]]) == "integer"){
    # replace NA with mean
    cleaned_up_final_model_data[[i]] <-
      case_when(is.na(cleaned_up_final_model_data[[i]]) ~
             as.integer(mean(cleaned_up_final_model_data[[i]], na.rm = TRUE)), TRUE ~
             cleaned_up_final_model_data[[i]])
  }
}
```

In [25]:

```
head(cleaned_up_final_model_data)
```

	race_code	party_cd	first_name	last_name	gender_code	ethnic_code	birth_age	zip_code_final	don
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<chr>	
92	W	UNA	george	abernathy	M	NL	75	27244	

182	W	REP	barbara	acosta	F	NL	59	27217
257	W	DEM	devlin	adams	F	NL	47	27217
337	W	DEM	mary	adams	F	NL	48	27215
338	W	REP	mary	adams	F	NL	70	27215
743	W	UNA	daniel	albaugh	M	NL	63	27349

This step is commented out in order to cache in the event we need to revise any other figures

In [26]:
```
# write.csv(cleaned_up_final_model_data, "../data/post_final_cleaned_up_final_model_data
```

In [27]:
```
# cleaned_up_final_model_data <- read.csv("../data/post_final_cleaned_up_final_model_dat
```

In [28]:
```
# There are 11, 176 NC voters who donated to a federal election in 2020
# (i.e., our target variable), which is about ~16 percent of voters.

# Note that the proportion that donated in 2016 is much smaller (~3.8%)

# Summary Statistics
skimtable <- skimr :: skim(cleaned_up_final_model_data)
skimtable_factor <- skimtable %>%
  select(skim_type,skim_variable,factor.top_counts) %>%
  filter(skim_type == "factor")
skimtable_numeric <- skimtable %>%
  select(skim_type,skim_variable,numeric.mean,numeric.sd) %>%
  filter(skim_type == "numeric")
# Numeric summary stats
table1 <- xtable(skimtable_numeric, caption = "Table 1")
# Categorical summary stats
table2 <- xtable(skimtable_factor)

# load zip code and county data (for crosswalk to add counties)
zip_to_county_data <-
  read_excel("../data/ZIP_COUNTY_122020.xlsx") %>%
  select(1, 2) %>%
  # limit to unique zip codes (bc several zip codes lie in multiple counties)
  distinct_at(vars(ZIP),.keep_all = T)
# join full data with zip code data to add zip code
cleaned_up_final_model_data <- cleaned_up_final_model_data %>%
  # add column for county fips code
  left_join(zip_to_county_data,by = c("zip_code_final" = "ZIP"))
```

In [29]:
```
# breakdown of response variables and other categorical variables
table2
```

A xtable: 7 × 3

skim_type	skim_variable	factor.top_counts
<chr>	<chr>	<chr>
factor	occupation_mode	Oth: 31029, Rea: 21013, RET: 5157, Ret: 1092
factor	employer_mode	Oth: 47383, RET: 4043, NOT: 2107, SEL: 1737
factor	gender	M: 34160, F: 33814, U: 1871

factor	political_party	REP: 27494, DEM: 24121, UNA: 17916, LIB: 301
factor	county	Wak: 9066, Mec: 5305, Ora: 3230, Gui: 3188
factor	donation_2016_binary	0: 67210, 1: 2635
factor	donation_2020_binary	0: 58669, 1: 11176

In [30]:

```
# summary statistics
head(table1, 10)
```

A xtable: 10 × 4

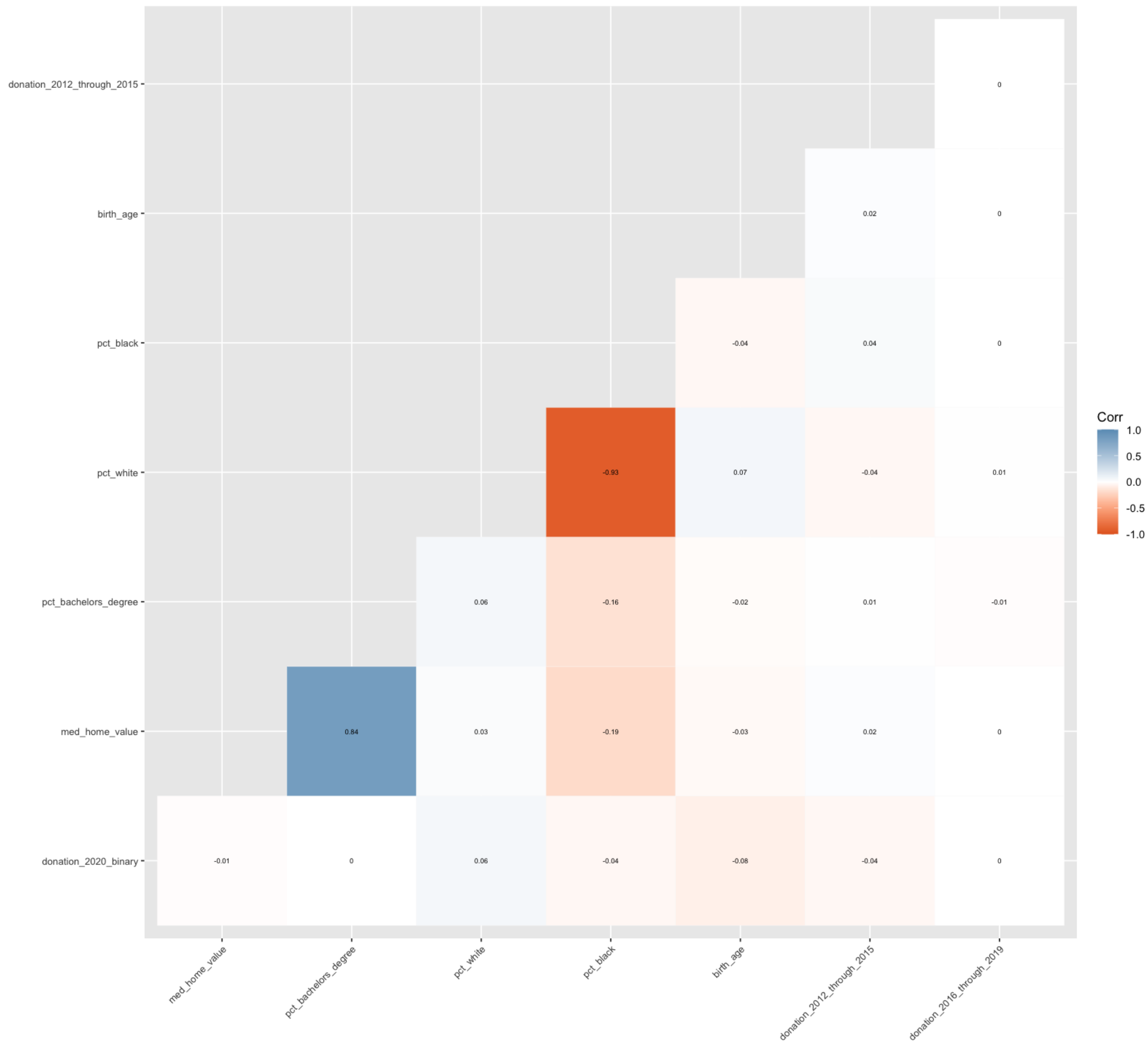
	skim_type	skim_variable	numeric.mean	numeric.sd
	<chr>	<chr>	<dbl>	<dbl>
1	numeric	birth_age	60.148001	14.88763
2	numeric	donation_amount_2012	12.564678	132.62637
3	numeric	donation_amount_2013	2.745166	79.73403
4	numeric	donation_amount_2014	4.163414	121.39888
5	numeric	donation_amount_2015	4.173649	133.29141
6	numeric	donation_amount_2016	9.773249	409.45214
7	numeric	donation_amount_2017	6.900171	109.96082
8	numeric	donation_amount_2018	154.073515	38678.68113
9	numeric	donation_amount_2019	10.270439	498.93881
10	numeric	donation_amount_2020	33.681983	757.02088

The correlation matrix below provides some cursory details on which various covariates are associated with donating in 2020. Past donation behavior, county-level median home value, county-level percent college education, donor's age, and donor's self-identified race as white show very little association with donating to a federal campaign in 2020. Donor's estimated race as black is negatively associated with donating to a federal campaign in 2020.

In [31]:

```
options(repr.plot.width=15, repr.plot.height=12)

corr_data <- cleaned_up_final_model_data %>%
  select(donation_2020_binary,
         med_home_value,
         pct_bachelors_degree, pct_white, pct_black,
         birth_age,donation_2012_through_2015,donation_2016_through_2019) %>%
  mutate(donation_2020_binary = as.numeric(donation_2020_binary))
### Correlation matrix
corr <- round(cor(corr_data, use = "complete.obs"), 2)
ggcorrplot(corr, type = "lower", lab = TRUE,
           outline.col = "white",
           ggtheme = ggplot2::theme_gray,
           colors = c("#E46726", "white", "#6D9EC1"),
           lab_col = "black", lab_size = 2,
           tl.cex = 8, tl.col = "black")
```

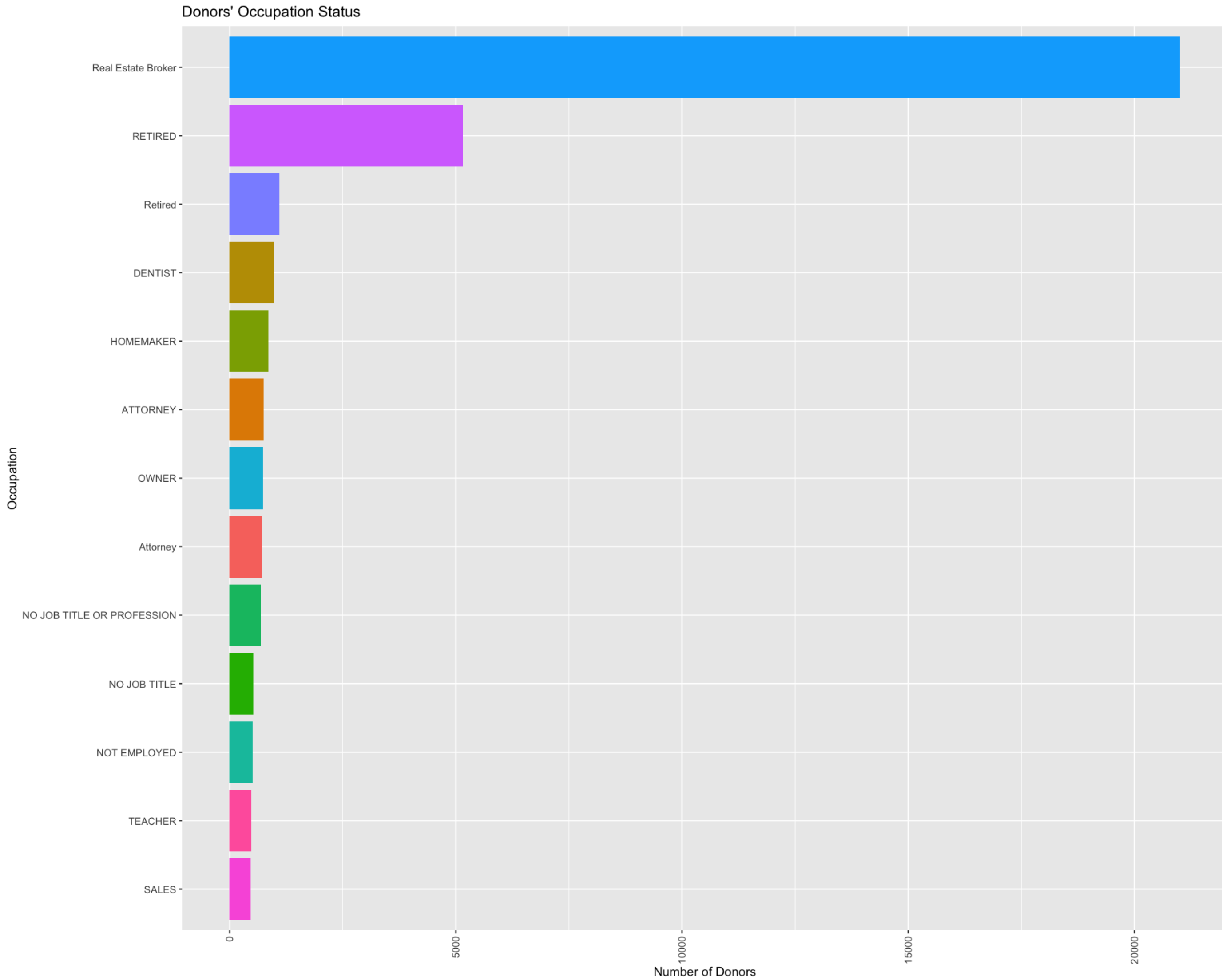


Those that donated less than 1000 USD between 2016 - 2019 were not likely to donate in 2020

In [32]:
```
cleaned_up_final_model_data %>%
  filter(donation_2016_through_2019 > 0 & donation_2016_through_2019 < 1000) %>%
  ggplot(aes(x = donation_2016_through_2019, y = donation_2020_binary)) +
  geom_jitter(height = 0.55, aes(color = as.factor(donation_2020_binary))) +
  labs(y = "2020 Donation Indicator",
       x = "Total Individual Donation Amount in US Dollars (2016-2019)",
       title = "Relationship between Past Individual Donations and 2020 Donations (less
#   theme(legend.position = "None")
```

We can see that normalizing vocations is an additional step that we need to take (Retired versus RETIRED, for example):

In [33]:
```
cleaned_up_final_model_data %>%
  filter(occupation_mode != "W" & occupation_mode != "NONE" & occupation_mode != "Other"
  group_by(occupation_mode) %>%
  count() %>%
  filter(n > 450) %>%
  ggplot(aes(x = forcats::fct_reorder(occupation_mode,n), y = n, fill = occupation_mode)
  geom_col() +
  labs(y = "Number of Donors",
       x = "Occupation",
       title = "Donors' Occupation Status") +
  theme(legend.position = "None",
        axis.text.x = element_text(
          angle = 90,
          vjust = 0.5,
          hjust = 1)) +
  coord_flip()
```


Donors' Occupation Status
Real Estate Broker
RETIRED
Retired
DENTIST
HOMEMAKER
ATTORNEY
OWNER
Attorney
NO JOB TITLE OR PROFESSION
NO JOB TITLE
NOT EMPLOYED
TEACHER
SALES
Occupation
0
5000
10000
15000
20000
Number of Donors