ACSC/STAT 3740, Predictive Analytics

WINTER 2025

Toby Kenney

Practice Final Examination

Model Solutions

This Sample examination has more questions than the actual final, in order to cover a wider range of questions. Estimated times are provided after each question to help your preparation.

[Note: All data on this exam are simulated.]

1. *A data scientist has written the code in the file* `PFQ1_code.R` *to analyse the data in the file* `PFQ1.txt`*. After 10 minutes, the code is still running. What is the problem?*

   The problem is that the code takes longer to run than expected. It is easy to see that the code should be linear in the number of predictions made. Running 5 predictions took about 13.5 seconds on my system, which means that runing 432 predictions should take about 20 minutes. If it were left to run for longer, it would finish.

   Depending on the situation, just waiting for the code to finish may be the best solution. If this code will only be used for this analysis, which will require a single report, then waiting a few minutes would be the easiest solution. When archiving the code, the analyst should add a comment about the running time, so that future users know to expect to have to wait if rerunning the analysis.

   If the code will be used to regularly update the analysis with new data, then the code should be changed to run faster. The list of predictions is created by concatenation, which is inefficient and has higher risk of bugs. Changing that to producing an empty list in advance and filling the entries is an obvious way to speed up the code. However, this is not the most time-consuming part of the code, so changing it will have negligible effect on the running time. The time-consuming part is fitting the leave-one-out cross-validation ridge regression. There is an algebraic trick to do this more efficiently. Alternatively, using 10-fold cross-validation would speed up the code at a slight cost to the accuracy of the prediction results.

2. *An intern has written the code in the file* `PFQ2_code.R` *to analyse the data in the file* `Sales_records_Mar.txt`*. The company wants to build the code into a standard analysis pipeline for processing monthly data updates. Modify the code to make it easier to maintain.*

   To make this script able to be run every month, we need to add code to find the appropriate name of the last month. The following code uses the current date to do this. It subtracts the day of the month from the current date to get a day in the previous month, then extracts the month as a string. This assumes that the analysis is being run for the previous month rather than an earlier month. An option will need to be added to override this.

```
get.last.month<-function(){
    ##returns the name of the previous month based on the current month.
    dat<-as.Date(date(),format="%a %b %d %H:%M:%S %Y")
    day<-as.numeric(as.character(dat,format="%d"))
    ## find current day of the month.
    ## if we subtract this many days, it will be the previous month.
    return(as.character(dat-day,format="%b"))
}
```

Alternatively, we could insist that the user input the file name every time the script is run.

Next we want to tidy up the code for inputting the monthly data. We produce a function that by default loads the data from the files for the previous month, but can be told to use a different month or a completely different file name.

```
read.monthly.data<-function(filename=NULL,
                            exchange.rates.file=NULL,
                            month=get.last.month()){
    ## By default the filenames for data and exchange rates are based
    ## on the previous month. However, this can be overwritten by the
    ## parameters.

    if(filename==NULL){
        ##Default Filename
        filename=paste("Sales_records_",month,".txt",sep="")
    }
    if(exchange.rates.file==NULL){
        ##Default Filename
        exchange.rates.file=paste("Exchange_rates",month,sep="_")
    }

    sales<-read.table(filename,stringsAsFactors=TRUE)
    exchange.rates<-read.table(exchange.rates.file,row.names=1)
    return(list("sales"=sales,"exchange.rates"=exchange.rates))
}
```

This has successfully converted the first and third lines of the script into general functions which would work in future months. Next we need to write code for the data cleaning. The existing code is extremely messy, and we can simplify the code a lot. The existing code is limited to countries currently in the data, but it would be better to make it general to deal with whatever countries are there, in case new countries are added. The existing code also mixes the cleaning and producing reports. The existing code also uses different methods for each country. While these all work, it makes maintaining the code very difficult. We should replace it with a standardised approach that works for a general currency.

```
get.hour<-function(times){
    ## Function to convert times in format HH:MM to the hour stored as
    ## a numeric vector.
    return(as.numeric(
        unlist(
            lapply(times,
                   function(x){
                       ## This function converts a time in HH:MM format
                       ## to the hours part. May not work in all locales.
                       unlist(strsplit(x,split=":"))[1]}
                   )
        )
    ))
}


clean.data<-function(sales,exchange.rates){

    ## transaction amount is stored as a string containing both
    ## numeric amount and currency. We need to convert it into two
    ## columns, one for numeric amount and one for currency.
    ## we also need to convert using the exchange rates.
    ## Finally, we need to convert the time of day into an hour variable.

    sales$transaction.amount<-as.character(sales$transaction.amount)


    sales$currency<-unlist(lapply(
        strsplit(sales$transaction.amount,
                 split="[0-9]*[.][0-9]*"),
        function(x){rev(x)[1]}))

    ## This works by splitting the field on the numeric value, resulting in
    ## an empty string before the number and a currency string after.
    ## Then the applied function extracts the last element from each list.

    sales$currency.amount<-as.numeric(
        unlist(lapply(
            strsplit(sales$transaction.amount,sales$currency),
            function(x){x[1]}
        ))
    )
    ## Now that we have extracted the currency, we use it to split the
    ## string, producing a numeric amount, and possibly an empty
    ## string after the currency.

    ## Convert using exchange rate from table.
    sales$converted.amount<-sales$currency3amount*exchange.rates[sales$currency]
    ## Convert time of day to the corresponding hour.
    sales$time.of.day<-get.hour(sales$time)
    return(sales)
}
```

Finally, we write code to produce the summary reports. The `dplyr` package has some tools to make this easier. The existing code has a bug which might cause problems — the code to process results by time selects the data for each hour then loops over the records at that hour using `for(i in 1:m)`. If `m` happens to be zero for any hour, this will give an error. Using the tools from the `dplyr` package also fixes this.

```
library(dplyr)


create.summary.tables<-function(sales.data,month=get.last.month()){
    ##First make reports breaking down total sales by country and department.
    country.totals<-sales.data%>%
        group_by(country,department)%>%
        summarise(total_sales=sum(converted.amount))

    write.table(country.totals,paste("Country_Department_Totals_",
                                     month,".txt",sep=""))

    ##Next break down total sales by time
    time.totals<-sales.data%>%
        group_by(time.of.day)%>%
        summarise(total_sales=sum(converted.amount))

    write.table(time.totals,paste("Sales_by_hour",month,".txt",sep=""))
}
```

Finally, the script will just run these functions to process the data.

```
dat<-read.monthly.data()
## We can set the filename if not using usual filenames

sales.data<-clean.data(dat$sales,dat$exchange.rates)
create.summary.tables(sales.data)
## We can set filenames for these reports if not using the standard ones.
```

3. *A scientist has written the code in `PFQ3_code.R` to analyse the data in the file `PFQ3.txt`. The results do not match the scientist's expectations. Determine whether there is a bug in the code, and what it is.*

   A good clue to the problem is that running the code produces a warning "`longer object is not length is not a multiple of shorter object length`". This is saying that we are trying to combine vectors of different lengths. The first warning is at the line "`data$mass[condition]/data$corrected.volume[Condition]`".

   This means that the vectors `data$mass[condition]` and `data$corrected.volume[Condition]` have different lengths. It might be easy to notice the problem here that the second `Condition` is incorrectly capitalised, and so actually refers to a different variable.

If we do not notice this problem at this stage, we might try to compare the estimated and theoretical values.

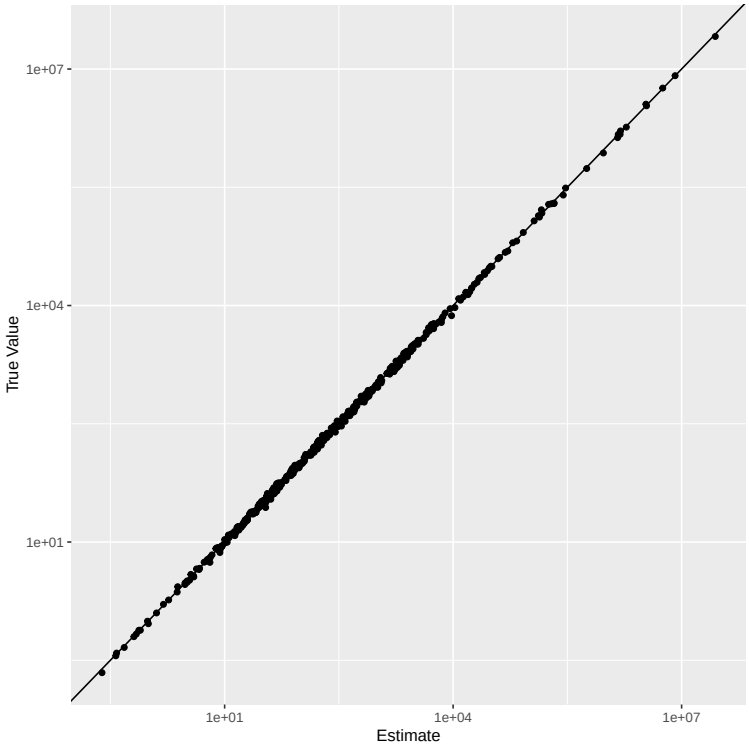
```
library(ggplot2)

### Compare estimated and theoretical values.

ggplot(data.frame(est=theoretical.values,
                  true=corrosion.data$corrosion[
                                        corrosion.data$humidity>55&
                                        corrosion.data$temperature<23]),
       mapping=aes(x=est,y=true))+
    geom_point()+
    geom_abline(mapping=aes(slope=1,intercept=0))
```

This gives an error that the vectors have different lengths. In particular `theoretical.values` has length 723. The problem appears to come from the `calculate.formula` function. We therefore try to run the function line-by-line.

```
data<-corrosion.data
min_humid<-55
max_temp<-23

condition<-data$humidity>min_humid&data$temperature<max_temp
svr<-data$surface.area[condition]/data$volume[condition]
density<-data$mass[condition]/data$corrected.volume[Condition]
surface.water<-data$surface.area[condition]*data$humidity[condition]

theoretical.values<-surface.water*exp(svr+(log(density)-1.5094)^2/4)
```

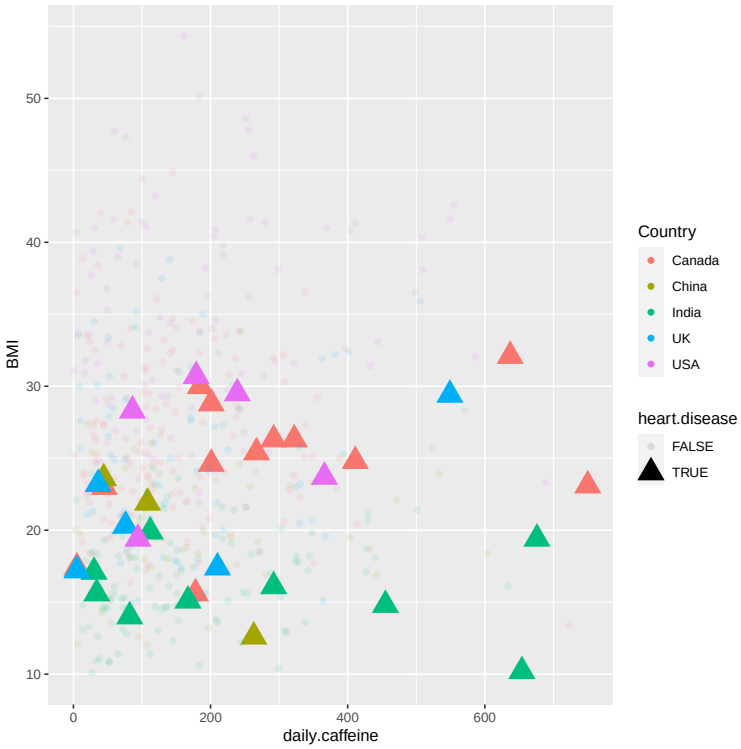
We can now examine all the variables created. We see that `svr` and `surface.water` both have length 437, while `density` has length 723. When we check the length of the vectors used to compute `density`, we get that `data$mass[condition]` has length 437, while `data$corrected.volume[Condition]` has length 723, clearly indicating that this is the incorrect vector. At this point, it should be clear that the incorrect capitalisation has led to this mistake, which is then easily fixed.

After fixing this mistake, the MSE is still quite large, but when we plot the estimated versus true values on a log-scale, we see a good correlation.



This indicates that the error is only because the estimates are on a log-scale, and the relative error is small, but this can still lead to a large absolute error in the theoretical values, and thus a large MSE.

4. *The file* `PFQ4.txt` *contains data from an experiment about the relation between caffeine and heart disease. The data are not formatted in a very convenient way. Read the data into* `R` *and reformat into a more convenient way, and use it to create the following plot.*



Make a list of all corrections made to the data.

First we read the tables into R.

```
PFQ4.Canada<-read.table("PFQ4.txt",
                        skip=2, ### 1 lines before table
                        header=1 ### first row is column headers
                       ,nrows=153, ### read 153 rows in this table
                        stringsAsFactors=TRUE) ### Not crucial here,
PFQ4.China<-read.table("PFQ4.txt",skip=159,header=1,nrows=52,
                       stringsAsFactors=TRUE)
PFQ4.India<-read.table("PFQ4.txt",skip=215,header=1,nrows=116,
                       stringsAsFactors=TRUE)
PFQ4.UK<-read.table("PFQ4.txt",skip=335,header=1,nrows=76,
                    stringsAsFactors=TRUE)
PFQ4.USA<-read.table("PFQ4.txt",skip=415,header=1, ### read to end of
                     stringsAsFactors=TRUE) ### file - no need to set nrows

### Check we read the right number of rows
### This will also show the headers, allowing us to check that
### those have been read correctly.

PFQ4.Canada[153,]
PFQ4.China[52,]
PFQ4.India[c(1,116),]
PFQ4.UK[76,]
PFQ4.USA[99,]
```

Next we combine them into a single table. The separate tables are rows of a larger table.

```
### We need to combine the separate tables into a single table, and we
###  need to create a "Country" variable.

PFQ4.Canada<-data.frame(Country="Canada",PFQ4.Canada)
PFQ4.China<-data.frame(Country="China",PFQ4.China)
PFQ4.India<-data.frame(Country="India",PFQ4.India)
PFQ4.UK<-data.frame(Country="UK",PFQ4.UK)
PFQ4.USA<-data.frame(Country="USA",PFQ4.USA)

PFQ4<-rbind(PFQ4.Canada,PFQ4.China,PFQ4.India,PFQ4.UK,PFQ4.USA)
```

Now we check for and fix formatting errors.

```
### Now we check the data for any additional problems.

summary(PFQ4)

PFQ4$Country<-factor(PFQ4$Country)

summary(PFQ4)

### There are several data entry problems for the heart.disease variable.

PFQ4$heart.disease[ PFQ4$heart.disease=="No"]<-"FALSE"
PFQ4$heart.disease[ PFQ4$heart.disease=="F"]<-"FALSE"
PFQ4$heart.disease[ PFQ4$heart.disease=="false"]<-"FALSE"
PFQ4$heart.disease[ PFQ4$heart.disease=="Yes"]<-"TRUE"
PFQ4$heart.disease[ PFQ4$heart.disease=="T"]<-"TRUE"

summary(PFQ4)

### Now only two non-zero levels for heart.disease.

PFQ4$heart.disease<-PFQ4$heart.disease=="TRUE"

### Only actually needed to fix the entries that correspond to TRUE
### for this to work.

summary(PFQ4)

### Looks good.
```

Finally, we plot the graph.

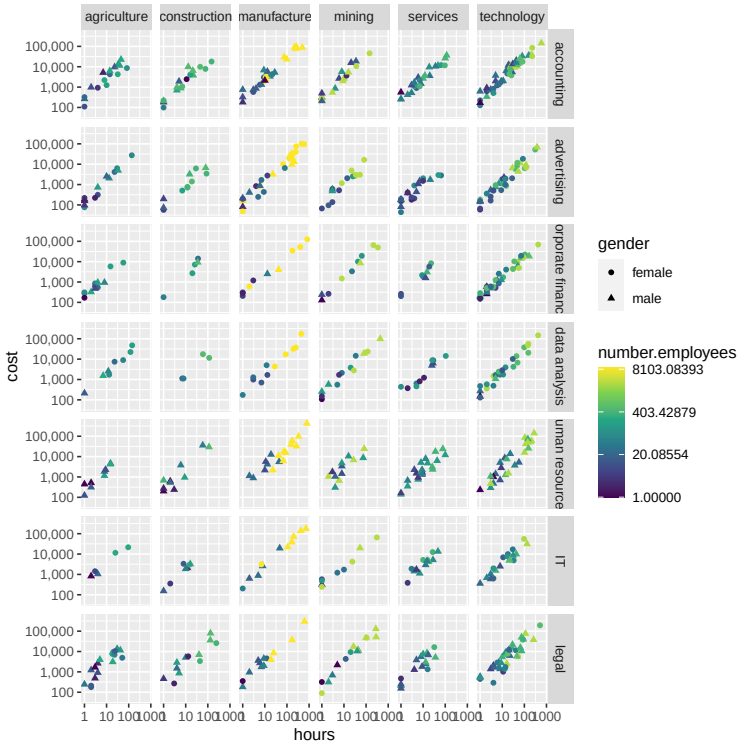
```
library(ggplot2)
ggplot(PFQ4,
       mapping=aes(x=daily.caffeine,
                   y=BMI,
                   shape=heart.disease,
                   size=heart.disease,
                   alpha=heart.disease,
                   colour=Country))+geom_point()
```

We made the following changes to the data:

- Add a "Country" variable.
- Combine the tables into a single data frame.

- Change the values "No", "F" and "false" to `FALSE`, and the values "T" and "Yes" to `TRUE` for the `heart.disease` variable.

5. *The file* `PFQ5.txt` *contains data from a consultancy company about customers and services. The data are not formatted in a very convenient way. Read the data into* `R` *and reformat into a more convenient way, and use it to create the following plot.*



Make a list of all corrections made to the data.

There are three relational tables included in the file, so we need to read and process each table individually, then combine them. We start by processing the employee table. There are a few corrections to the factor levels.

```
PFQ5.employees<-read.table("PFQ5.txt",skip=2,nrow=47,
                           header=1,stringsAsFactors=TRUE)
summary(PFQ5.employees)

### gender and expertise have multiple ways of coding the same values
library(forcats)

PFQ5.employees$gender<-fct_recode(PFQ5.employees$gender,
                                  "male"="male",
                                  "female"="female",
                                  "female"="F")

table(PFQ5.employees$expertise)
PFQ5.employees$expertise<-fct_recode(PFQ5.employees$expertise,
                                     "accounting"="accounting",
                                     "advertising"="advertising",
                                     "corporate finance"= "corporate finance",
                                     "data analysis"="data analysis",
                                     "human resources"="HR",
                                     "human resources"="human resources",
                                     "IT"="IT",
                                     "legal"="law",
                                     "legal"="legal")
summary(PFQ5.employees)
### Now the factor levels are different.
```

Next we process the customer table. We merge a few levels for the "industry" variable.

```
### Next the Customers table
PFQ5.customers<-read.table("PFQ5.txt",skip=53,nrow=180,header=1,
                           stringsAsFactors=TRUE)
summary(PFQ5.customers)
table(PFQ5.customers$industry)

### Some of these seem equivalent and should be merged.

PFQ5.customers$industry<-fct_recode(PFQ5.customers$industry,
                                    "agriculture"="agriculture",
                                    "agriculture"="farming",
                                    "construction"= "construction",
                                    "manufacture"="manufacture",
                                    "mining"="mining",
                                    "services"="services",
                                    "technology"="technology",
                                    "technology"="tech")
table(PFQ5.customers$industry)

### Now seems OK.
```

Then we process the contracts table. The difficulty here is that the cost is listed in currency, mostly dollars, but several entries are in euros, and need to be converted.

```
PFQ5.contracts<-read.table("PFQ5.txt",skip=238,header=1)
summary(PFQ5.contracts)

### cost is formated as character because the table shows currency

PFQ5.contracts$cost
gsub("[$,]","",PFQ5.contracts$cost)
PFQ5.costs<-as.numeric(gsub("[$,]","",PFQ5.contracts$cost))

### This fixes the amounts in dollars, but some are in euros.
PFQ5.contracts$cost[is.na(PFQ5.costs)]

### Use exchange rate given in file.
PFQ5.costs[is.na(PFQ5.costs)]<-
    round(1.432406*as.numeric(gsub("[€,]","",
                                   PFQ5.contracts$cost[is.na(PFQ5.costs)])))
### round values to whole numbers.

PFQ5.costs[16:20]
### Check the values seem correct

### put corrected values in original data frame.
PFQ5.contracts$cost<-PFQ5.costs
summary(PFQ5.contracts)
### Looks OK now.
```

We then merge the tables using the `left_join` function from `tidyr`.

```
library(dplyr)
library(tidyr)

### dplyr package allows us to chain joins.
PFQ5<-PFQ5.contracts%>%
    left_join(PFQ5.employees,by=c("employee"="ID"))%>%
    left_join(PFQ5.customers,by=c("customer"="ID"))

summary(PFQ5)
### Seems OK.
```

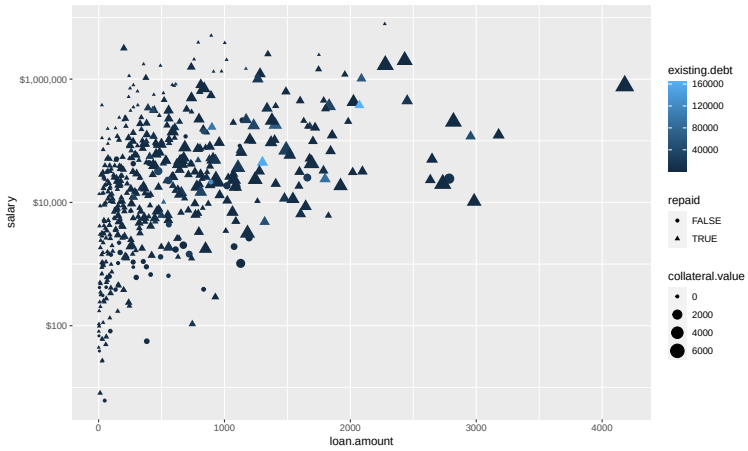
Finally, we can make the plot.

```
ggplot(PFQ5,
       mapping=aes(x=hours,
                   y=cost,
                   colour=number.employees,
                   shape=gender))+
    facet_grid(expertise~industry)+
    geom_point()+
    scale_x_log10()+
    scale_y_log10(labels=scales::comma)+
    scale_colour_viridis_c(trans="log")
```

We have made the following changes to the data:

- In the employee table, "gender" has some values "F" for "female".
- In the employees table, I have merged "HR" into "human resources", and "law" into "legal".
- In the customers table, I have merged "farming" into "agriculture", and "tech" into "technology".
- In the contracts table, I have converted the currency amounts to numeric.
- In the contracts table, a few currency amounts were in euros, and needed to be converted to dollars.
- I have joined the three tables to get full information for each contract.

6. *Use* `ggplot` *to produce the following plot from the data in file* `PFQ6.txt`.



The plot was produced using the code:

```
ggplot(Loan.data,
       mapping=aes(x=loan.amount,
                   y=salary,
                   size=collateral.value,
                   colour=existing.debt,
                   shape=repaid))+
    geom_point()+
    scale_y_log10(labels=scales::dollar)
```

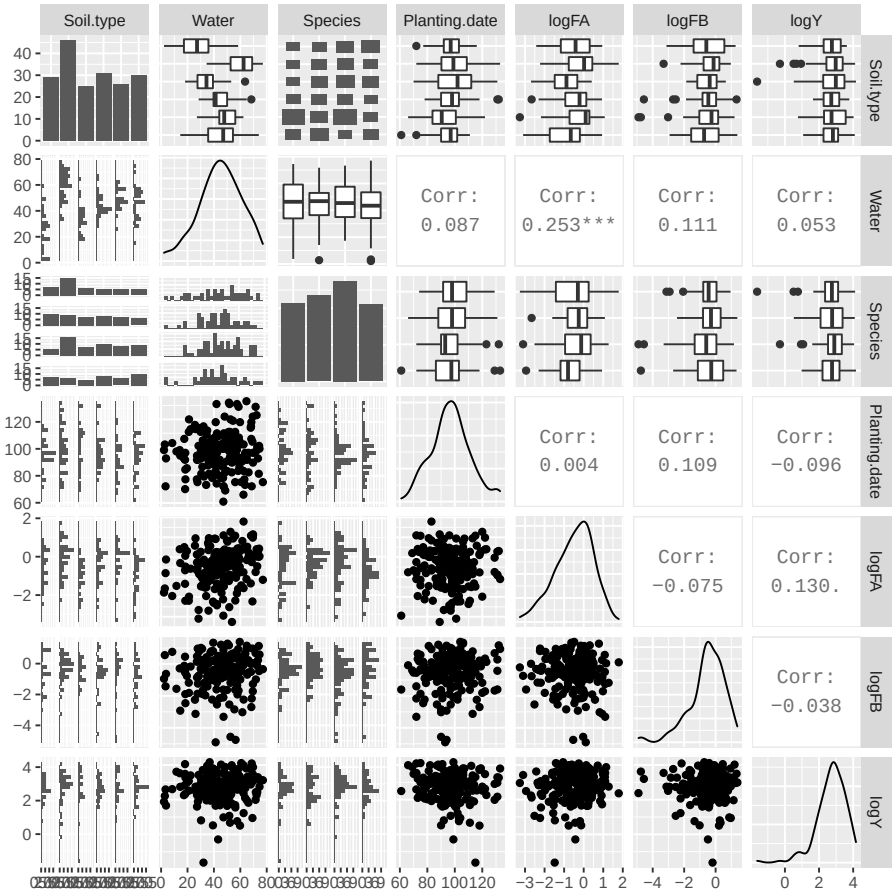
For some reason, rerunning this produces the legends in a different order. To change the order, we can add the code `+guides(size=guide_legend(order=1),shape=guide_legend(order=3),colour=guide_colourbar(order=2))`

7. *The file* `PFQ7.txt` *contains the following data from an experiment to determine the effect of fertilisers on growth of bean crops:*

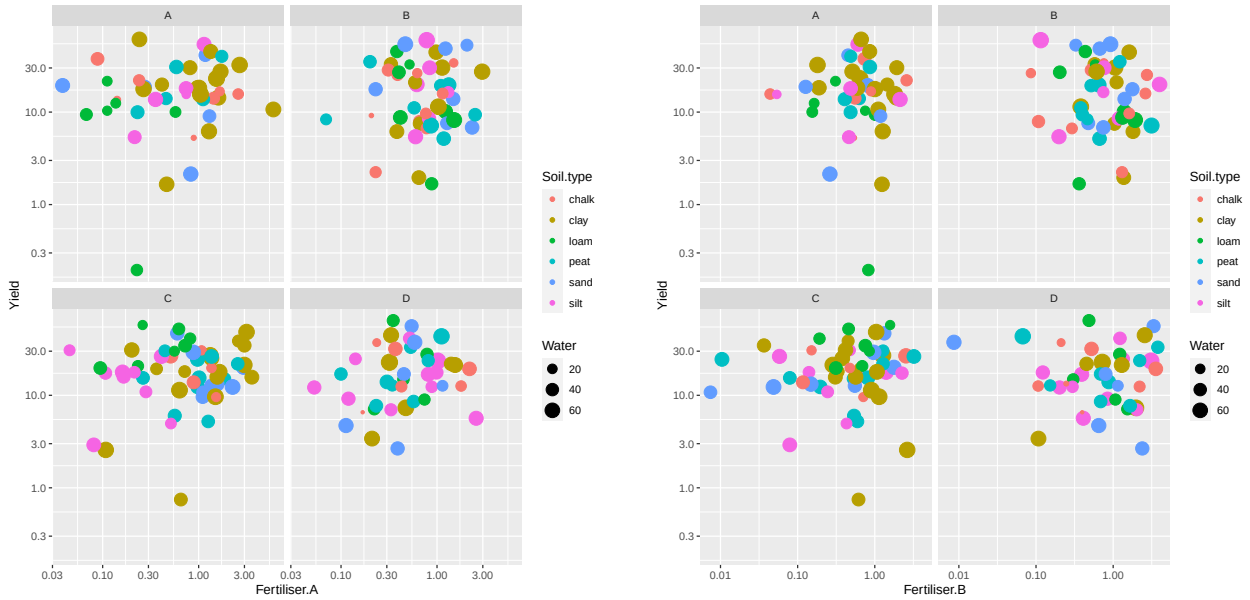
Variable	*Meaning*
Soil.type	*The type of soil*
Water	*The amount of water (l/hectare) provided to the crop*
Species	*The species of bean*
Planting.date	*The date on which the crop was planted*
Fertiliser.A	*The quantity of fertiliser A (l/hectare) provided to the crop*
Fertiliser.B	*The quantity of fertiliser B (l/hectare) provided to the crop*
Yield	*The quantity of crop harvested (kg/hectare)*

Construct a plot or plots to show this data for the purpose of data exploration.

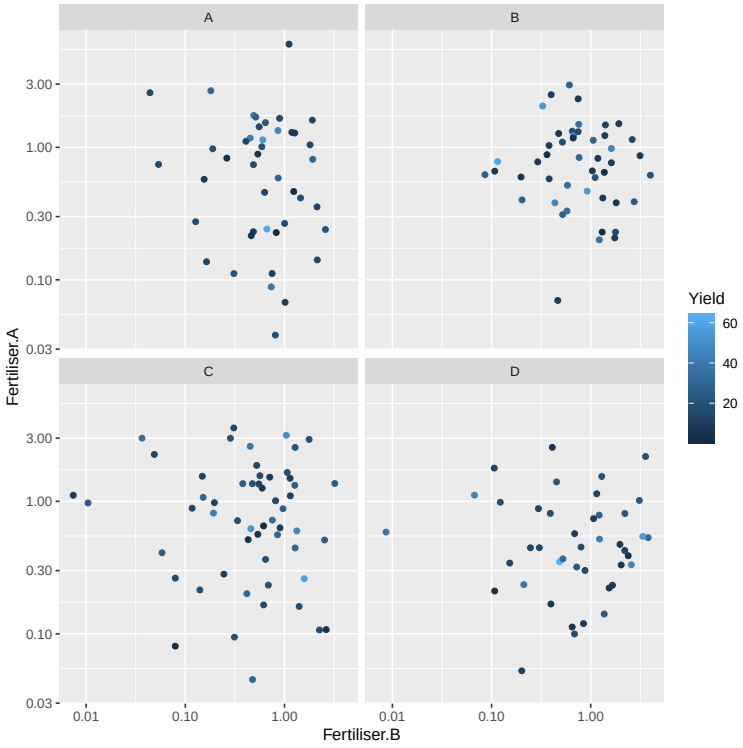
There are a number of plots we might make. It is natural to start with pairwise scatterplots. We have log-transformed several variables.



We might also plot yield against Fertiliser A (using log scales), or against Fertiliser B.



Alternatively, we could plot the quantities of the two fertilisers and use colour (or size or alpha) to indicate the yield.



These plots were produced using the code

```
### Read the data
Crop.data<-read.table("PFQ7.txt")
library(ggplot2)
library(GGally)
library(dplyr)

### pairwise scatter plots (not shown in model solution)
ggpairs(Crop.data)

### log transformations of some variables are appropriate
ggpairs(Crop.data%>%
        mutate(logFA=log(Fertiliser.A),
               logFB=log(Fertiliser.B),
               logY=log(Yield))%>%
        select(-c("Fertiliser.A","Fertiliser.B","Yield"))) # remove original variables.

### Separate plots for each species seem appropriate
ggplot(Crop.data,mapping=aes(y=Yield,  #response variable on y-axis
                             x=Fertiliser.A,
                             size=Water,
                             colour=Soil.type))+
    geom_point()+
    facet_wrap(Species~.)+
    scale_x_log10()+  #log scale is appropriate.
    scale_y_log10()  #log scale is appropriate.

### similar plot for Fertiliser B
ggplot(Crop.data,mapping=aes(y=Yield,
                             x=Fertiliser.B,
                             size=Water,
                             colour=Soil.type))+
    geom_point()+
    facet_wrap(Species~.)+
    scale_x_log10()+scale_y_log10()

#### We could also use facet_grid(Species~Soil.type), but then each
#### subplot has too few points.

### alternatively plot quantities of fertiliser and use colour to show yield.
ggplot(Crop.data,mapping=aes(y=Fertiliser.A,
                             x=Fertiliser.B,
                             colour=Yield))+
    geom_point()+
    scale_x_log10()+scale_y_log10()+
    facet_wrap(Species~.)

### There are a variety of other plots we could make.
```



```
ggplot(Crop.data,mapping=aes(x=Fertiliser.A,
                             y=Fertiliser.B,
                             colour=Water,
                             alpha=Yield))+
    geom_point()+
    scale_x_log10()+scale_y_log10()
```

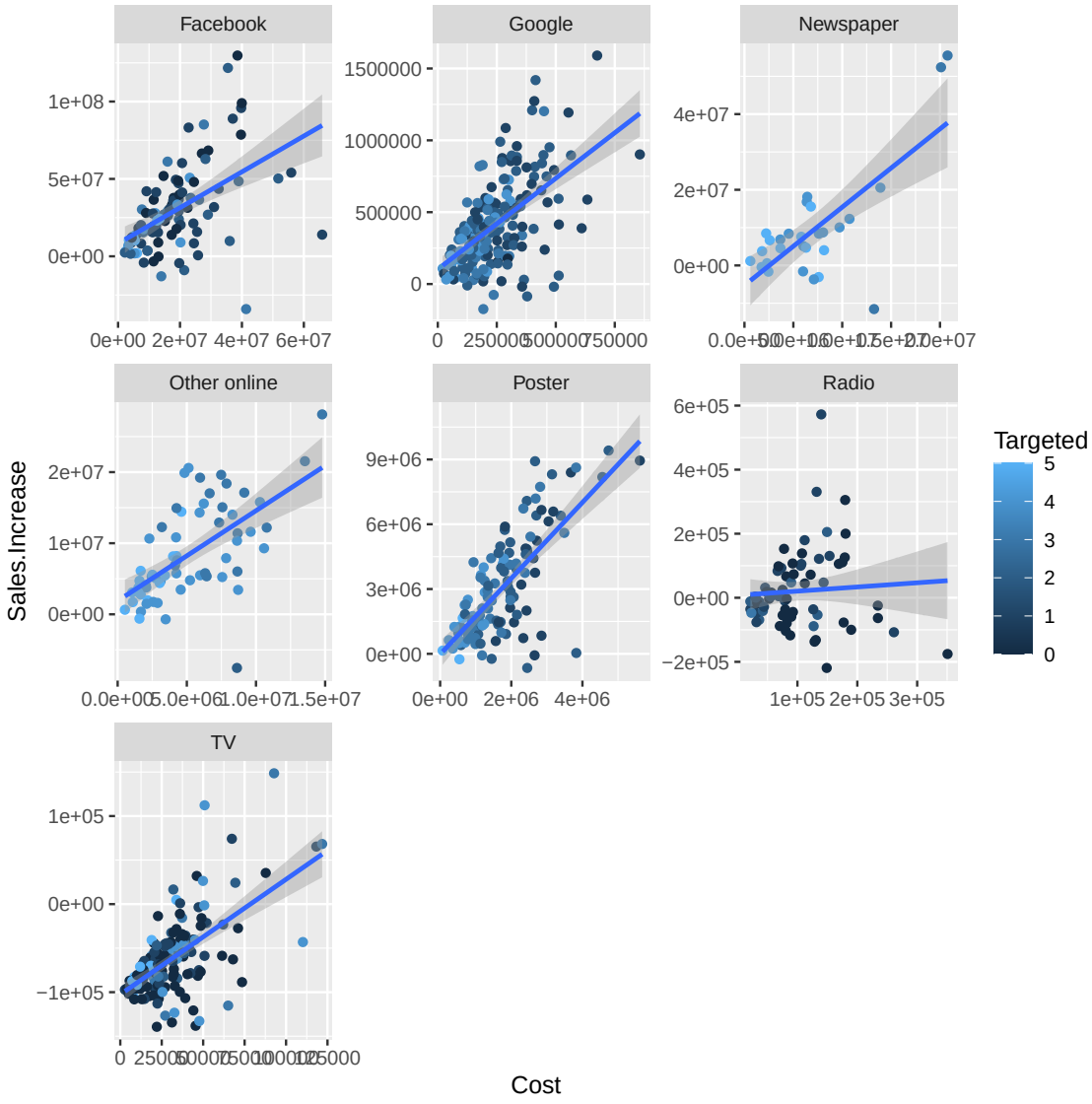
8. *The file* `PFQ8.txt` *contains the following data on advertising campaigns from a company's marketing department.*

Variable	*Meaning*
Type	*The medium of the advertising campaign*
Duration	*The length of time the campaign was running*
Targeted	*Extent to which the campaign was targetted on a scale 1–5*
Cost	*The total amount spent on the advertising campaign.*
Target.audience	*The number of individuals expected to see the advert*
Sales.Increase	*The increase in total sales in the two-month period after the campaign, compared with the two-month period before.*

Perform data exploration on this data set, and summarise (with tables and plots to support where appropriate) your initial conclusions about data issues and appropriate models. You should take into account any concerns with data collection and processing.

We first look at summary statistics and pairwise scatter plots, coloured by type. We have log-transformed many of the variables. Sales increase is heavy-tailed and skewed, but also has negative values, so we could not log-transform it.

We see that the log-transformed variables have different linear relationships with high correlation for each type. Since the types show such different patterns, we use a facet_wrap to make a separate plot for each type.



From this plot, we see that a linear relation between cost and sales increase seems reasonable. There is clear heteroskedasticity, with higher cost corresponding to larger variance in sales increase. It is hard to see a relation between targetted and sales increase, partly because the level of targetting varies a lot between different types of advertising campaign.

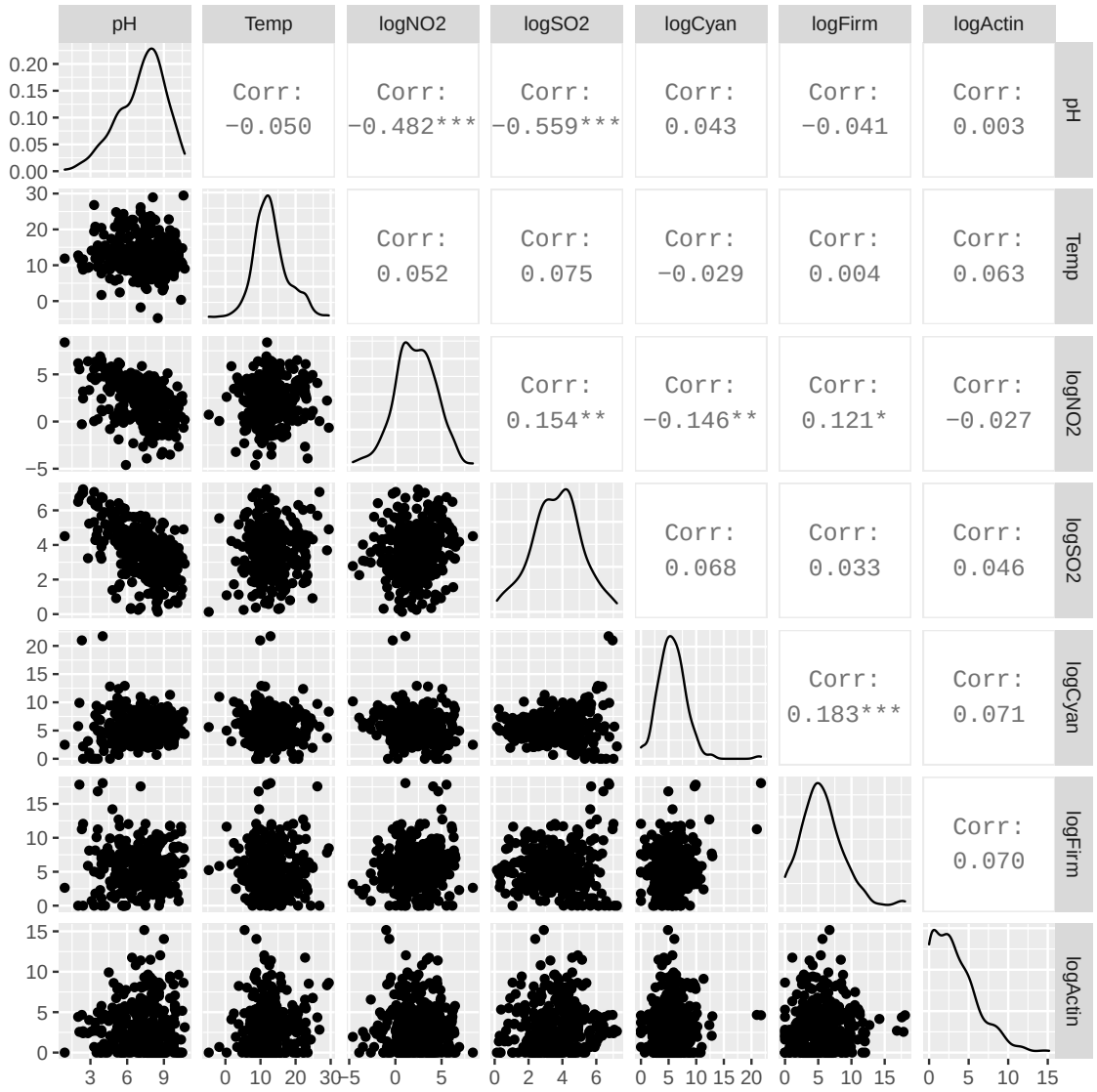
```
Advertising.data<-read.table("PFQ8.txt")

### Start with a summary and pairwise plots
summary(Advertising.data)
library(GGally)
ggpairs(Advertising.data,mapping=aes(colour=Type))

### log-transform the skewed predictors
library(dplyr)
ggpairs(Advertising.data%>%
        mutate(logDur=log(Duration),
               logCost=log(Cost),
               logTarget=log(Target.audience))%>%
        select(-c("Duration", # and remove the
                  "Cost",     # original predictors
                  "Target.audience")),
        mapping=aes(colour=Type))

### Different types of advertising campaigns have different patterns, so a
### facet_wrap may be appropriate.
ggplot(Advertising.data,
       mapping=aes(y=Sales.Increase,
                   x=Cost,
                   colour=Targeted))+
    geom_point()+
    facet_wrap(Type~.,scales="free")+   # use different scales on each subplot.
    geom_smooth(method="lm")
```

9. *The file* `PFQ9.txt` *contains the following data from an experiment about pollution and bacteria. The data set contains the following variables.*

Variable	*Meaning*
NO2	*The concentration of nitrogen dioxide in the water (ppm)*
SO2	*The concentration of sulphur dioxide in the water (ppm)*
pH	*The alkalinity of the water (7 is neutral, 0 is strong acid, 14 is strong alkali)*
Temp	*The temperature of the water (°C)*
Cyanobacteria	*The abundance of Cyanobacteria in the water*
Firmicutes	*The abundance of Firmicutes in the water*
Actinobacteria	*The abundance of Actinobacteria in the water*

Perform data exploration on this data set, and summarise (with tables and plots to support where appropriate) your initial conclusions about data issues and appropriate models. You should take into account any concerns with data collection and processing.

We start by making pairwise scatterplots, after log-transforming most of the variables. Since the bacterial abundances include some zeros, we add one to the abundance before taking the log.



After the log transformation, most of the variables are approximately normal. There is a slight break from non-normality for the log-bacterial abundances because these are actually count data.

There are several outliers in bacterial abundance. It may be appropriate to remove these. There is clear negative correlation between pH and the log-transformed abundance of pollutants NO_2 and SO_2. The scattergraph shows a typical bivariate normal pattern. This suggests a linear model between the log-transformed variables may be appropriate. It may be necessary to model the bacterial abundances using a count model, to account for the discrete nature of the data. Poisson regression can sometimes be used for this, but given the large variance of the bacterial counts, it is likely that there is overdispersion, so negative binomial regression is probably better.

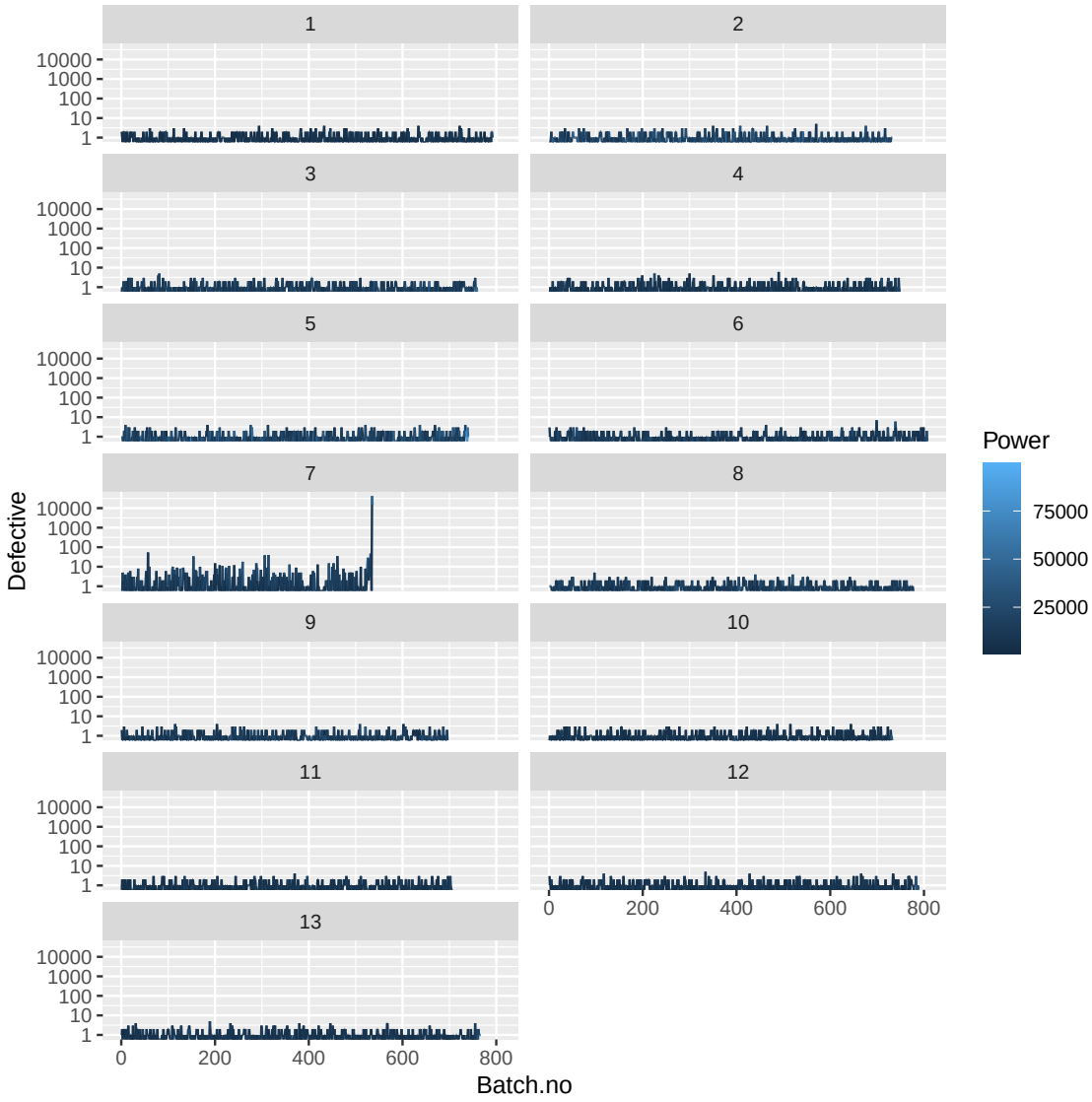
```
### read data
Pollution.data<-read.table("PFQ9.txt")

### summary and pairwise plots
summary(Pollution.data)
library(GGally)
ggpairs(Pollution.data)
library(dplyr)

### log-transform some variables
ggpairs(Pollution.data%>%
        mutate(logNO2=log(NO2),
               logSO2=log(SO2),
               logCyan=log(Cyanobacteria+1),
               logFirm=log(Firmicutes+1),
               logActin=log(Actinobacteria+1))%>%
        select(-c("NO2","SO2","Cyanobacteria","Firmicutes","Actinobacteria")))
```

10. *The file* `PFQ10.txt` *contains the following data from a company's quality control department. The company is monitoring the number of defective products produced over time and wants to develop a method for quickly detecting when the machine has started to produce defective units.*

Variable	*Meaning*
Machine	*The indentification number of the machine*
Batch.no	*The number of the batch produced. Machine batches of 1,000 units are produced sequentially, starting from number 1.*
Production.time	*The time taken to produce the batch*
Power	*The amount of energy used by the machine*
Defective	*The number of defective units in the batch*

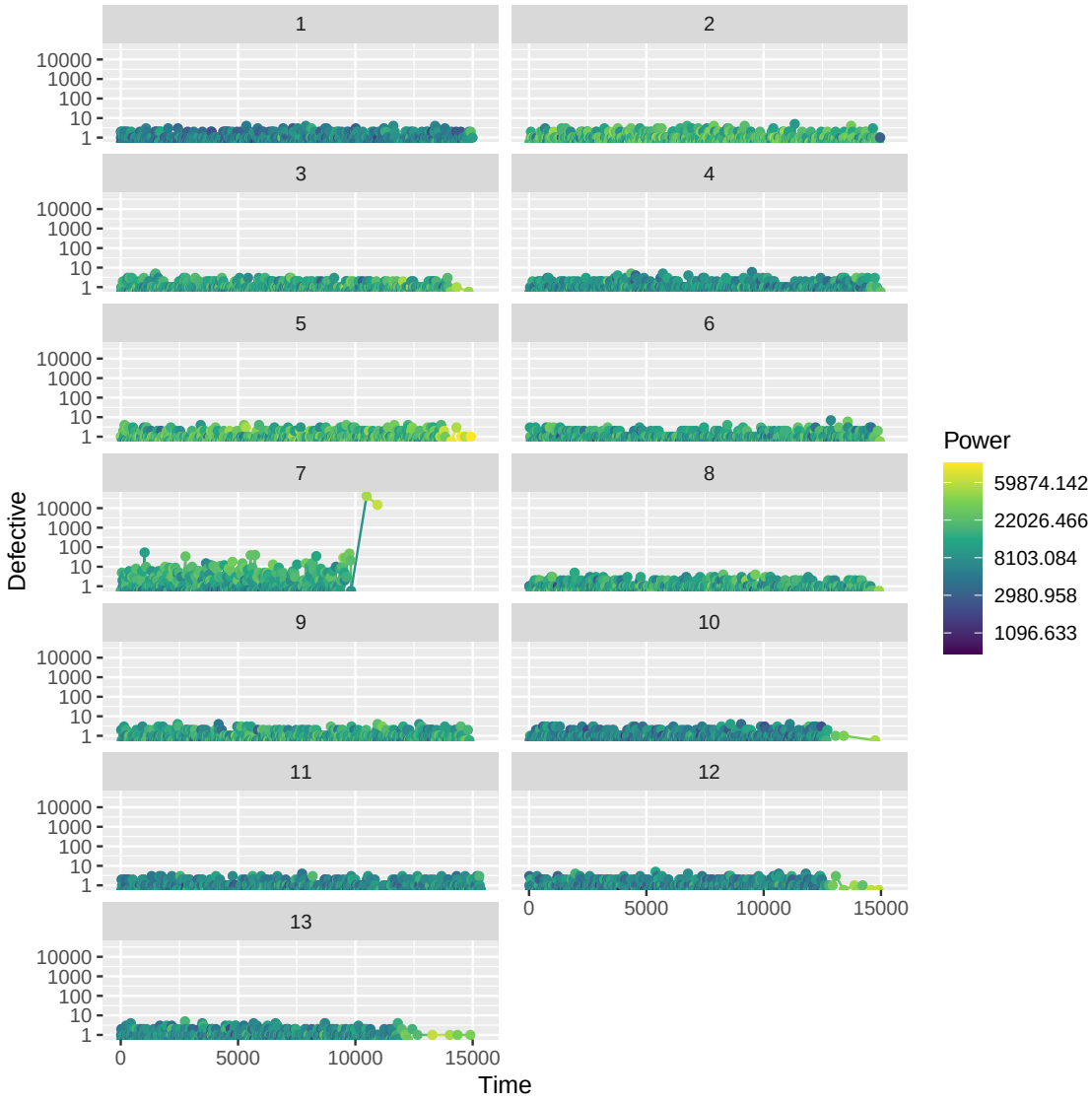
Perform data exploration on this data set, and summarise (with tables and plots to support where appropriate) your initial conclusions about data issues and appropriate models. You should take into account any concerns with data collection and processing.

In this case, the standard pairwise scatterplot is not very informative, because it does not account for the time series nature of the data. Instead, it makes more sense to make a plot for each machine, showing the series of batches from that machine.



In this plot, we have used a log-scale for the y-axis because of the distribution of number of defective units. From this plot, we see that most of the machines had a fairly stable output in terms of number of defective units, with the exception of Machine 7, which always produced more defective units, and at the end of its lifetime, started producing really large numbers of defective units.

It would also be valuable to monitor production time. This can easily be added to the x-axis, since the x-axis currently only records batch number, which is just a sequence. Therefore, if we calculate cumulative time that each machine is running, and use that as the x-axis, we can see more patterns.



From this plot, we see that towards the end of their lifetime, many machines start to use more power and take longer to produce batches. Thus, these predictors can be monitored to predict machine failure.

The major issue with the data is that there are only 13 machines, so the conclusions we learn may not be very general. To analyse the data, we should first decide if and when each machine has started failing, based on the observable variables. This allows us to make a binary response variable, which our objective is to predict as quickly as possible after it begins. We may need to use some time series methods. From the plot, the time series seems fairly stationary, with little autocorrelation. However, we should fit models to confirm this. Because the number of defective units is small and discrete, it may be necessary to use a discrete model, such as a Poisson distribution for this variable.

```
### Read data, summarise and pairwise scatterplot
Quality.data<-read.table("PFQ10.txt")
summary(Quality.data)
library(GGally)
ggpairs(Quality.data)


### Use facet_wrap to make one plot for each machine.
ggplot(Quality.data,
       mapping=aes(x=Batch.no,
                   y=Defective,
                   colour=Power))+
    geom_line()+
    facet_wrap(Machine~.,ncol=1)+
    scale_y_log10()

### Plot cumulative time instead of batch number to show production time
library(dplyr)
machinetime<-rep(0,13)
for(i in seq_len(13)){
    machinetime[i]<-sum(Quality.data$Production.time[Quality.data$Machine<i])
}

### machinetime is the sum of time for all lower-numbered machines.

ggplot(Quality.data%>%group_by(Machine),
       mapping=aes(x=cumsum(Production.time)-machinetime[Machine],
                   y=Defective,
                   colour=Power))+
    geom_line()+
    geom_point()+
    facet_wrap(Machine~.,ncol=2)+
    scale_y_log10()

### Use a log-scale for colour
ggplot(Quality.data%>%group_by(Machine),
       mapping=aes(x=cumsum(Production.time)-machinetime[Machine],
                   y=Defective,
                   colour=Power))+
    geom_line()+
    geom_point()+
    facet_wrap(Machine~.,ncol=2)+
    scale_y_log10()+
    scale_colour_viridis_c(trans="log")

### If this were a final plot, we would change the x-axis label to
### "time" to make it clearer.
```

11. *A doctor has collected the following data about age and various health conditions in the file* `PFQ11.txt`.

Variable	*Meaning*
Age	*The age of the patient*
BMI	*The BMI of the patient (weight (kg)/(height (m))^2*
SBP	*Systolic blood pressure of the patient*
Respire	*Respiratory rate of patient (breaths/second)*
Heart	*The heart rate of the patient*
Glucose	*Blood sugar of patient*
RBC	*Concentration of red blood cells*
WBC	*Concentration of white blood cells*

Fit a random forest model to predict the number of white blood cells for a patient from the other predictors. Use this model to predict the concentration of white blood cells for the patients in the file `PFQ11_test.txt`.

We use the caret package to train a random forest model. We use repeated cross-validation with 10 folds and 2 repeats to tune the `mtry` parameter in the range 1–7. This cross-validation selects `mtry=1`. The data set is not too large, so fitting 500 trees is not too time consuming.

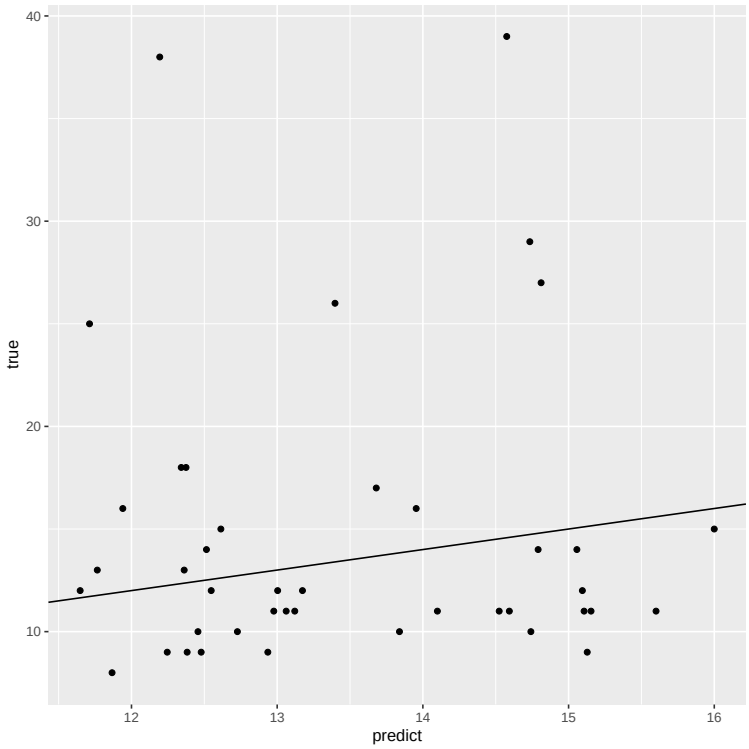
The fitted model assigns the following variable importances:

Variable	Relative Importance
Glucose	100.00
RBC	93.48
Age	65.51
BMI	55.95
Heart	48.96
SBP	48.05
Respire	0.00

For the test data, it produces the following predictions:

Number	Prediction	No.	Prediction	No.	Prediction	No.	Prediction	No.	Prediction	No.	Prediction
1	11.64782	8	13.95414	15	13.68006	22	12.54722	29	11.86680	36	12.34150
2	12.61349	9	14.09967	16	12.45632	23	15.15416	30	12.97681	37	11.76532
3	14.79098	10	15.09472	17	13.06177	24	13.00323	31	14.73404	38	14.59342
4	14.74125	11	12.93579	18	13.17359	25	14.57551	32	12.36176	39	14.81148
5	12.38216	12	11.71216	19	15.12799	26	15.60033	33	11.94046	40	12.72702
6	15.10648	13	15.05701	20	12.19403	27	12.47848	34	14.52375	41	13.12066
7	13.39702	14	15.99981	21	12.51457	28	13.83945	35	12.37437	42	12.24543

We compare these to the actual values on test data:



We see that there is some predictive ability, but the residuals are fairly large and skewed.

```
Health.data<-read.table("PFQ11.txt")
Health.test.data<-read.table("PFQ11_test.txt")
library(caret)

set.seed(2021041505)
### If you use this seed, you should get the exact results shown in
### the model solution. If you use a different seed (or don't set the
### seed) then your results will be similar, but different.


Health_rf<-train(x=Health.data[,-8], # remove response variable
                 y=Health.data$WBC,
                 method="rf",
                 trControl=trainControl(method="repeatedcv",number=10,repeats=2),
                 tuneGrid=expand.grid(mtry=seq_len(7)),
                 ntree=500)

varImp(Health_rf)
Health.predict<-predict(Health_rf,newdata=Health.test.data)

library(ggplot2)
### compare predictions to true values on test data
ggplot(data.frame(true=Health.test.data$WBC,
                  predict=Health.predict),
       mapping=aes(x=predict,y=true))+
    geom_point()+
    geom_abline(mapping=aes(slope=1,intercept=0))
```

12. *An insurance company has collected the following data on fire insurance claims in the file* `PFQ12.txt`.

Variable	*Meaning*
doy	*Day of year 1=1st January, 365=31st December*
rent	*The monthly rent for the premises*
alarm	*Whether the premises has a fire alarm*
sprinklers	*Number of sprinklers on the premises*
area	*Total area of the premises*
claim.amount	*The amount claimed for the loss event*

Fit a generalised linear model, with a gamma response variable, using a log link, and log-transforming the predictors rent and area, to predict the claim amount from the other predictors.

Use this model to predict the claim amounts for the claims in the file `PFQ12_test.txt`.

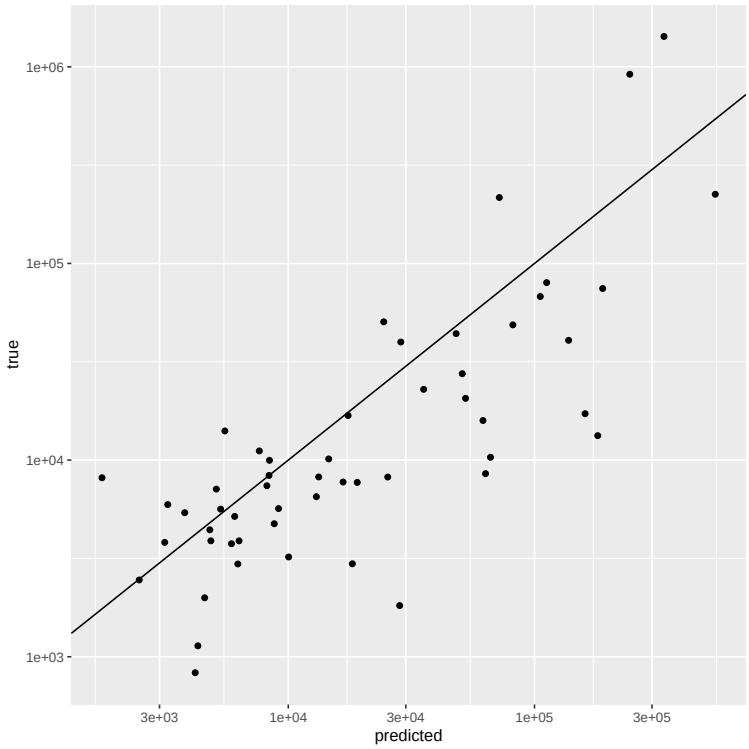
```
### read data
Fire.data<-read.table("PFQ12.txt")
Fire.test.data<-read.table("PFQ12_test.txt")

### fit gamma regression
gamma.regression<-glm(claim.amount~log(rent)+log(area)+doy+alarm+sprinklers,
                      data=Fire.data,
                      family=Gamma(link="log")) # log is canonical link for gamma
summary(gamma.regression)

### predict data and compare to true values
Q12_predictions<-predict(gamma.regression,newdata=Fire.test.data,type="response")
library(ggplot2)

ggplot(data.frame(true=Fire.test.data$claim.amount,
                  predicted=Q12_predictions),
       mapping=aes(x=predicted,
                   y=true))+
    geom_point()+
    scale_x_log10()+ # use log scale because variance is related to mean
    scale_y_log10()+ # and the data have better spread on log scale.
    geom_abline(mapping=aes(slope=1,intercept=0)) # add y=x line.
```

The gamma regression identifies the predictors log-rent and log-area as clearly significant, and the predictor alarm as only marginally significant.

Using the fitted model to predict the test data, we make the following predictions.

Number	Prediction	No.	Prediction	No.	Prediction	No.	Prediction	No.	Prediction	No.	Prediction
1	50887.141	10	9132.275	19	3149.101	28	4801.880	37	5877.836	46	25346.475
2	8186.525	11	13286.025	20	137581.235	29	189204.612	38	81674.038	47	19053.795
3	6057.860	12	14596.779	21	8385.098	30	5101.568	39	63258.180	48	105528.516
4	160509.409	13	180641.570	22	7628.137	31	52478.107	40	1751.861	49	28641.376
5	4188.211	14	13005.143	23	243763.899	32	16695.138	41	4574.113	50	48047.573
6	8769.903	15	10033.773	24	8350.792	33	542251.021	42	35456.903	51	4293.542
7	335655.972	16	3240.651	25	17498.028	34	66250.877	43	18206.174	52	28337.866
8	61756.890	17	112012.326	26	6237.051	35	2481.413	44	6310.153	53	71956.856
9	5312.797	18	3795.993	27	5526.378	36	4848.507	45	24408.726		

Compared with the observed values, we get the following plot:



We have clearly predicted the values reasonably well. There may be a slight upward bias. This could be due to the log transformation. For the gamma distribution, the median is below the mean, so if our prediction is the mean value of the gamma distribution, most observed values should be below the prediction (note that the axes on this plot are log-transformed.

13. *A scientist has collected the following data about planets in the file* `PFQ13.txt`.

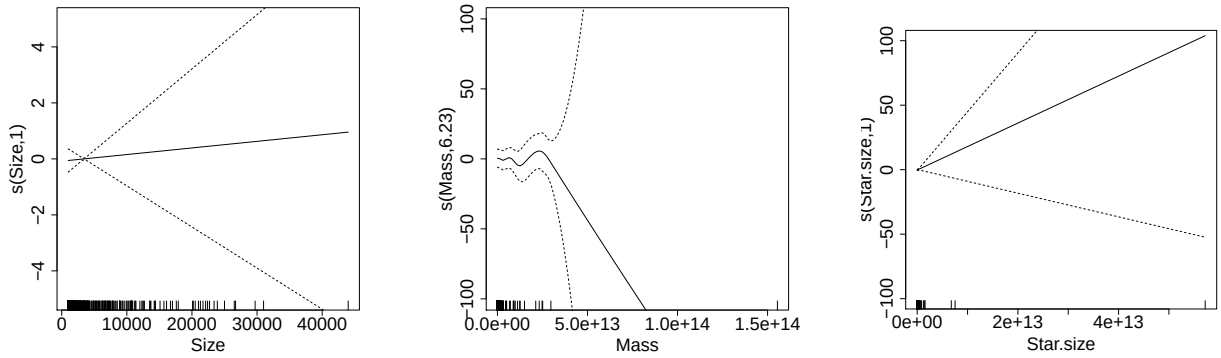
Variable	*Meaning*
Size	*The radius of the planet (km)*
Mass	*The mass of the planet (tonnes)*
Composition	*What the planet is mostly made from*
Star.size	*The radius of the star that the planet orbits*
Orbit.distance	*The distance at which the planet orbits the star*
Revolution.time	*The time taken for the planet to complete a turn on its axis*
Earth.distance	*Distance from Earth's solar system (light-years)*
Oxygen	*Whether the planets atmosphere has a detectable quantity of oxygen.*

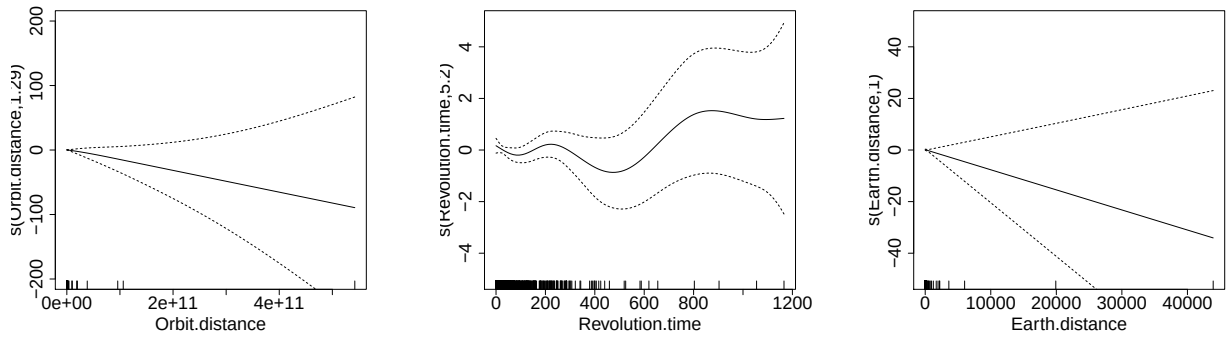
Fit a generalised additive model to determine whether a planets atmosphere has a detectable quantity of oxygen, and use it to predict the results for the planets listed in the file `PFQ13_test.txt`.

We use the mgcv package to fit the GAM.

```
### read data

Planet.data<-read.table("PFQ13.txt")
Planet.test.data<-read.table("PFQ13_test.txt")

### use the mgcv library for GAM fitting
library(mgcv)
Planet.gam<-gam(Oxygen~s(Size)+s(Mass)+Composition+s(Star.size)+s(Orbit.distance)+s(Revolut
                Planet.data,
                family=binomial(link="logit")) ### logit link is default for 0-1 data.

plot(Planet.gam) ### check the fitted splines.

### plot them individually with the following code

plot.gam(Planet.gam,select=1,ylim=c(-5,5)) # By default the plot uses the same
plot.gam(Planet.gam,select=2,ylim=c(-100,100)) # y-axis limits for each variable
plot.gam(Planet.gam,select=3,ylim=c(-100,100)) # which is not desirable
plot.gam(Planet.gam,select=4,ylim=c(-200,200))
plot.gam(Planet.gam,select=5,ylim=c(-5,5))
plot.gam(Planet.gam,select=6,ylim=c(-50,50))

summary(Planet.gam)

### predict test data and compare with true values.
Planet.predict<-predict(Planet.gam,newdata=Planet.test.data,type="response")
rbind(round(Planet.predict,4),Planet.test.data$Oxygen)
```

In the fitted model, only the composition is significant, and only composition rock is significantly different from the other compositions. However, the model does fit smooth functions for all the continuous predictors:





The predicted probabilities for the test data are given in the following table (planets with oxygen are in bold):

No.	Prediction	No.	Prediction	No.	Prediction	No.	Prediction
1	0.2427	**9**	0.4618	**17**	0.2126	25	0.3717
2	0.9332	10	0.4721	**18**	0.2242	**26**	0.4343
3	0.2580	11	0.1118	**19**	0.1666	27	0.2753
4	0.2221	**12**	0.3794	20	0.2460	28	0.4797
5	0.2640	13	0.2147	21	0.1637	**29**	0.2696
6	0.4315	14	0.0922	22	0.2015	30	0.2613
7	0.4141	15	0.2512	**23**	1.0000	31	0.2531
8	0.1967	**16**	0.0000	**24**	0.2418		

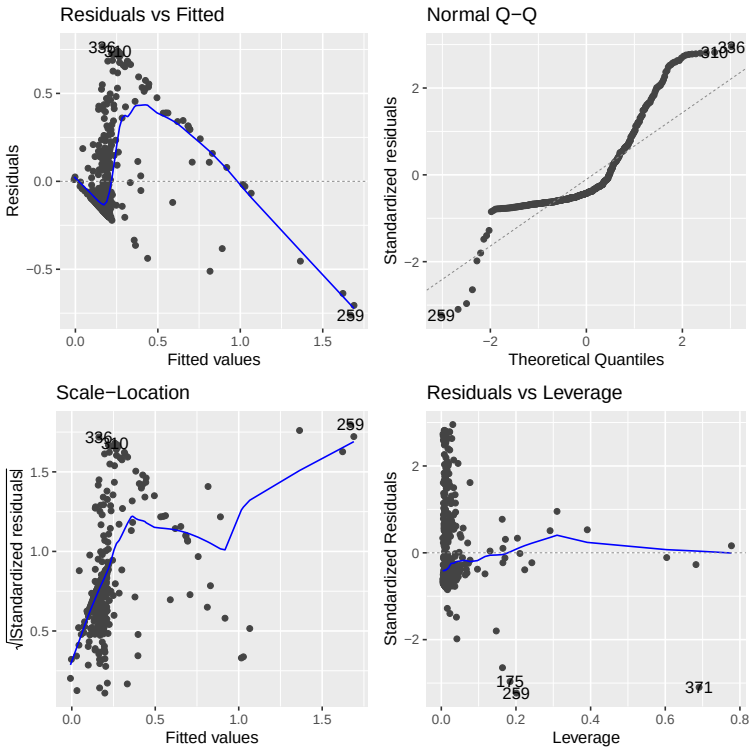
We see that the predicted probabilities are very bad in some cases, with confident wrong predictions.

14. *A data scientist is analysing data about spending and financial security in the file* `PFQ14.txt`.

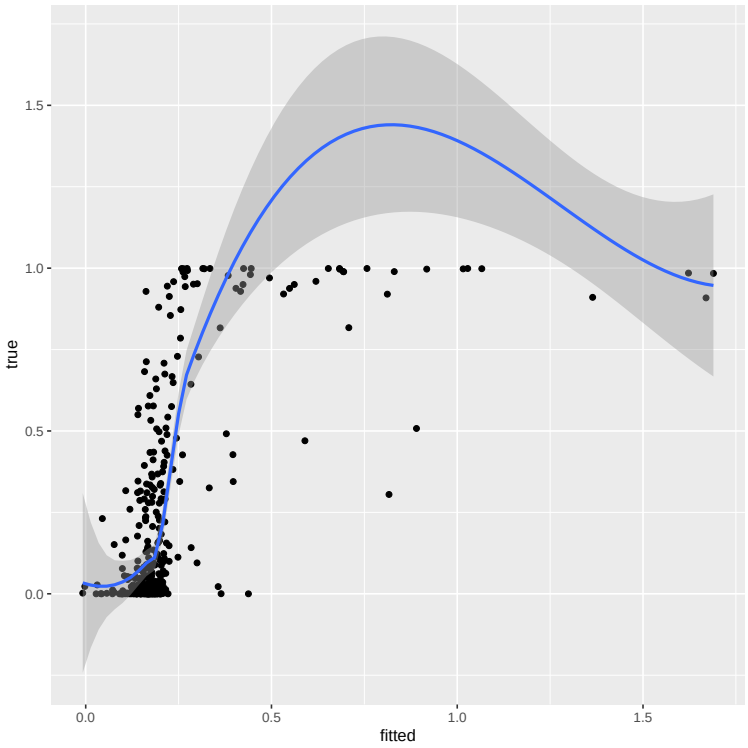
Variable	*Meaning*
Year	*The year in which the data were collected*
Family.size	*The number of individuals supported in the household*
Food.spending	*Average Total monthly amount spent on food*
Clothing.spending	*Average Total monthly amount spent on food*
Housing.spending	*Average monthly spending on rent or mortgage payments*
Travel.spending	*Average monthly spending on travel*
Entertainment.spending	*Average monthly spending on entertainment*
Annual.salary	*The annual combined salary of the household*
Pension.percent	*The expected annual combined pension of the household, expressed as percent of current income.*
Age	*The age of the highest earner in the household*
Debt	*The total level of debt owed by the household*
Assets	*The total value of assets owned by the household*
Retirment.success	*Whether the family was able to retire at their planned time or earlier with the intended pension*

He has fitted a linear model to predict which households will successfully be able to retire, using the code in the file `PFQ14.R`. *Perform diagnostics to test which of the assumptions of this model are reasonable. What changes would you suggest making to the model to better model the data?*

From the standard diagnostic plots:



we see that the residuals show a clear non-linear relation with the fitted values. This is partly because the response variable is between 0 and 1. It is also because the mean response shows a highly non-linear relation with the fitted value. Looking at a plot of fitted value versus true value



we see that a logistic link function might be appropriate. The other diagnostic plots lead to similar conclusions — the residuals are clearly not normal, because they lie in the interval $[0, 1]$. The scale of the residuals varies with fitted value, but this can be explained by the non-linear relationship.

Based on this, I would suggest refitting with a logistic link function, and probably a beta distribution for the response variable.

```
### standard diagnostic plots.
library(ggfortify)
autoplot(lm_model_full)

summary(Spending.data)

ggplot(data.frame(fitted=lm_model_full$fitted.values,
                  true=Spending.data$Retirement.success),
       mapping=aes(x=fitted,y=true))+
    geom_point()+geom_smooth()

### Compare predicted and true values
ggplot(data.frame(fitted=lm_predict,
                  true=test.data$Retirement.success),
       mapping=aes(x=fitted,y=true))+
    geom_point()+
    geom_smooth()
```

15. *An actuary is reviewing data about catastrophe insurance claims in the file* `PFQ15.txt`.

Variable	*Meaning*
Event.type	*The type of catastrophe event*
Area.size	*The size of the area affected (km^2)*
Area.people	*The number of people living in the affected area*
Total.damage	*The total claims for the affected area*

She has fitted a generalised additive model, a random forest model and a generalised linear model including a number of interaction terms and polynomial terms, to predict the total damage, using the code in the file `PFQ15.R`. *Assess which of these models is better at predicting the data. [You may need to modify the code provided to do this.]*

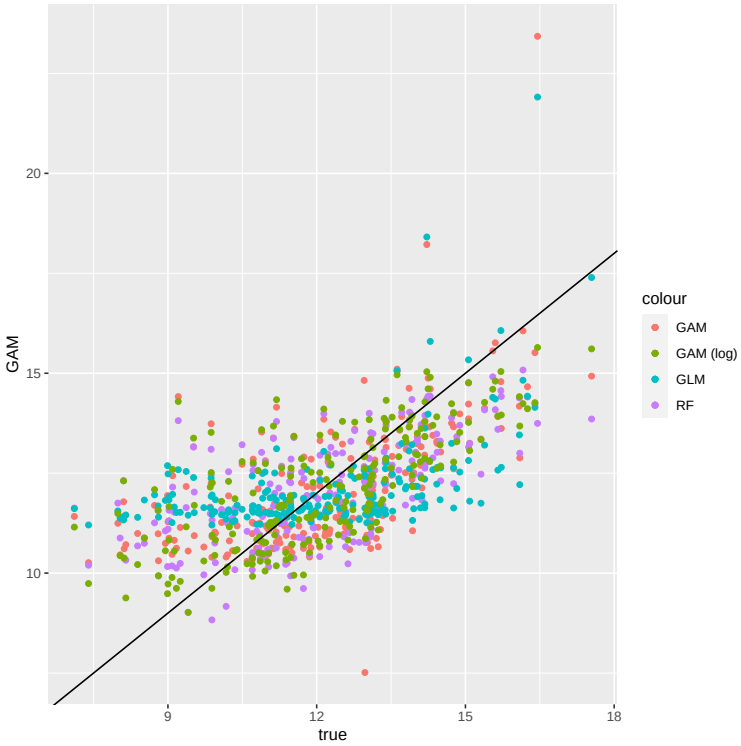
Because there are only 220 data points, cross-validation is a good way to compare accuracy of the methods, since it allows us to get test predictions for all 220 data points, giving us a larger sample to compare the models.

```
### Use 5-fold cross-validation to assess prediction

### create folds
Folds<-createFolds(Catastrophe.data$Total.damage,5)

### prepare empty vectors to store predictions.
gam_pred<-rep(0,220)
glm_pred<-rep(0,220)
rf_pred<-rep(0,220)
gam_full_pred<-rep(0,220)


for(i in seq_len(5)){ # for each fold
    training.data<-Catastrophe.data[-Folds[[i]],]
    test.data<-Catastrophe.data[Folds[[i]],]

    ## fit models to training data
    gam_model_fold<-gam(log(Total.damage)~Event.type+s(Area.size)+s(Area.people),data=train
    RF_model_fold<-train(y=log(training.data$Total.damage),
                         x=training.data%>%select(-c("Total.damage")),
                         method="rf",
                         trControl=trainControl(method="repeatedcv",
                                                number=5,
                                                repeats=2),
                         tuneGrid=expand.grid(mtry=seq_len(3)),
                         ntree=500)
    glm_model_fold<-glm(log(Total.damage)~.,data=training.data)
    gam_model_all_log_fold<-gam(log(Total.damage)~s(log(Area.size))+s(log(Area.people)),dat

    ## predict test data
    gam_pred[Folds[[i]]]<-predict(gam_model_fold,newdata=test.data)
    glm_pred[Folds[[i]]]<-predict(glm_model_fold,newdata=test.data)
    rf_pred[Folds[[i]]]<-predict(RF_model_fold,newdata=test.data)
    gam_full_pred[Folds[[i]]]<-predict(gam_model_all_log_fold,newdata=test.data)
}

### calculate MSE (on log scale)
sum((gam_pred-log(Catastrophe.data$Total.damage))^2)
sum((glm_pred-log(Catastrophe.data$Total.damage))^2)
sum((rf_pred-log(Catastrophe.data$Total.damage))^2)
sum((gam_full_pred-log(Catastrophe.data$Total.damage))^2)

### plot predictions vs. true values
compare_predictions<-data.frame(true=log(Catastrophe.data$Total.damage),
                                GAM=gam_pred,
                                GLM=glm_pred,
                                RF=rf_pred,
                                GAM_log=gam_full_pred)

ggplot(compare_predictions,
       mapping=aes(x=true,
                   y=GAM,
                   colour="GAM"))+
    geom_point()+
    geom_point(mapping=aes(y=RF,colour="RF"))+
    geom_point(mapping=aes(y=GLM,colour="GLM"))+
    geom_point(mapping=aes(y=GAM_log,colour="GAM (log)"))+
```

We compare the MSE of these predictions on both the log and the original scale. Depending on context, either measure of accuracy may be prefered. If we are looking to minimise absolute error, then the original scale is prefered. On the other hand, relative error might be more important, since the premium is usually set with a certain percentage loading to cover the relative error, rather than the absolute error. This is better estimated using the log-scale MSE. Furthermore, the original scale MSE is heavily influenced by the large claims, so is driven by a very small number of data points, while the log-scale MSE uses more of the data points.

Method	MSE on log scale	MSE on original scale
GAM	596.0357	2.232511×10^{20}
GLM	673.9013	1.063913×10^{19}
RF	518.7965	2.457232×10^{15}
GAM (log-transformed predictors)	452.0215	1.984853×10^{15}

We see that by these measures the GAM with log-transformed predictors performs best, followed by random forest. For the GAM and GLM, the measures of accuracy disagree on which is better. We also plot the predicted versus observed values for each method, to ensure that these differences in MSE are general patterns, rather than being caused by a small number of outliers.



We see that while there are outliers in the predictions for GLM and GAM, the predictions are generally worse for these models. We conclude that the GAM with log-transformed predictors is the best model.

16. *A scientist has analysed some data and written the following conclusion to their paper.*

The purpose of this analysis was to determine the extent to which an individual's fertility can be predicted from genetic data. The data were taken from the study of [1]. The researchers collected genetic samples from childless couples, both aged 20–30, who were trying to conceive. They then followed the couples over a 1-year period, recording which of them successfully conceived during that period. They also conducted a survey about other factors that may be influencing their success.

There were a total of 32,041 couples enrolled in the study, from four different cities: London, UK; Montreal, Canada; Qingdao, China; and Rome, Italy. The city where the couple were based was included as a predictor variable, since the effects of several predictor variables could be affected by the location. The study was conducted over a period of 3 years from 2013–2016. There was an attrition rate of 13% in the study. Couples who did not complete the study were removed from the analysis (even if they had conceived before that time). This approach is open to some concerns, since couples who stopped the study might be different from couples who remained in the study. It is possible that more couples who failed to conceive left the study, leading to potentially biased results. Further research is needed to develop a better method to handle attrition. Alternatively, a study with a shorter follow-up period and more couples might have lower attrition rates, leading to less potential bias in the conclusions.

The genetic data comprised 12,358 single nucleotide polymorphisms (SNPs) from each male participant in the study, and 12,704 SNPs from each female participant. Thus for each couple, there are a total of 25,062 genetic predictors, in addition to 72 other predictors from the survey. There were some missing responses from the survey variables, and other ambiguous responses were removed from the data. [1] removed three of the survey predictors from the analysis — sex time of day, menstual frequency, and sex frequency, because many responses were missing for these predictors. In our analysis, we were able to include these responses because the random forest method we used is able to handle missing values.

[1] was unable to find any significant genes associated with fertility when correcting for the effect of other predictors. However, the methodology used was very conservative. In this paper, we used a new approach to the statistical modelling, which is more able to identify interactions between genes. This approach was first suggested by [2]. The basic idea is to divide the data into two parts, use the first part to screen the genetic predictors, to find a shortlist of the predictors most likely to be associated with fertility, then to use the second part of the data to fit a random forest model to predict number of children from the selected predictors.

The work of [2] and [3] has shown that this approach can be very effective at identifying complex interactions between genes in other contexts. We have modified the approach slightly, following the suggestion of [4] to divide into two subsets for screening the predictors. We found that this approach produced a lower test error on the data.

To assess the predictive performance of our method, we performed our approach using two thirds of the data as training data, and one third of the data as test data. The training data was divided into two screening subsets, each comprising one sixth of the training data, and one modelling subset comprising the remaining two thirds of the training data. Cross-validation was used on the modelling subset to select the tuning parameter for random forest. 81% of the couples who completed the study successfully conceived during that period. We found that using only the survey predictors, we were able to predict with 86% accuracy whether a couple would conceive. Using the genetic data, we were able to increase this to 88%. This clearly indicates that the genetic data is useful for predicting conception success.

For comparison, we used LASSO to fit a generalised linear model to predict whether a couple would conceive. For this model, there was no improvement in test accuracy over the model with just the survey predictors. This indicates that the effect of the genetic data on fertility is not linear. Since the genetic variables are mostly binary, this indicates that interactions between the genes are responsible for the effect on fertility.

From the variable importance given by random forest, the most important genes are MEB12, INT21 and RMS7. Fitting a random forest model using only those three genes produces a test accuracy of 87%, indicating that the effect of genes on fertility is spread over a large number of genes, each of which has only a minor effect. The LASSO fitting with logistic regression selected 4 genes: MEB12, RMS7, OLB13 and QJA14. The first two of these were important in the random forest method. However, INT21, which was important under random forest was not selected by LASSO, suggesting that this gene is only important because of its interaction

with other genes. The genes OLB13 and QJA14 were very low in the random forest variable importance. This may suggest that these genes are surrogates for a combination of other genes, that may be better predictors in a nonlinear model.

We also looked at different performance measures. In particular, we considered weighted accuracy, where we increased the weight of the couples who failed to conceive, so that both groups had the same weight. Under this performance measure, using only the survey variables, the LASSO logistic regression achieves a weighted test accuracy of 64%, while random forest achieves a weighted test accuracy of 71%. When the genetic variables are included, the weighted accuracy improved to 66% for LASSO and 74% for random forest. This shows that the genetic predictors are important even in the linear model.

write an abstract for the paper.

In this paper we re-analyse the data from [1] on the relation between an individual's genome and their fertility. The dataset includes 25062 genetic predictors and 72 environmental predictors, for 32041 childless couples, trying to conceive, and information about whether the couples managed to conceive within a year.

In the previous reasearch, no significant association between the genetic data and fertility was found. We applied an approach based on data-splitting and random forest [2,3,4].

81% of the couples in the study conceived during the 1-year period. Using only environmental variables, we predicted with 86% test accuracy whether the couple would conceive. Using the genetic variables in addition, the test accuracy improved to 88%.

We identified MEB12, INT21 and RMS7 as the most important genes for predicting fertility. Using only these three genes with the environmental variables, test accuracy was 87%. Thus these variables improve prediction over just the environmental variables, but other genes with smaller effects are still important.

For comparison, we fitted a generalised linear model, with LASSO variable selection. This model showed no improvement in test accuracy over the model with just environmental predictors. Thus it appears that interactions between genes are important for predicting fertility.

17. *The following quotes come from a report on the effect of financial hedging on a company's profit. Where in the report should they be placed? Justify your answers.*

(i)

The MSE of the random forest model on the test data was 22.4. This is not significantly better than the generalised additive model. Because the generalised additive model is more interpretable, we therefore prefer it to the random forest model.

This is probably from the "Data Analysis" or "Results" section. It is selecting the model to use, based on the data, which would generally be done in the "Results" section. For a paper focused on data analysis methodology, this quote might be in the executive summary. However, since the report is focused on the effect of hedging, it is probably not necessary to put this statement in the executive summary.

(ii)

While hedging against adverse financial conditions reduces the expected profits by 2%, it greatly reduces the probability of a large loss. Given the company's limited capacity to absorb large losses, we consider this reduction in risk to justify the proposed hedging scheme.

This seems to be the main conclusion of the report, and is stated fairly concisely. A statement like this should definitely appear in the executive summary. The statement might also appear in the "Conclusion" or "Discussion" section. It could be expanded in that section, but could also be repeated in exactly this form.

(iii)

Previous work by [1] used a neural network to predict profit values. While the accuracy of their predicted profits was reasonable, their method ignored the dangers of currency fluctuations and the timing of conversion. [2] modified this approach to account for timing of currency conversion. However, their changes to the model caused a decrease in accuracy on the test data.

This clearly belongs in the "Introduction" or "Background" section. It is discussing previous work on the problem, using different methods, and explaining why this previous work is not sufficient for the purposes of the report. It is sometimes reasonable to mention previous work in the executive summary, but not at this level of detail.

(iv)

There may be potential to develop a hedging scheme that is able to achieve a similar reduction in risk with lower reduction in profit, for example, the schemes suggested by [3] are promising. However the proposed scheme has a major advantage of simplicity, which can lead to reduced implementation costs. The reduction in implementation costs, which can be quite substantial [4], should be enough to outweigh any theoretical advantages of the superior hedging scheme.

This clearly belongs in the "Conclusions" or "Discussion" section. It is discussing potential future work, and explaining why stopping before performing the future work is a reasonable decision.

18. *A scientist has analysed the data in the file* `PFQ18_a.txt` *using the commands in* `PFQ18.R`*. The data show the change in log abundance of a number of common gut bacterial genera, in response to treatment of patients with antibiotics. For reference, the taxonomy of the relevant bacteria is in the file* `PFQ18_b.txt`*. She has concluded the following:*

    (a) *The phyla* Firmicutes *and* Proteobacteria *have very uniform levels of reduction in response to glycopeptides.*

    (b) *The classes in phylum* Firmicutes *react in very different ways to tetracyclines*

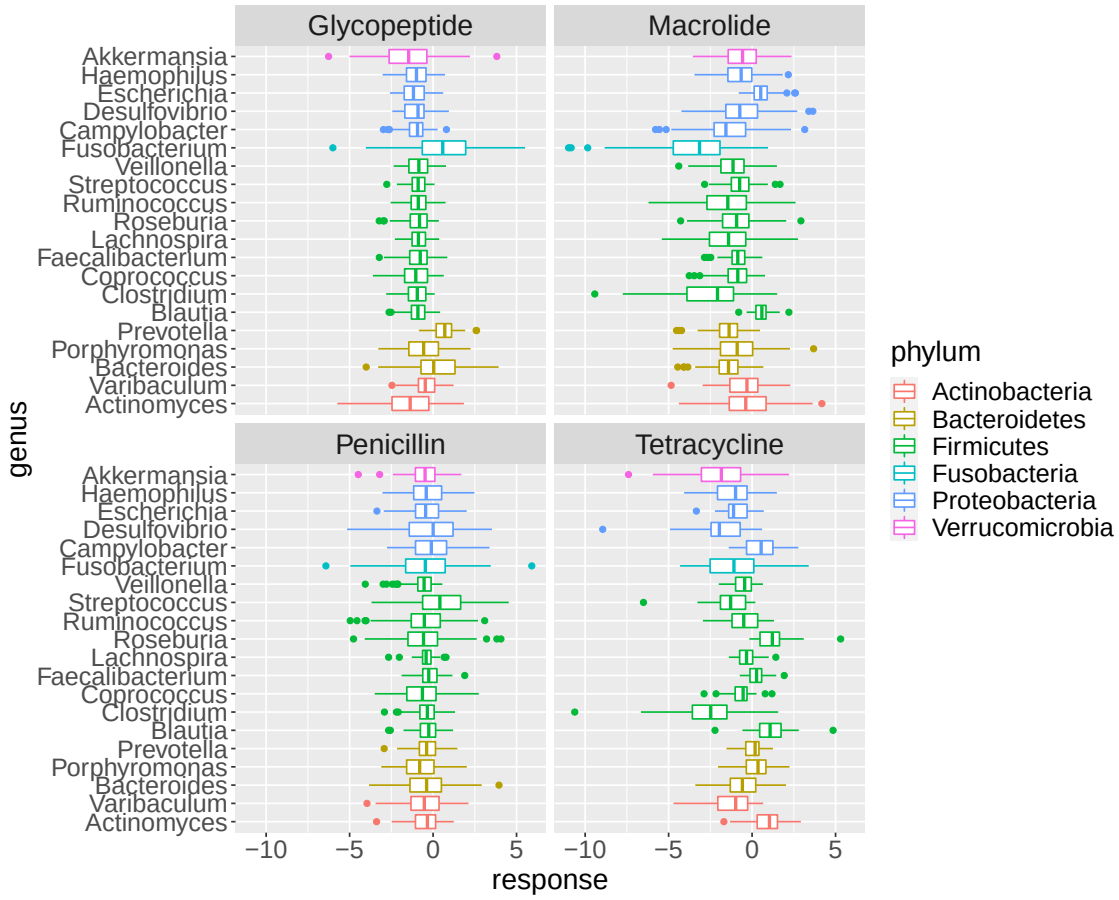
    (c) *From the responses of 6 indicator genera — Actinomyces, Akkermansia, Clostridium, Desulfovibrio, Fusobacterium, and Roseburia — it is possible to predict the response of all other genera with reasonable accuracy.*

    Display the data and analysis results so as to demonstrate the conclusions.

    We use the following theme for all the plots:

```
plottheme<-theme(axis.title=element_text(size=18),
                 axis.text=element_text(size=16),
                 legend.text=element_text(size=16),
                 legend.title=element_text(size=18),
                 strip.text=element_text(size=18))
```

One suitable plot is a boxplot

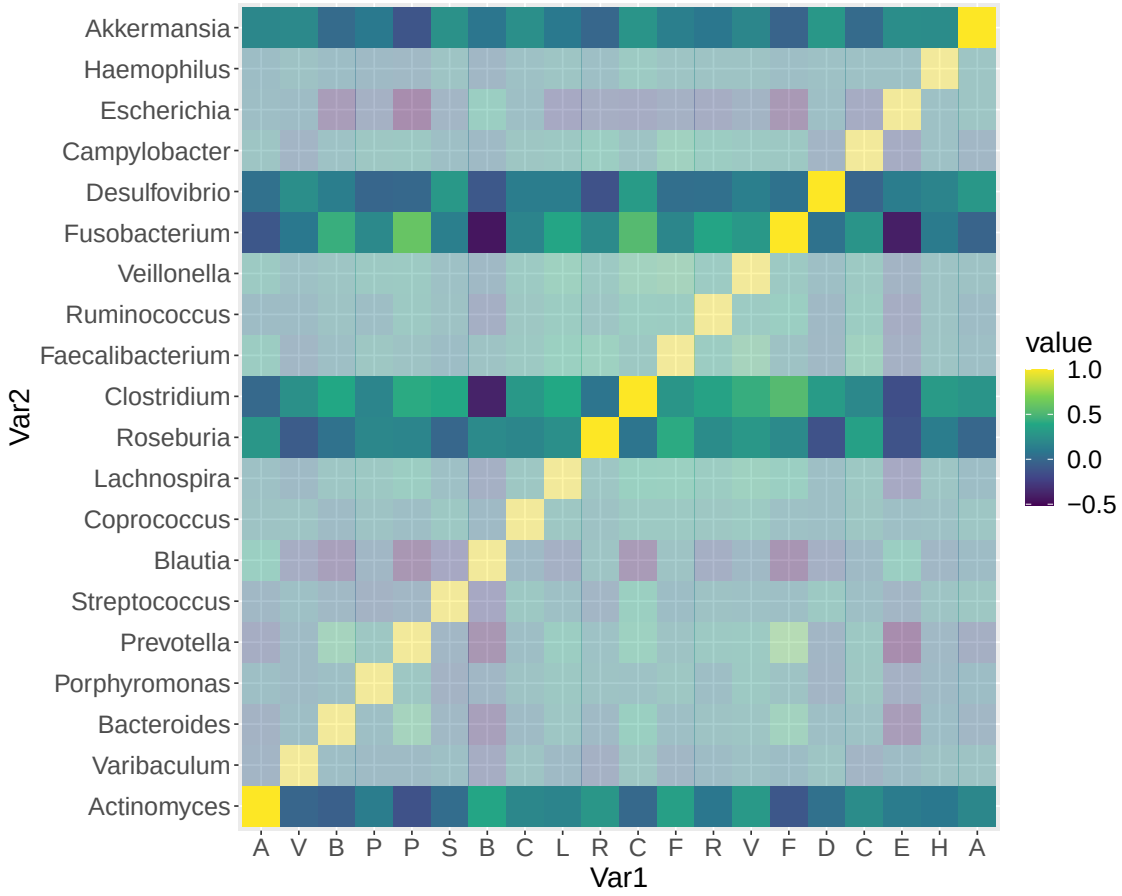


This clearly shows conclusions (a) and (b), allowing a comparison of the distributions for each genus. It would also be possible to use a violin plot, which shows the whole distribution for each genus. This might show more patterns, but for the conclusions (a) and (b), the boxplot is sufficient.

```
plottheme<-theme(axis.title=element_text(size=18),
                 axis.text=element_text(size=16),
                 legend.text=element_text(size=16),
                 legend.title=element_text(size=18),
                 strip.text=element_text(size=18))
```

a

To show Conclusion (c), we can plot a heatmap of pairwise correlations, highlighting the chosen genera.

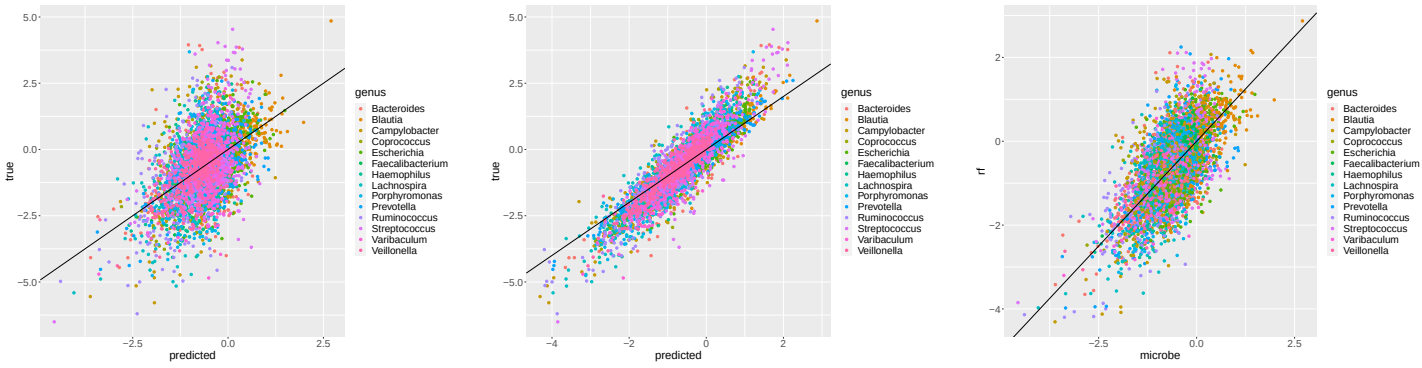



```
plottheme<-theme(axis.title=element_text(size=18),
                 axis.text=element_text(size=16),
                 legend.text=element_text(size=16),
                 legend.title=element_text(size=18),
                 strip.text=element_text(size=18))
```

b

We can also compare the predicted and true values for the prediction using the indicator microbes, and for a random forest prediction using the drug information

Indicator genera Random forest Compare models




```
plottheme<-theme(axis.title=element_text(size=18),
                 axis.text=element_text(size=16),
                 legend.text=element_text(size=16),
                 legend.title=element_text(size=18),
                 strip.text=element_text(size=18))
```

c