pre-processing

35680875 - E Magee

26 November 2020

```r
library(tidyverse) #loading in packages
```

```
## Warning: package 'tidyverse' was built under R version 4.0.3

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## Warning: package 'tidyr' was built under R version 4.0.3

## Warning: package 'readr' was built under R version 4.0.3

## Warning: package 'forcats' was built under R version 4.0.3

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
raw_data = read.csv("../data/Data.csv") #reading in data
```

Looking at the data set the ingredients go up to 476 so assuming there are none missing, the first 477 columns can be omitted for the group data. The first column is an arbitrary number indicating the observation. There are 811 variable in total.

```r
raw_data$Pass = as.numeric(raw_data$Response >= 50) #creating numerical variable for if it passes test

raw_data[is.na(raw_data)] <- 0 #changes missing values to 0

raw_data$LAB <- as.factor(raw_data$LAB) #converting lab to factor

no_ingredients = raw_data[, 478:812] #slicing the data set to get just groups, lab and response

#names(no_ingredients) #verifying that slicing worked as intended, commented out for the sake of the kn
```

Useful to standardise the data since the scales are very different across the data set. Don't want to standardise response, lab or pass though. Standardise columns other than these then add them to the data set.

```
standardised_no_ingredients = data.frame(scale(no_ingredients[,1:332])) #standardising data.

standardised_no_ingredients$LAB = no_ingredients$LAB
standardised_no_ingredients$Response = no_ingredients$Response
standardised_no_ingredients$Pass = no_ingredients$Pass
```

Columns that have lots of 0s may appear weird when standardisation happens. Maybe normalise between 0 and 1 would be more appropriate.

Discuss outliers. Maybe they shouldn't be removed since a particularly high measurement for one property could have a significant impact on the response variable. Also, variables that are mostly 0 value may bring up false outliers.

Do we want to do PCA? There are lots of variables so this might be good for dimensionality reduction.

```
write.csv(standardised_no_ingredients, "../data/standardised_data_460.csv")
```

Wrote to csv, used PCA code from 403 labs to perform a preliminary PCA to check how appropriate it would be.

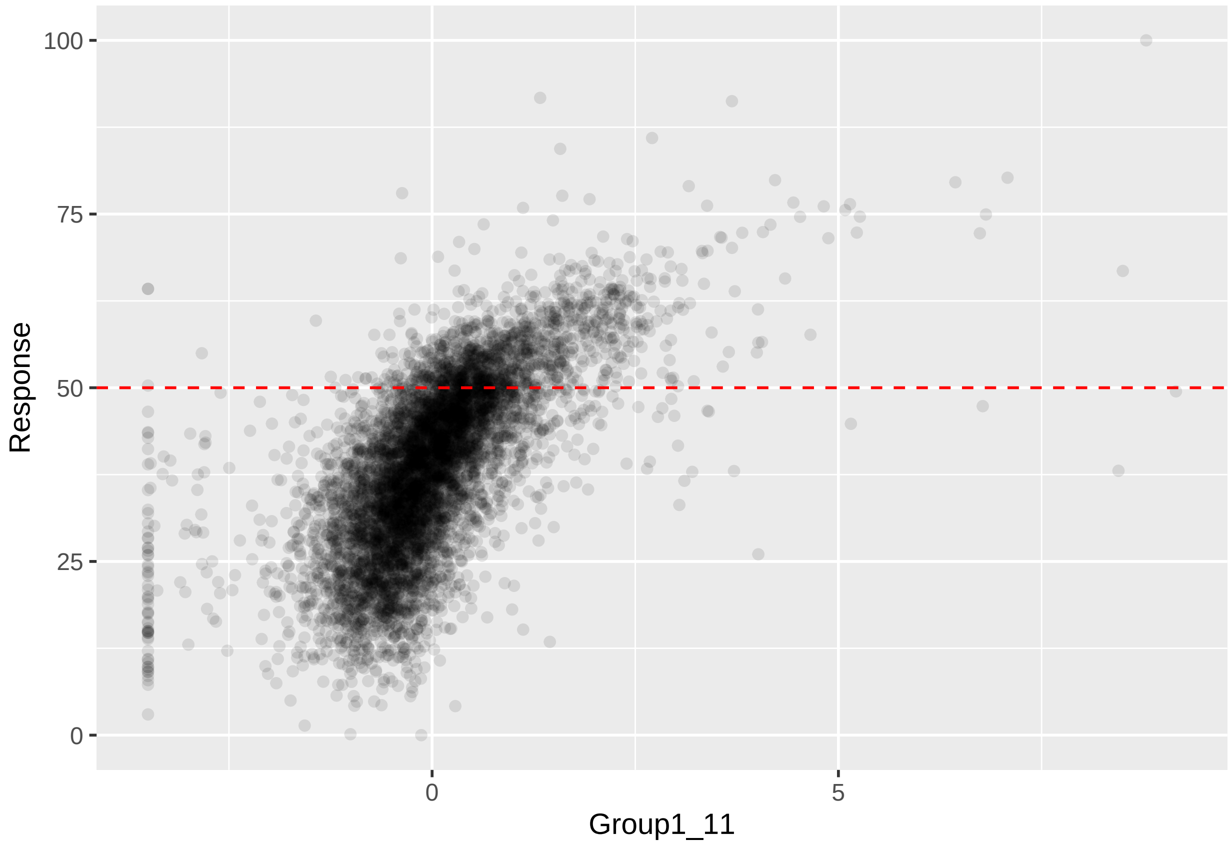
47 Principle Components are required to reach 85% explained variance.

59 are required for 90% and 77 for 95%

PCA does reduce the dimensionality but maybe not worth using for analysis since it loses interpretability and doesn't make the data much simpler to work with.

Brief Exploratory with pre-processed data

```
ggplot(standardised_no_ingredients, aes(x= Group1_11, y = Response))+
  geom_point(alpha = 0.1)+
  geom_hline(yintercept = 50, color = "red", linetype = "dashed")
```



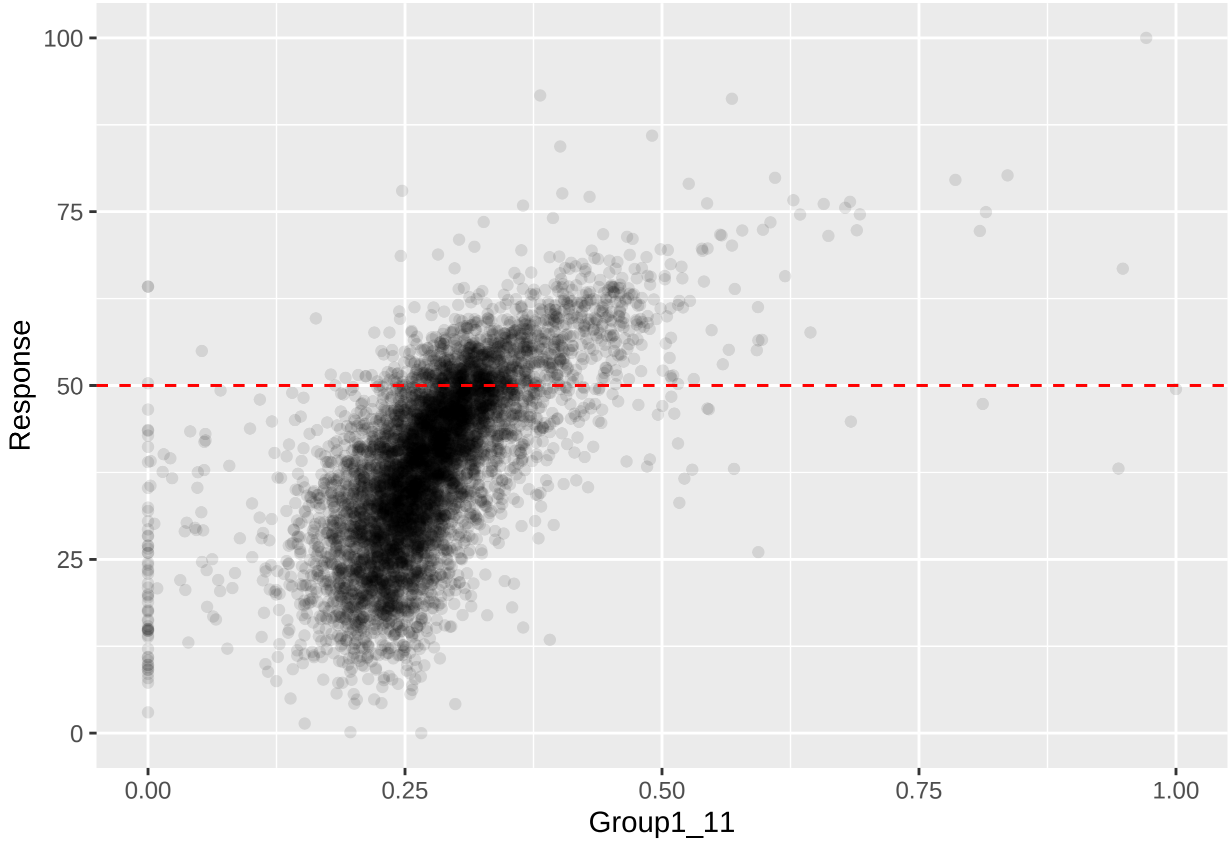
looking at the above exploratory plot it agrees with Alex's finding that group1_11 is strongly correlated with the response. Since the value has a hard lower limit maybe normalisation would be more appropriate.

```
library(BBmisc) #loading in package with a normalisation function
```

```
## Warning: package 'BBmisc' was built under R version 4.0.3

##
## Attaching package: 'BBmisc'

## The following objects are masked from 'package:dplyr':
##
##     coalesce, collapse

## The following object is masked from 'package:base':
##
##     isFALSE
```

```
#normalising the data between 0 and 1
normalised_no_ingredients = normalize(no_ingredients[,1:332], method = "range")

normalised_no_ingredients$LAB = no_ingredients$LAB
normalised_no_ingredients$Response = no_ingredients$Response
normalised_no_ingredients$Pass = no_ingredients$Pass
```

```
ggplot(normalised_no_ingredients, aes(x= Group1_11, y = Response))+
  geom_point(alpha = 0.1)+
  geom_hline(yintercept = 50, color = "red", linetype = "dashed")
```



Can see from this plot that the distribution doesn't change at all but the scale on x has changed. Not sure which one is more appropriate.