Final Project - Bank Dataset

Aaron Abromowitz, Stephanie Duarte, Dammy Owolabi

2024-04-20

Youtube Link: <https://youtu.be/oeHvLTXBvNQ?si=Wxctb9p1mOr-wJOV>

# EDA

The first thing that is done when you get a dataset is to peform an Exploratory Data Anslysis, to see what characteristics the data has. The variable we are trying to predict is the y column whic represents if the client has subscribed to a term deposit. The possible values are “yes” or “no”.

## Create training and test set

Going forward, we will use the training set for all analysis and model building. The test set will be used at the end to get metrics for the various models we create.

# Pull in data  
data<-read.csv('https://raw.githubusercontent.com/stedua22/6372-Project-2/main/bank-additional-full.csv',stringsAsFactors = T, sep=";")  
  
# Set levels to use for later  
data$y <- relevel(data$y, ref="yes")  
data$month <- factor(data$month, levels=c('mar','apr','may','jun','jul','aug','sep','oct','nov','dec'))  
data$day\_of\_week <- factor(data$day\_of\_week, levels=c('mon','tue','wed','thu','fri'))  
  
# Duration was removed since the dataset explanation file said that it was created after y variable was known, so shouldn't be used for prediction.  
data$duration <- c()  
  
# Create the train and test split  
train\_perc <- .8  
set.seed(1234)  
train\_indices <- sample(nrow(data), floor(train\_perc \* nrow(data)))  
train\_data <- data[train\_indices, ]  
nrow(train\_data)

## [1] 32950

test\_data <- data[-train\_indices, ]   
nrow(test\_data)

## [1] 8238

## Look at summary statistics for numeric variables

There are several numeric variables where we can look at the min/max, quartiles, median, and mean.

summary(train\_data$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 17.00 32.00 38.00 40.07 47.00 98.00

summary(train\_data$campaign)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 2.000 2.561 3.000 56.000

summary(train\_data$pdays)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 999.0 999.0 962.8 999.0 999.0

summary(train\_data$previous)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.0000 0.0000 0.1733 0.0000 7.0000

summary(train\_data$emp.var.rate)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -3.40000 -1.80000 1.10000 0.07483 1.40000 1.40000

summary(train\_data$cons.price.idx)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 92.20 93.08 93.75 93.57 93.99 94.77

summary(train\_data$cons.conf.idx)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -50.80 -42.70 -41.80 -40.51 -36.40 -26.90

summary(train\_data$euribor3m)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.634 1.344 4.857 3.614 4.961 5.045

summary(train\_data$nr.employed)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 4964 5099 5191 5167 5228 5228

## Look at summary statistics for categorical variables

There are several categorical variables. Summary statistics don’t make as much sense for them, but you can look at the distribution of values in the different categories.

summary(train\_data$job)

## admin. blue-collar entrepreneur housemaid management   
## 8283 7426 1189 836 2355   
## retired self-employed services student technician   
## 1391 1130 3185 712 5359   
## unemployed unknown   
## 827 257

summary(summary(train\_data$job))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 257.0 833.8 1290.0 2745.8 3728.5 8283.0

length(summary(train\_data$job))

## [1] 12

summary(train\_data$marital)

## divorced married single unknown   
## 3699 19937 9248 66

summary(summary(train\_data$marital))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 66 2791 6474 8238 11920 19937

length(summary(train\_data$marital))

## [1] 4

summary(train\_data$education)

## basic.4y basic.6y basic.9y high.school   
## 3345 1859 4803 7646   
## illiterate professional.course university.degree unknown   
## 16 4172 9738 1371

summary(summary(train\_data$education))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 16 1737 3758 4119 5514 9738

length(summary(train\_data$education))

## [1] 8

summary(train\_data$default)

## no unknown yes   
## 26060 6888 2

summary(summary(train\_data$default))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2 3445 6888 10983 16474 26060

length(summary(train\_data$default))

## [1] 3

summary(train\_data$housing)

## no unknown yes   
## 14918 794 17238

summary(summary(train\_data$housing))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 794 7856 14918 10983 16078 17238

length(summary(train\_data$housing))

## [1] 3

summary(train\_data$loan)

## no unknown yes   
## 27197 794 4959

summary(summary(train\_data$loan))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 794 2876 4959 10983 16078 27197

length(summary(train\_data$loan))

## [1] 3

summary(train\_data$contact)

## cellular telephone   
## 20989 11961

summary(summary(train\_data$contact))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 11961 14218 16475 16475 18732 20989

length(summary(train\_data$contact))

## [1] 2

summary(train\_data$month)

## mar apr may jun jul aug sep oct nov dec   
## 440 2098 11023 4233 5760 4931 450 572 3305 138

summary(summary(train\_data$month))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 138.0 480.5 2701.5 3295.0 4756.5 11023.0

length(summary(train\_data$month))

## [1] 10

summary(train\_data$day\_of\_week)

## mon tue wed thu fri   
## 6772 6513 6506 6874 6285

summary(summary(train\_data$day\_of\_week))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 6285 6506 6513 6590 6772 6874

length(summary(train\_data$day\_of\_week))

## [1] 5

summary(train\_data$poutcome)

## failure nonexistent success   
## 3438 28425 1087

summary(summary(train\_data$poutcome))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1087 2262 3438 10983 15932 28425

length(summary(train\_data$poutcome))

## [1] 3

## Examine bank client data

The dataset includes age, job, marital, education, default, housing, and loan columns, which were identified as client data.

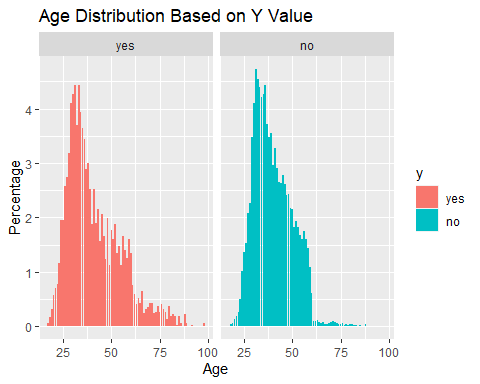
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.1 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

# Plot age  
summary <- train\_data %>%  
 group\_by(age,y) %>%  
 summarize(count=n())

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

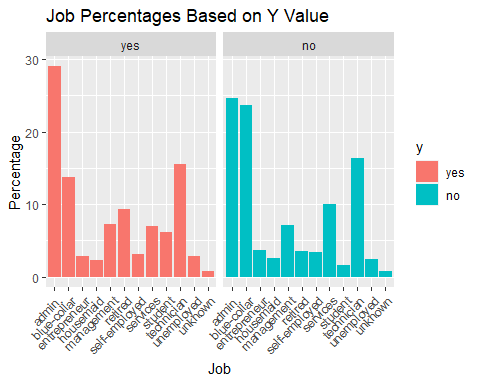
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=age,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Age') + ggtitle('Age Distribution Based on Y Value')



# Plot job  
summary <- train\_data %>%  
 group\_by(job,y) %>%  
 summarize(count=n())

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

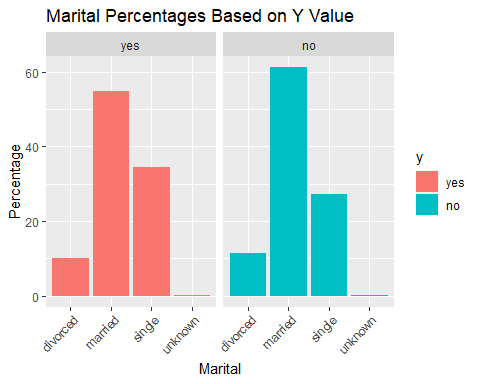
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=job,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Job') + ggtitle('Job Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Plot marital  
summary <- train\_data %>%  
 group\_by(marital,y) %>%  
 summarize(count=n())

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

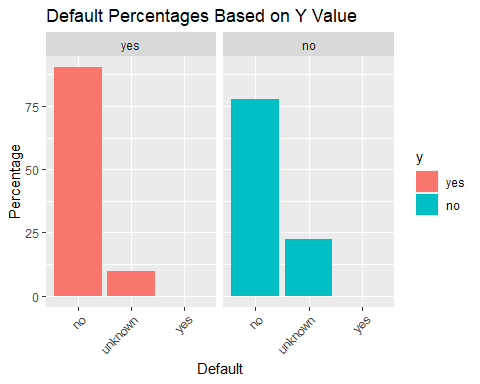
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=marital,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Marital') + ggtitle('Marital Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Plot default  
summary <- train\_data %>%  
 group\_by(default,y) %>%  
 summarize(count=n())

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

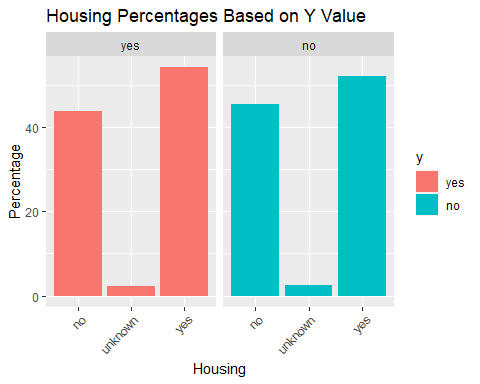
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=default,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Default') + ggtitle('Default Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Plot housing  
summary <- train\_data %>%  
 group\_by(housing,y) %>%  
 summarize(count=n())

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

summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=housing,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Housing') + ggtitle('Housing Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))

 None of these variables appear to show any strong indicator for yes or no. It looks like higher ages tend to lean more towards yes, and certain jobs (Ex: admin) lean more towards yes. When Y = no, there are more than about double the unknown values, but still far under 50%.

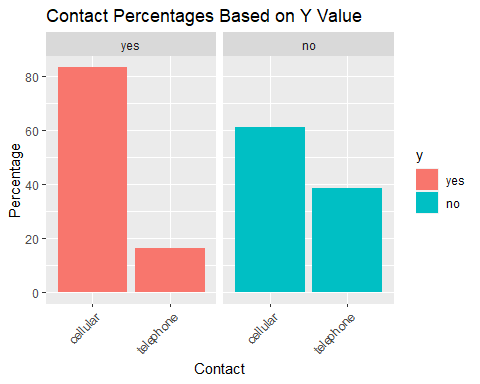
## Examine data related with the last contact of the current campaign

The dataset includes contact communication type, month of contact, and day of week of contac, for the last contact of the current campaign to sell term deposits.

# Plot contact  
summary <- train\_data %>%  
 group\_by(contact,y) %>%  
 summarize(count=n())

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

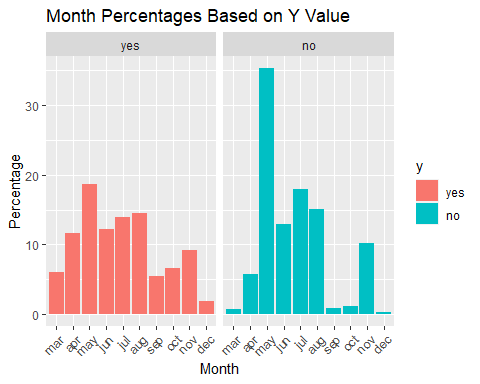
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=contact,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Contact') + ggtitle('Contact Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Plot month  
summary <- train\_data %>%  
 group\_by(month,y) %>%  
 summarize(count=n())

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

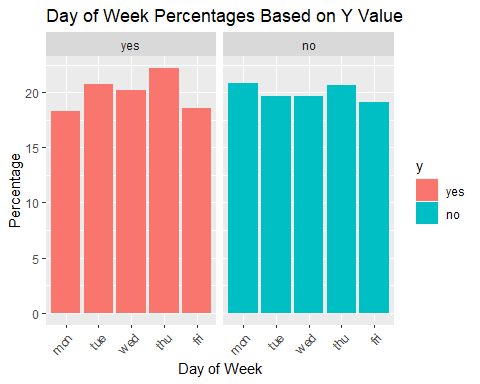
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=month,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Month') + ggtitle('Month Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Plot day of week  
summary <- train\_data %>%  
 group\_by(day\_of\_week,y) %>%  
 summarize(count=n())

## `summarise()` has grouped output by 'day\_of\_week'. You can override using the  
## `.groups` argument.

summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=day\_of\_week,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Day of Week') + ggtitle('Day of Week Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))

 If y is Yes, there are ~20% more likely to be contacted on your cell phone than on landline. There are certain months that also seem to have more term deposit sales in them. Day of week looks to not have much change.

## Examine other data related with the current campaign or previous campaigns

There are variables for number of contacts performed during this campaign and for this client (campaign), number of days that passed by after the client was last contacted from a previous campaign (pdays), number of contacts performed before this campaign and for this client (previous), and outcome of the previous marketing campaign (poutcome).

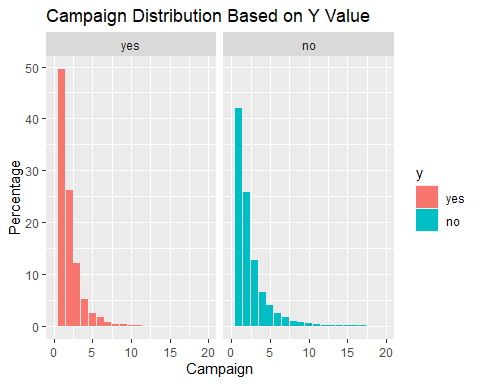
# Plot campaign  
summary <- train\_data %>%  
 group\_by(campaign,y) %>%  
 summarize(count=n())

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

summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=campaign,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Campaign') + ggtitle('Campaign Distribution Based on Y Value') + xlim(c(0,20))

## Warning: Removed 23 rows containing missing values (`position\_stack()`).

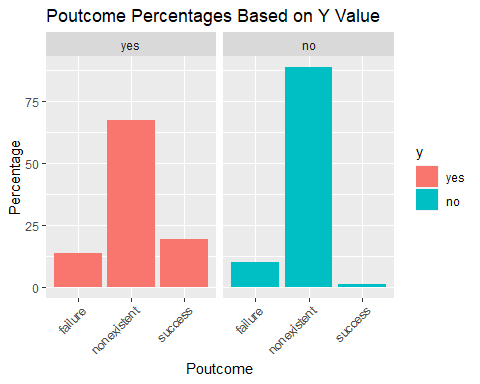
## Warning: Removed 1 rows containing missing values (`geom\_bar()`).



# Plot poutcome  
summary <- train\_data %>%  
 group\_by(poutcome,y) %>%  
 summarize(count=n())

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

summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=poutcome,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Poutcome') + ggtitle('Poutcome Percentages Based on Y Value') +   
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))

 Campaign distributions seem pretty similar between Yes and No. For Poutcome, success is more common for the yes than for no.

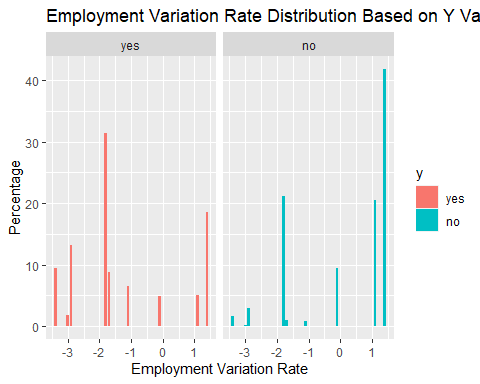
## Examine socio-economic data

There are variables for socio economic data for Employement Variation Rate, Consumer Price Index, Consumer Confidence Index, Euribor 3 month rate, and Numer of Employees.

# Plot emp.var.rate  
summary <- train\_data %>%  
 group\_by(emp.var.rate,y) %>%  
 summarize(count=n())

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

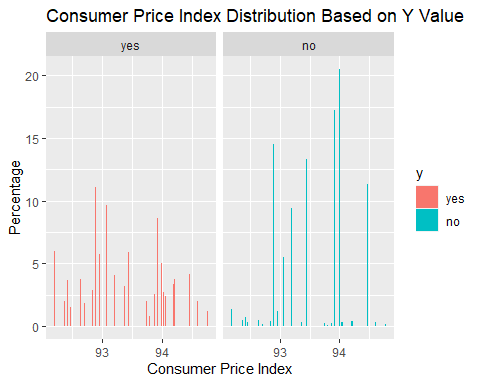
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=emp.var.rate,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Employment Variation Rate') + ggtitle('Employment Variation Rate Distribution Based on Y Value')



# Plot cons.price.idx  
summary <- train\_data %>%  
 group\_by(cons.price.idx,y) %>%  
 summarize(count=n())

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

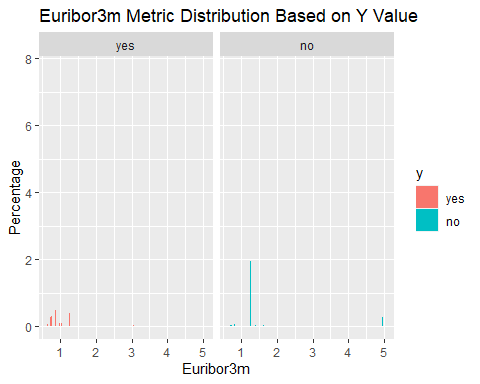
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=cons.price.idx,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Consumer Price Index') + ggtitle('Consumer Price Index Distribution Based on Y Value')



# Plot euribor3m  
summary <- train\_data %>%  
 group\_by(euribor3m,y) %>%  
 summarize(count=n())

## `summarise()` has grouped output by 'euribor3m'. You can override using the  
## `.groups` argument.

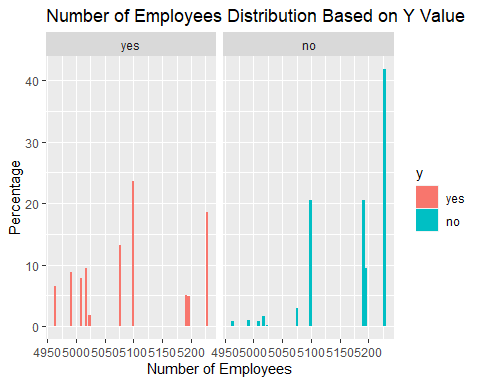
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=euribor3m,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Euribor3m') + ggtitle('Euribor3m Metric Distribution Based on Y Value')



# Plot nr.employed  
summary <- train\_data %>%  
 group\_by(nr.employed,y) %>%  
 summarize(count=n())

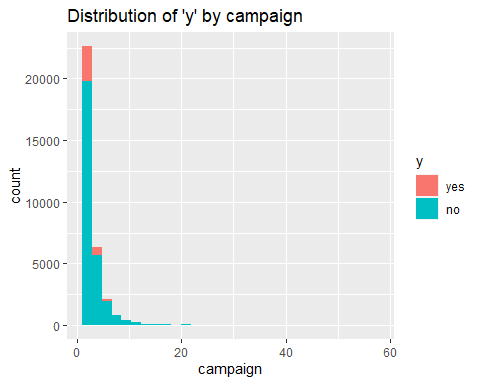
## `summarise()` has grouped output by 'nr.employed'. You can override using the  
## `.groups` argument.

summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data[train\_data$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data[train\_data$y == 'yes',]) \* 100  
summary %>% ggplot(aes(x=nr.employed,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Percentage') + xlab('Number of Employees') + ggtitle('Number of Employees Distribution Based on Y Value')

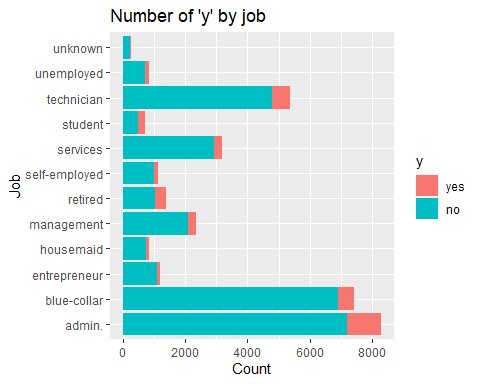


### Analyzing campaign  
  
ggplot(train\_data) +   
 geom\_histogram(mapping = aes(x=campaign, fill=y)) +   
 ggtitle("Distribution of 'y' by campaign")

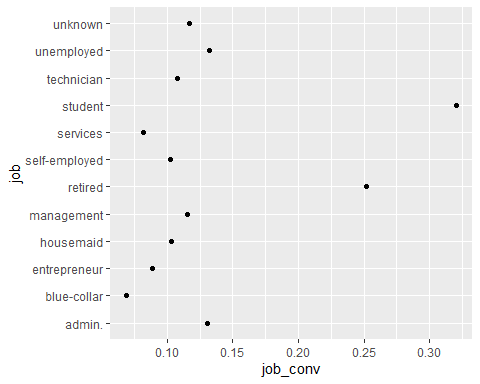
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



### Analyzing JOB  
  
ggplot(train\_data) +   
 geom\_bar(mapping = aes(x=job, fill = y)) +   
 coord\_flip() + #Added coord flip here to make it more readable  
 ggtitle("Number of 'y' by job") +   
 ylab("Count") +   
 xlab("Job")

 Admin, technician and blue collar jobs are the top 3 subscribers by volume

df2 <- train\_data %>%   
 group\_by(job) %>%   
 count(y) %>%   
 mutate(job\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
  
ggplot(df2, aes(x=job, y=job\_conv)) +   
 geom\_point() +   
 coord\_flip()

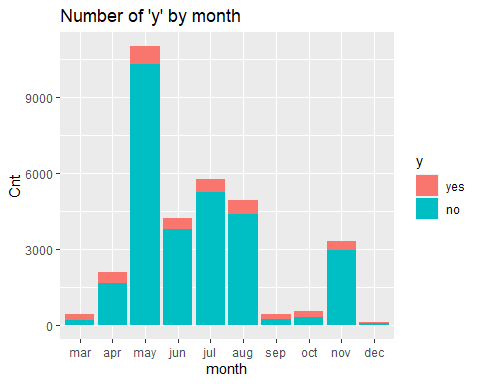


Above, I looked at the ratio of “yes” vs “no” and see that students and retired persons convert at much higher rates than those of other professions. And ‘blue collar’ has the lowest conversion rate

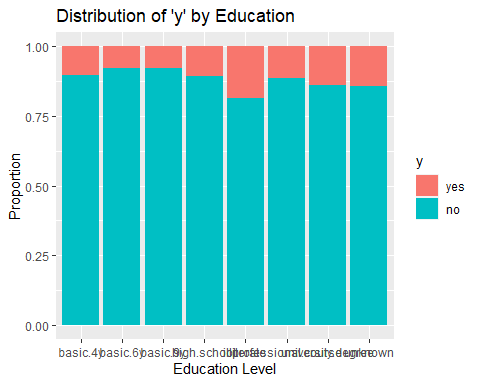
So, if they were to want to improve the cost effectiveness of their campaigns they might want to target more ‘students’ and ‘retirees’

### Analyzing By Month

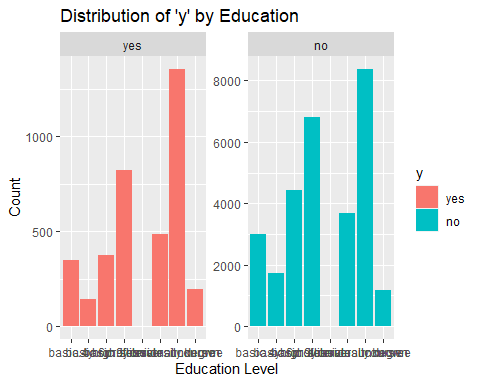
ggplot(train\_data) +   
 geom\_bar(mapping = aes(x=month, fill = y)) +   
 ggtitle("Number of 'y' by month") +  
 ylab("Cnt") + xlab("month")



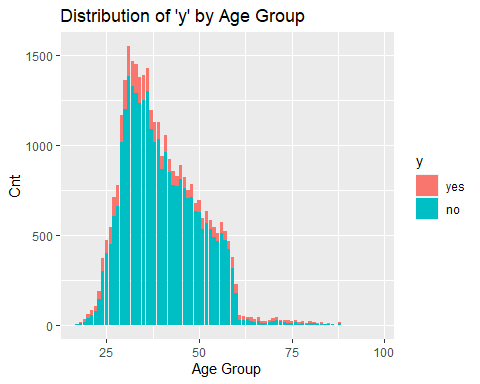
#Education  
ggplot(train\_data, aes(x = education, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Distribution of 'y' by Education") +   
 ylab("Proportion") +   
 xlab("Education Level")



ggplot(train\_data, aes(x = education, fill = y)) +   
 geom\_bar() +   
 facet\_wrap(~ y, scales = "free\_y") +   
 ggtitle("Distribution of 'y' by Education") +   
 ylab("Count") +   
 xlab("Education Level")

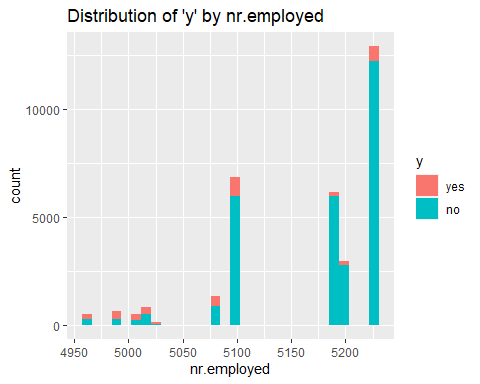


#Age  
ggplot(train\_data) +   
 geom\_bar(mapping = aes(x=age, fill = y)) +   
 ggtitle("Distribution of 'y' by Age Group") +   
 ylab("Cnt") +   
 xlab("Age Group")

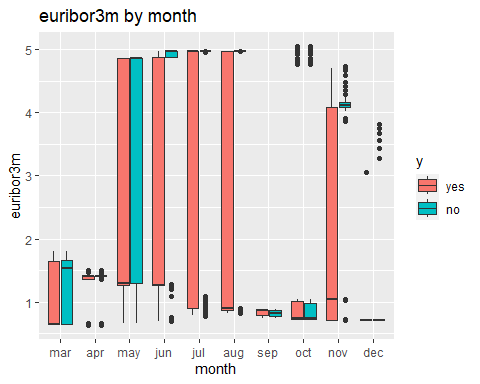


# by nr.employed  
ggplot(train\_data) + geom\_histogram(mapping = aes(x = nr.employed, fill = y)) +  
 ggtitle("Distribution of 'y' by nr.employed")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

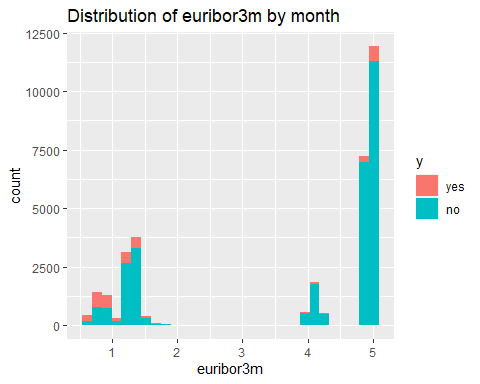


# Euribor 3 month rate  
ggplot(train\_data, aes(x = month , y = euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("euribor3m by month")



ggplot(train\_data) + geom\_histogram(mapping = aes(x = euribor3m, fill = y)) +  
 ggtitle("Distribution of euribor3m by month")

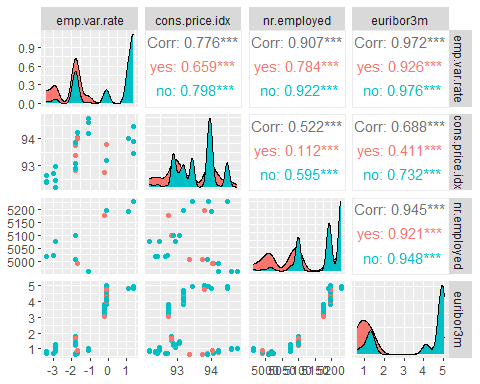
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# Correlations of socio-economic variables  
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

ggpairs(train\_data[,c('emp.var.rate','cons.price.idx','nr.employed','euribor3m')],aes(color = train\_data$y))

 The Consumer Price Index data seems to have more of a flat distribution for those with a term deposit. Employment Variation Rate, Euribor3m, and Number Employed all seem to have similar distributions where there is a higher percentage without term deposits for higher values of the index.

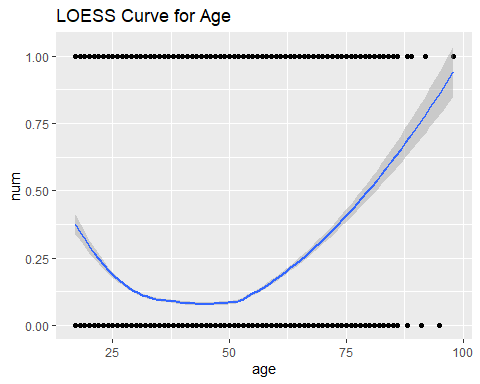
Looking at the paired correlations between out of the socio economic variables, we can see that Employment Variation Rate, Euribor3m, and Number Employed are highly correlated.

### LOESS Curves

LOESS curves can be useful to plot for numeric variables. They show if there is a general trend to higher or lower probabilities if the numeric variable increases. If they increase and then decrease, or vice-versa, this shows that the relationship between the two variables isn’t quite as simple.

# LOESS for Age  
train\_data$num <- ifelse(train\_data$y=="yes",1,0)  
train\_data %>% ggplot(aes(x=age,y=num)) +   
 geom\_point() + geom\_smooth(method="loess") +   
 ggtitle('LOESS Curve for Age')

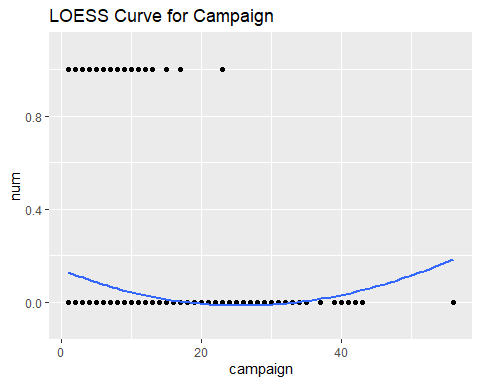
## `geom\_smooth()` using formula = 'y ~ x'



# LOESS for Campaign  
train\_data$num <- ifelse(train\_data$y=="yes",1,0)  
train\_data %>% ggplot(aes(x=campaign,y=num)) +   
 geom\_point() + geom\_smooth(method="loess", size = 1, span = 2, se = FALSE) +   
 ggtitle('LOESS Curve for Campaign') + ylim(c(-.1,1.1))

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## `geom\_smooth()` using formula = 'y ~ x'



# LOESS for nr.employed   
train\_data %>% ggplot(aes(x=nr.employed,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for nr.employed') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 5228.1

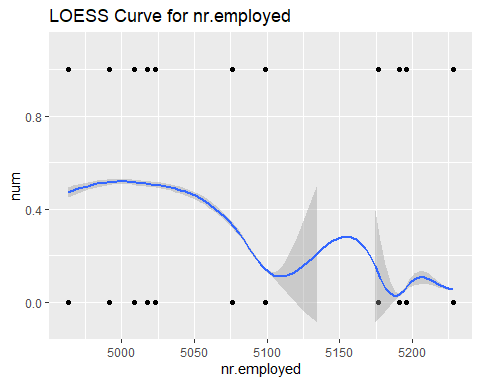
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 37.1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 1.4919e-14

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 5228.1

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 37.1

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 1.4919e-14



# LOESS for emp.var.rate   
train\_data %>% ggplot(aes(x=emp.var.rate,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for emp.var.rate')

## `geom\_smooth()` using formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 1.424

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 0.324

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 1.0203e-26

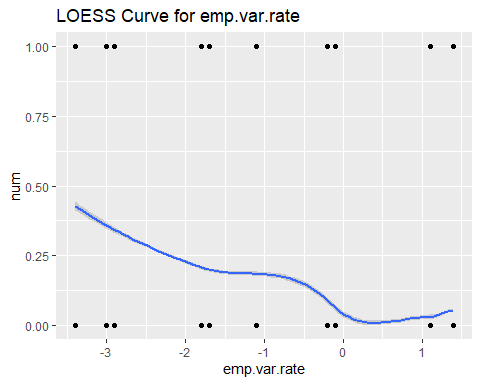
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : There are other near singularities as well. 0.09

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 1.424

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 0.324

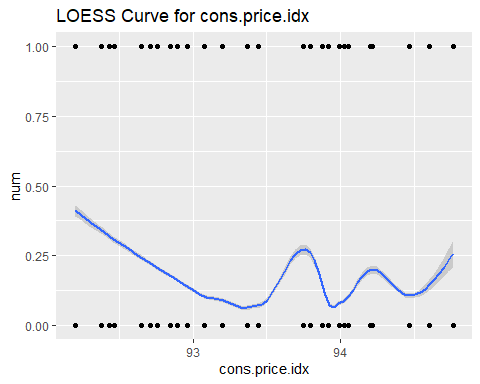
## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 1.0203e-26

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 0.09



# LOESS for cons.price.idx   
train\_data %>% ggplot(aes(x=cons.price.idx,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for cons.price.idx')

## `geom\_smooth()` using formula = 'y ~ x'

 The LOESS plots for Age and Campaign show that No becomes more likely as Age and Campaign increase, and then less likely. However, the LOESS plots for nr.employed, emp.var.rate, and cons.price.idx show a general downward trend toward a higher likelihood of No as those values increase.

# LOESS for nr.employed by Month  
train\_data %>% ggplot(aes(x=nr.employed,y=num,color = month)) +   
 geom\_point() + geom\_smooth(method="loess", size = 1, span=1.1) +   
 ggtitle('LOESS Curve for nr.employed by Month') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 5008.2

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 95.286

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 2.775e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : There are other near singularities as well. 9079.5

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 5008.2

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 95.286

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 2.775e-15

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 9079.5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 5008.2

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 95.286

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 5.8467e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : There are other near singularities as well. 9079.5

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 5008.2

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 95.286

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 5.8467e-15

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 9079.5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3  
  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 4963.3

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 56.813

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 3.1301e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : There are other near singularities as well. 3227.8

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 4963.3

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 56.813

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 3.1301e-15

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 3227.8

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3  
  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3  
  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3  
  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : Chernobyl! trL<k 3

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 5022.7

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 161.06

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 0

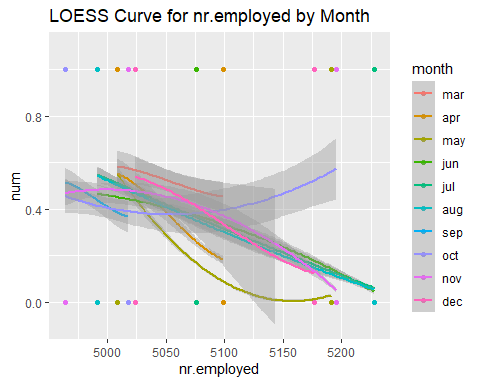
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : There are other near singularities as well. 25940

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 5022.7

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 161.06

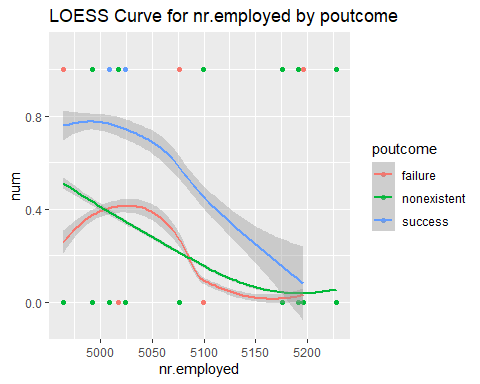
## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 0

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : There are other near  
## singularities as well. 25940



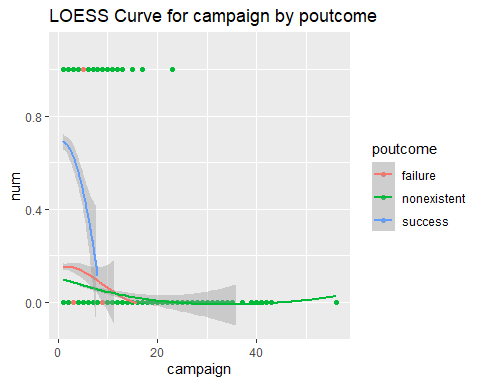
# LOESS for nr.employed by poutcome  
train\_data %>% ggplot(aes(x=nr.employed,y=num,color = poutcome)) +   
 geom\_point() + geom\_smooth(method="loess", size = 1, span = 1) +   
 ggtitle('LOESS Curve for nr.employed by poutcome') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'



# LOESS for campaign by poutcome  
train\_data %>% ggplot(aes(x=campaign,y=num,color = poutcome)) +   
 geom\_point() + geom\_smooth(method="loess", size = 1, span = 1.1) +   
 ggtitle('LOESS Curve for campaign by poutcome') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

 The LOESS curves for these variable combinations show that using these variables in the same model or even as an interaction between these two variables could be useful.

# LOESS for nr.employed squared  
train\_data %>% ggplot(aes(x=(nr.employed)^2,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for nr.employed Squared') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : pseudoinverse used at 2.7333e+07

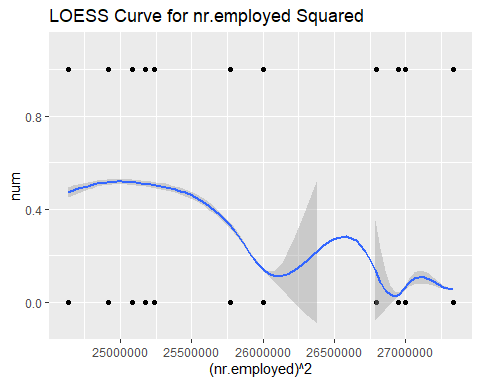
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : neighborhood radius 3.8655e+05

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,  
## : reciprocal condition number 2.0272e-14

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at  
## 2.7333e+07

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius  
## 3.8655e+05

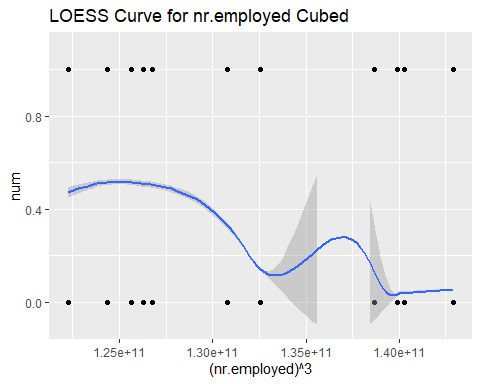
## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x  
## else if (is.data.frame(newdata))  
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition  
## number 2.0272e-14



# LOESS for nr.employed cubed  
train\_data %>% ggplot(aes(x=(nr.employed)^3,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for nr.employed Cubed') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

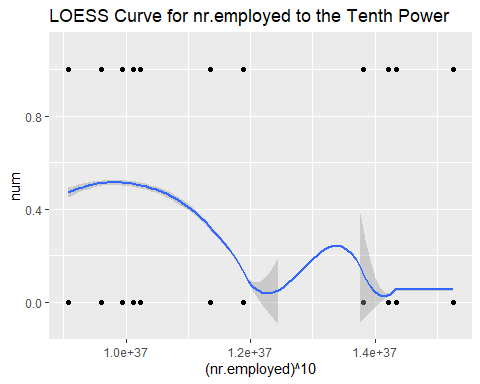
## Warning: Removed 10 rows containing missing values (`geom\_smooth()`).



# LOESS for nr.employed tenth power  
train\_data %>% ggplot(aes(x=(nr.employed)^10,y=num)) +   
 geom\_point() + geom\_smooth(method="loess",span=.5) +   
 ggtitle('LOESS Curve for nr.employed to the Tenth Power') +   
 ylim(c(-.1,1.1))

## `geom\_smooth()` using formula = 'y ~ x'

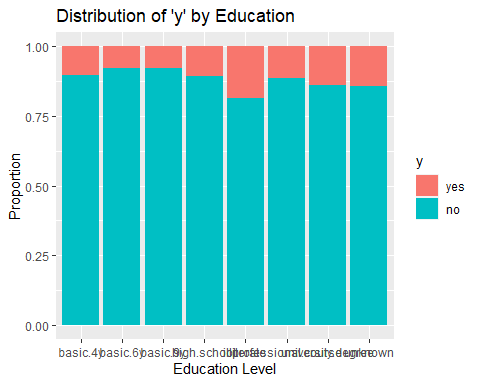
## Warning: Removed 11 rows containing missing values (`geom\_smooth()`).

 The LOESS curves for powers of nr.employed (particularly comparing the tenth power to the first or second power) seem to improve as the power increases. This points to adding polynomial complexity terms to the model could be useful.

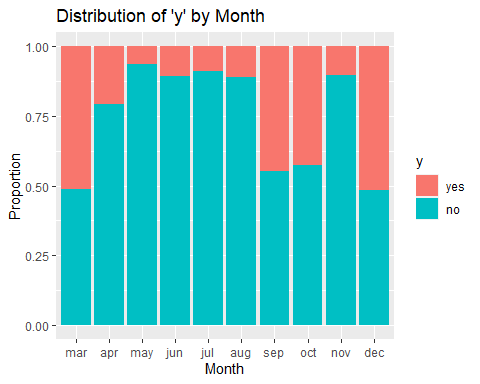
### Percentage Plots

It can be helpful to look at plots that sum up to 100% for Yes and No results for categorical variables. This can show that certain values have a higher or lower percentage of Yes results.

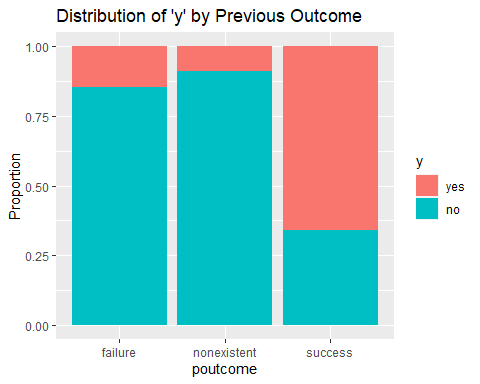
# Percentage plot for education  
ggplot(train\_data, aes(x = education, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Distribution of 'y' by Education") +   
 ylab("Proportion") +   
 xlab("Education Level")



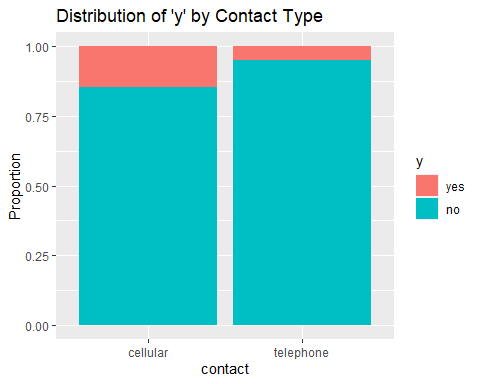
# Percentage plot for month  
ggplot(train\_data, aes(x = month, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Distribution of 'y' by Month") +   
 ylab("Proportion") +   
 xlab("Month")



# Percentage plot for poutcome  
ggplot(train\_data, aes(x = poutcome, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Distribution of 'y' by Previous Outcome") +   
 ylab("Proportion") +   
 xlab("poutcome")



# Percentage plot for contact  
ggplot(train\_data, aes(x = contact, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Distribution of 'y' by Contact Type") +   
 ylab("Proportion") +   
 xlab("contact")

 Education shows very similar Yes/No percentages across the different Education Types, which indicates that it won’t be a useful variable for model building. Month and poutcome have 1 or more values with very high values of yes, which could be useful for model building. Contact seems to have one category with roughly double the number of yes, so it might be slightlty useful in model building.

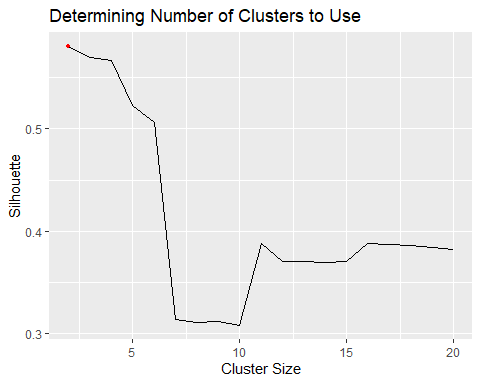
## Clustering

We wanted to see if a clustering analysis of the data would be useful. We only looked at the numeric variables. The purpose of a clustering analysis is to see if splitting the data into clusters allows for additional insight, particularly into classification.

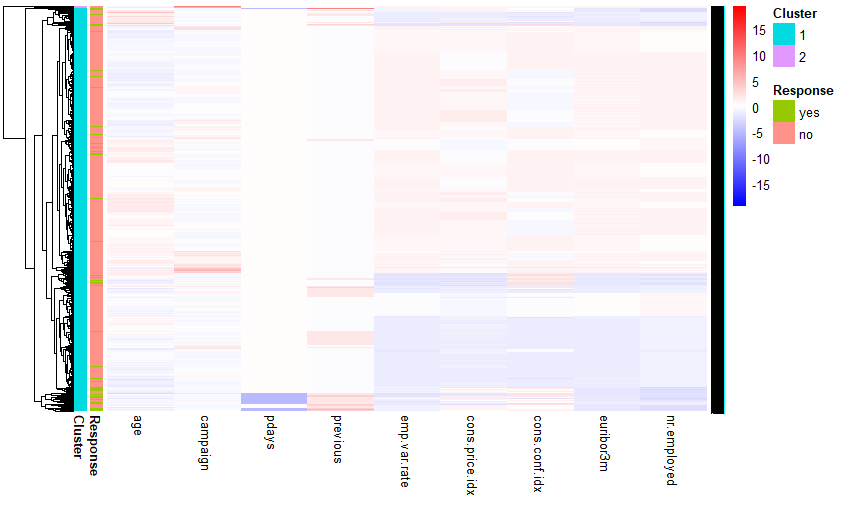
First, we will try to determine the number of clusters to use. The metric we used for comparing cluster sizes is the Silhoutte Statistic. Below is the code I ran, but it has trouble knitting. The heat maps are in the PPT though.

library(RColorBrewer)  
library(pheatmap)  
library(cluster)  
df.numeric <- train\_data[ , sapply(train\_data, is.numeric)]  
center.scale=scale(df.numeric)  
mydist<-dist(center.scale)  
sim.clust<-hclust(mydist,method="complete")  
max\_clusters <- 20  
my.sil<-c()  
for (i in 2:max\_clusters){  
 print(i)  
 sil.result<-silhouette(cutree(sim.clust,i),mydist)  
 my.sil[i-1]<-summary(sil.result)$avg.width  
}

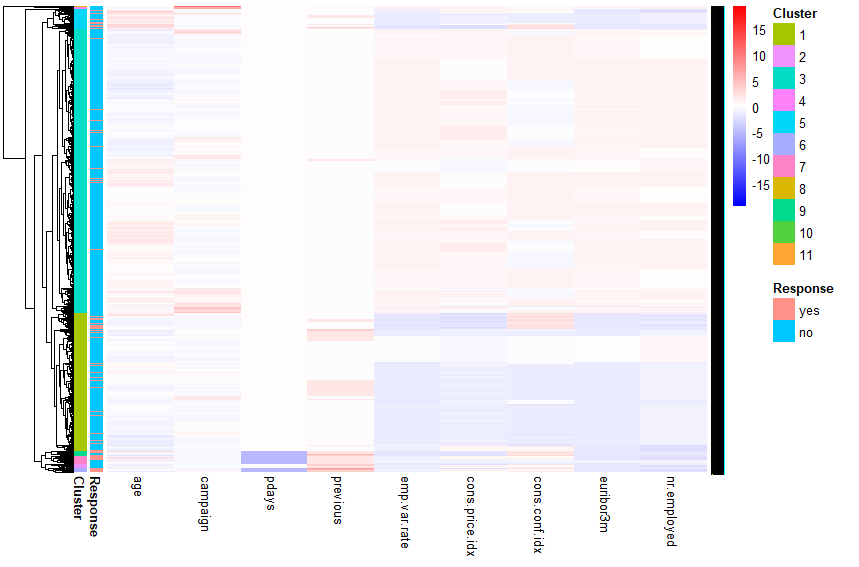
max\_clusters <- 20  
my.sil <- read.csv('https://raw.githubusercontent.com/stedua22/6372-Project-2/main/my\_sil.csv')  
my.sil <- my.sil$x  
ggplot(data = data.frame(x=2:max\_clusters, y=my.sil),aes(x=x,y=y)) + geom\_line() +   
 ylab('Silhouette') + xlab('Cluster Size') + ggtitle('Determining Number of Clusters to Use') +   
 geom\_point(data = data.frame(x = 2, y = my.sil[1]), aes(x = x, y = y), size = 1, color = "red", fill = "red", shape = 21)

 It looks like 2 clusters is the highest, so we will try that.

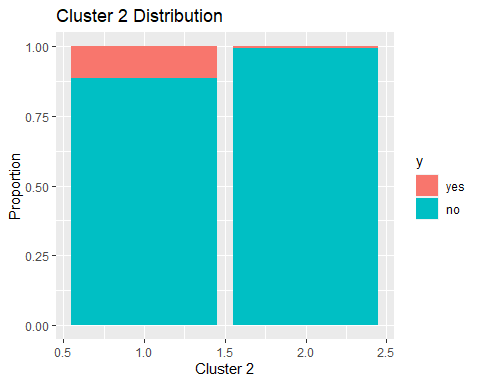
num\_clusters <- 2  
rownames(df.numeric)<-paste("R",1:nrow(df.numeric),sep="")  
annotation\_row<-data.frame(Response=factor(train\_data$y),Cluster=factor(cutree(sim.clust,num\_clusters)))  
rownames(annotation\_row)<-rownames(df.numeric)  
pheatmap(df.numeric,annotation\_row=annotation\_row,cluster\_cols=F,scale="column",fontsize\_row=3,legend=T  
 ,color=colorRampPalette(c("blue","white", "red"), space = "rgb")(100))

 One of the clusters is huge and the other is tiny. Let’s try with 11 clusters.

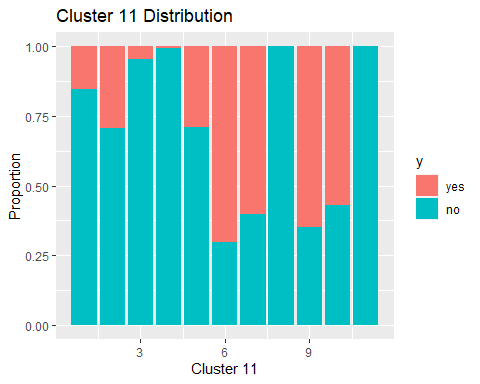
num\_clusters <- 11  
rownames(df.numeric)<-paste("R",1:nrow(df.numeric),sep="")  
annotation\_row<-data.frame(Response=factor(train\_data$y),Cluster=factor(cutree(sim.clust,num\_clusters)))  
rownames(annotation\_row)<-rownames(df.numeric)  
pheatmap(df.numeric,annotation\_row=annotation\_row,cluster\_cols=F,scale="column",fontsize\_row=3,legend=T  
 ,color=colorRampPalette(c("blue","white", "red"), space = "rgb")(100))

 This is a little bit more interesting. Some clusters had a lot of ‘yes’ results. Others seem to be the same distribution as before.

cluster2 <- read.csv('https://raw.githubusercontent.com/stedua22/6372-Project-2/main/cluster2.csv')  
cluster11 <- read.csv('https://raw.githubusercontent.com/stedua22/6372-Project-2/main/cluster11.csv')  
train\_data$cluster2 <- cluster2$x  
train\_data$cluster11 <- cluster11$x  
  
ggplot(train\_data, aes(x = cluster2, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Cluster 2 Distribution") +   
 ylab("Proportion") +   
 xlab("Cluster 2")



ggplot(train\_data, aes(x = cluster11, fill = y)) +   
 geom\_bar(position = "fill") +   
 ggtitle("Cluster 11 Distribution") +   
 ylab("Proportion") +   
 xlab("Cluster 11")

 The 11 cluster example does seem like it does a reasonable job of picking out clusters with

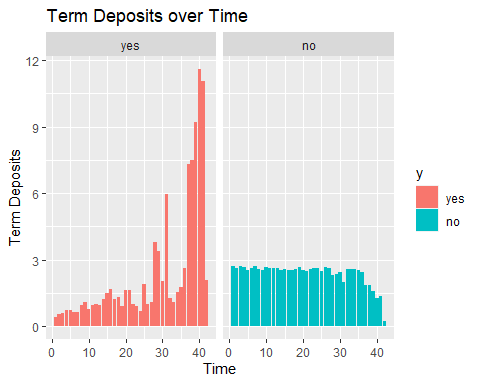
## Variables over time

We want to see if there was any variation in term deposits over time. The file explaining the dataset mentioned that the data was in order of date.

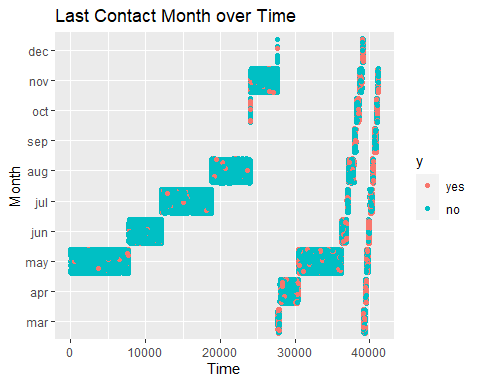
# Sorting data by row number  
train\_data\_sorted <- train\_data  
train\_data\_sorted$num <- as.numeric(rownames(train\_data\_sorted))  
train\_data\_sorted$group <- ceiling(train\_data\_sorted$num / 1000)  
summary <- train\_data\_sorted %>%  
 group\_by(group,y) %>%  
 summarize(count=n())

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

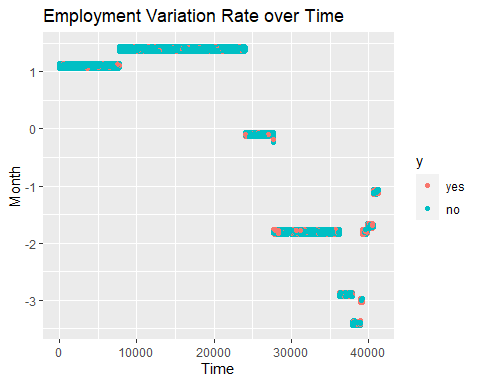
summary$perc <- 0  
summary$perc[summary$y == 'no'] <- summary$count[summary$y == 'no'] / nrow(train\_data\_sorted[train\_data\_sorted$y == 'no',]) \* 100  
summary$perc[summary$y == 'yes'] <- summary$count[summary$y == 'yes'] / nrow(train\_data\_sorted[train\_data\_sorted$y == 'yes',]) \* 100  
  
# Term deposit over time  
summary %>% ggplot(aes(x=group,y=perc,fill=y)) + geom\_bar(stat="identity") + facet\_wrap(~y) +   
 ylab('Term Deposits') + xlab('Time') + ggtitle('Term Deposits over Time')



# Month over time  
train\_data\_sorted %>% ggplot(aes(x=num, y=month, color=y)) + geom\_jitter() +   
 xlab('Time') + ylab('Month') + ggtitle('Last Contact Month over Time')



# Employment Variation Rate over time  
train\_data\_sorted %>% ggplot(aes(x=num, y=emp.var.rate, color=y)) + geom\_jitter() +   
 xlab('Time') + ylab('Month') + ggtitle('Employment Variation Rate over Time')



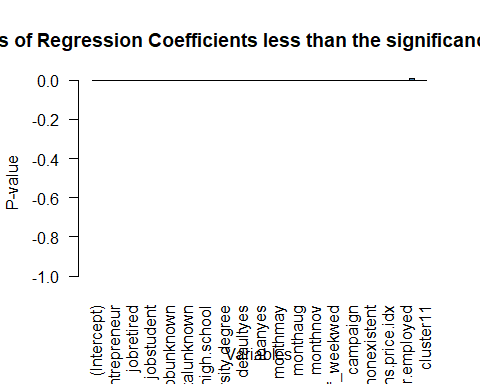
#### LOOKING AT THE PVALUE DISTRIBUTIONS

Looking at how each variable in the model, significantly impacts our response variable

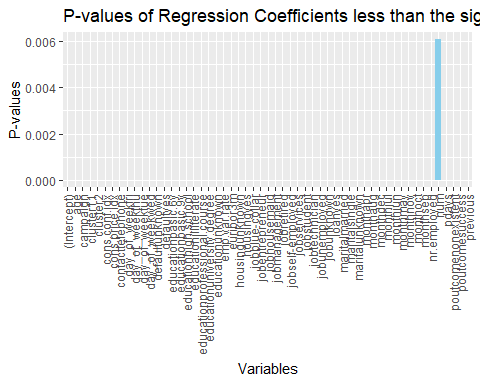
log.model <-glm(y ~ . ,data = train\_data,family="binomial")

## Warning: glm.fit: algorithm did not converge

# Extract variable names  
variable\_names <- rownames(summary(log.model)$coefficients)  
  
# Setting the levels back  
data$y <- relevel(data$y, ref="yes")  
  
# getting the p-values from the model3  
p\_values <- summary(log.model)$coefficients[, 4] # Assuming p-values are in the 4th column of the summary table  
p\_values <- data.frame(p\_values)$p\_values  
  
df <- data.frame(variable\_names, p\_values) #combining the pvalues and variable names into a dataframe  
  
df <- df[!df$p\_value == 0 , ] #removing varaiables with pvalue = 0  
df$p\_values <- log(df$p\_values) \* -1  
  
barplot(df$p\_values,   
 main = "P-values of Regression Coefficients less than the significance level 0.05",   
 xlab = "Variables",   
 ylab = "P-value",  
 names.arg = df$variable\_names,  
 las = 2, # Rotate x-axis labels vertically for better readability  
 col = "steelblue", # Set color of bars  
 ylim = c(exp(0.05) \* -1, max(df$p\_values) \* 1.2) # Set ylim from the significance level to the max p-values  
   
)



library(ggplot2)  
ggplot(df,aes(variable\_names,p\_values, fill = ifelse(p\_values > (exp(0.05) \* -1), "Positive", "Negative"))) + #filtering just the highly significant p-values  
 geom\_bar(stat="identity", fill = "skyblue") +   
 #geom\_text(aes(label = variable\_names), vjust = -0.5) + # Add text labels on top of bars  
 scale\_fill\_manual(values = c("Positive" = "skyblue", "Negative" = "salmon")) +  
 labs(x = "Variables", y = "P-values", title = "P-values of Regression Coefficients less than the significance level 0.05") +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) # Rotate x-axis labels for better readability

 From the plot above, we can see that the top 5 highly significant values with respoct to the response variable y are months(most of them), poutcome, emp.var.rate, contact, cons.price.idx, which are similar to our selected simple logistic model

# PCA models

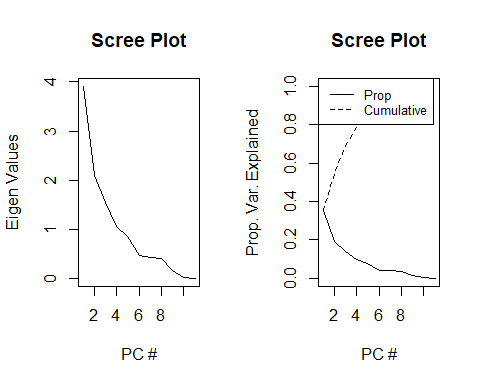
#make sure "success" level is defined as "yes"  
str(train\_data$y)

## Factor w/ 2 levels "yes","no": 1 2 2 2 2 2 1 2 2 2 ...

train\_data$num <- c()  
  
  
#PCA  
df.numericPC <- train\_data[ , sapply(train\_data, is.numeric)]  
pc.result<-prcomp(df.numericPC,scale.=TRUE)  
pc.scores<-pc.result$x  
pc.scores<-data.frame(pc.scores)  
pc.scores$y<-train\_data$y  
  
  
  
#Eignenvector Matrix  
View(pc.result$rotation)  
  
#Scree plot  
eigenvals<-(pc.result$sdev)^2  
eigenvals

## [1] 3.90935320 2.10748012 1.55067031 1.06927372 0.86991295 0.46536151  
## [7] 0.42482392 0.40536507 0.16265162 0.02466925 0.01043832

par(mfrow=c(1,2))  
plot(eigenvals,type="l",main="Scree Plot",ylab="Eigen Values",xlab="PC #")  
plot(eigenvals/sum(eigenvals),type="l",main="Scree Plot",ylab="Prop. Var. Explained",xlab="PC #",ylim=c(0,1))  
cumulative.prop<-cumsum(eigenvals/sum(eigenvals))  
lines(cumulative.prop,lty=2)  
legend("topleft", legend=c("Prop","Cumulative"),  
 lty=1:2, cex=0.8)



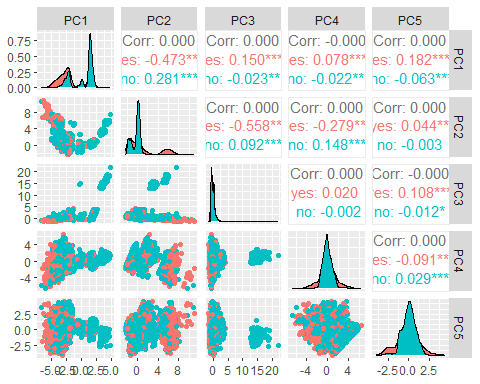
data.frame(PC=1:length(eigenvals),Prop=eigenvals/sum(eigenvals),Cumulative=cumulative.prop)

## PC Prop Cumulative  
## 1 1 0.3553957456 0.3553957  
## 2 2 0.1915891022 0.5469848  
## 3 3 0.1409700280 0.6879549  
## 4 4 0.0972067021 0.7851616  
## 5 5 0.0790829955 0.8642446  
## 6 6 0.0423055922 0.9065502  
## 7 7 0.0386203567 0.9451705  
## 8 8 0.0368513697 0.9820219  
## 9 9 0.0147865105 0.9968084  
## 10 10 0.0022426592 0.9990511  
## 11 11 0.0009489382 1.0000000

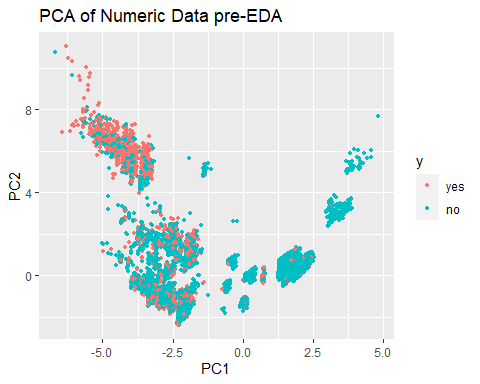
# Calculate the variance explained by each principal component  
var\_explained <- pc.result$sdev^2 / sum(pc.result$sdev^2)  
cum\_var\_explained <- cumsum(var\_explained)  
  
# Find the number of components that explain at least 90% of the variance  
num\_comp\_90 <- which(cum\_var\_explained >= 0.9)[1]  
  
# Print the number of components  
print(num\_comp\_90) #We would need 5 to retain approximately 90%

## [1] 6

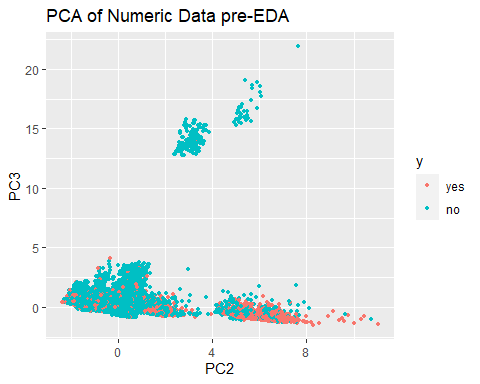
#Plotting PCA variables with the two colors:  
  
  
pc.result<-prcomp(df.numericPC,scale.=TRUE)  
PC <- data.frame(diagnosis = train\_data$y)  
PC$PC1 <- pc.result$x[,1]  
PC$PC2 <- pc.result$x[,2]  
PC$PC3 <- pc.result$x[,3]  
PC$PC4 <- pc.result$x[,4]  
PC$PC5 <- pc.result$x[,5]  
ggpairs(PC[,-1],aes(color=PC[,1]))



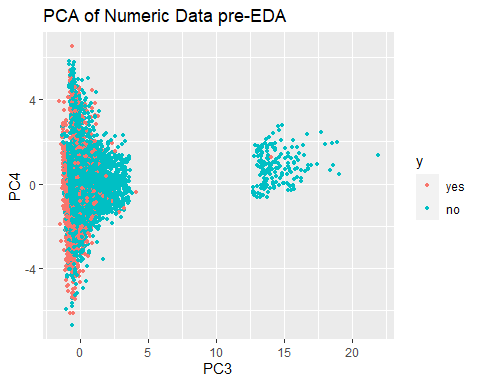
#Use ggplot2 to plot the first few pc's  
ggplot(data = pc.scores, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



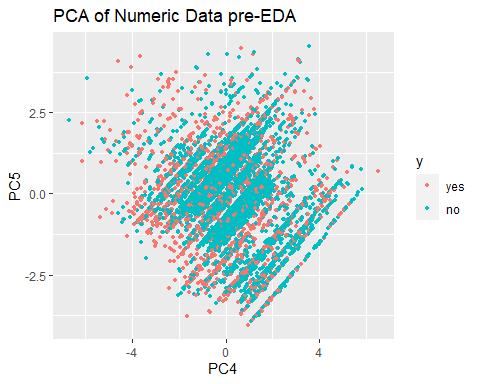
ggplot(data = pc.scores, aes(x = PC2, y = PC3)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



ggplot(data = pc.scores, aes(x = PC3, y = PC4)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



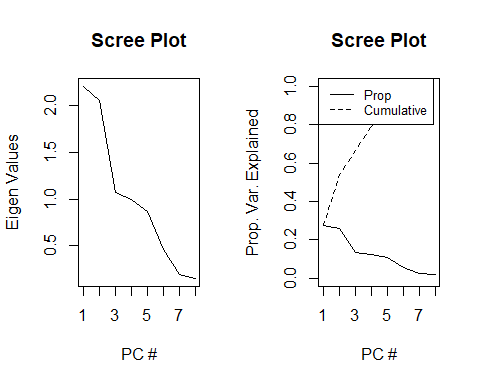
ggplot(data = pc.scores, aes(x = PC4, y = PC5)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



# PCA without campaign, euribor3m,and nr.employed as they are more like factors and not continuous  
  
  
#Performing PCA on predictors  
  
df.numeric2 <- df.numericPC[,-c(2,8,9)]  
pc.result2<-prcomp(df.numeric2,scale.=TRUE)  
pc.scores2<-pc.result2$x  
pc.scores2<-data.frame(pc.scores2)  
pc.scores2$y<-train\_data$y  
#pc.scores2  
  
  
#Eignenvector Matrix  
View(pc.result2$rotation)  
  
#Scree plot  
eigenvals2<-(pc.result2$sdev)^2  
eigenvals2

## [1] 2.2091986 2.0617699 1.0752687 0.9925676 0.8622411 0.4611968 0.1881167  
## [8] 0.1496406

par(mfrow=c(1,2))  
plot(eigenvals2,type="l",main="Scree Plot",ylab="Eigen Values",xlab="PC #")  
plot(eigenvals2/sum(eigenvals2),type="l",main="Scree Plot",ylab="Prop. Var. Explained",xlab="PC #",ylim=c(0,1))  
cumulative.prop<-cumsum(eigenvals2/sum(eigenvals2))  
lines(cumulative.prop,lty=2)  
legend("topleft", legend=c("Prop","Cumulative"),  
 lty=1:2, cex=0.8)



data.frame(PC=1:length(eigenvals2),Prop=eigenvals2/sum(eigenvals2),Cumulative=cumulative.prop)

## PC Prop Cumulative  
## 1 1 0.27614982 0.2761498  
## 2 2 0.25772124 0.5338711  
## 3 3 0.13440859 0.6682796  
## 4 4 0.12407095 0.7923506  
## 5 5 0.10778013 0.9001307  
## 6 6 0.05764961 0.9577803  
## 7 7 0.02351459 0.9812949  
## 8 8 0.01870507 1.0000000

# Calculate the variance explained by each principal component  
var\_explained <- pc.result2$sdev^2 / sum(pc.result2$sdev^2)  
cum\_var\_explained <- cumsum(var\_explained)  
  
# Find the number of components that explain at least 90% of the variance  
num\_comp\_90 <- which(cum\_var\_explained >= 0.9)[1]  
  
# Print the number of components  
print(num\_comp\_90) #We would need 4 to retain approximately 90%

## [1] 5

#Plotting PCA variables with the two colors:  
  
pc.result<-prcomp(df.numeric2[,-c(2,8,9)],scale.=TRUE)  
PC <- data.frame(diagnosis = train\_data$y)  
PC$PC1 <- pc.result2$x[,1]  
PC$PC2 <- pc.result2$x[,2]  
PC$PC3 <- pc.result2$x[,3]  
PC$PC4 <- pc.result2$x[,4]  
ggpairs(PC[,-1],aes(color=PC[,1]))

