Workshop: Logistic Regression Pima Indians Example Using R

Dr. Teresa Brunsdon

University of Warwick

March 2021

Contents

1 Introduction

This exercise will illustrate another example of plotting and then modelling data in order to understand relationships involving a binary dependent variable. It will then go on to fit a model and produce some of the model diagnostics.

NOTE: In this session we will be using TWO ANOVA and TWO boxplot functions. One is `anova` which is in the main stats R package. The other is `Anova` which is part of the car package. Similarly, `boxplot` is the main stats R package whilst `Boxplot` is from the car package. Please make sure you use the correct one. They are distinguished by the first letter which is either a capital or not.

1.1 The Data Set

We will be using the Pima Indian data set. This data set is a well-known one and is available in various places such as the UCI website. We will use the version that is available with the MASS (Ripley, et al. 2021) package. It concerns a population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, who were tested for diabetes according to World Health Organization criteria. The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. The version in the MASS package uses the 532 complete records after dropping the (mainly missing) data on serum insulin. It consists of the following variables:

npreg: number of pregnancies.
glu: plasma glucose concentration in an oral glucose tolerance test.
bp : diastolic blood pressure (mm Hg).
skin: triceps skin fold thickness (mm).
bmi: body mass index (weight in kg/(height in m)\^2).
ped: diabetes pedigree function.
age: age in years.
type: Yes or No, for diabetic according to WHO criteria.

1.2 Setting Up

You will need to download some code that will install the packages required, read in the data and then produce a variety of plots etc. There is an additional file that you should already have from a previous workshop that adjusts some of the plots in the ggpairs plot. Here it is renamed, and it is best if you download this version for use here. .

- ***ST404 Logistic workshop Pima Example.R***
- ***ggpairsPima.R***

1.3 Installing and Loading Packages

We will be making use of a number of packages; some you will have used before and should therefore have already installed. The code at the start of the file (BLOCK I) checks if you have these installed and if not installs these packages. It also then loads them. The packages include:

- `car` (Fox and Weisberg 2019);
- `Stat2Data` (Cannon, et al. 2019);
- `plyr` (Wickham, The Split-Apply-Combine Strategy for Data Analysis 2011);
- `MASS` (Ripley, et al. 2021)
- `arm` (Gelman , et al. 2018) and
- `hnp` (de Andrade Moral, Hinde and Demetrio 2018)
- `sjPlot` (Lüdecke, et al. 2021)
- `DAAG` (Maindonald and Braun 2020)
- `effects` (J. Fox, et al. 2020)
- `ROCR` (Sing, et al. 2020)
- `pROC` (Robin, et al. 2021)

In addition the second R file checks and installs the following:

- `ggplot2` (Wickham, ggplot2: Elegant Graphics for Data Analysis 2016)
- `GGally` (Schloerke, et al. 2021)
- `ggmosaic` (Jeppson, et al. 2021)

1.3.1 Install and load on your system

1) Load and open the first of the two R files above (***ST404 Logistic workshop Pima Example.R)***. Then Run BLOCK *I*.

2 Loading the Data

2.1 Preparing the Data

2) Run BLOCK II which loads the data. The data comes as two data sets - a training and a test set. In this exercise we are only going to produce plots (although we will fit one model along the way) so for now we join these together.

3 Initial Plotting: Univariate

3.1 Initial Univariate Plots

3) Run BLOCK *III* to produce the univariate plots for all the variables. Notice that the code contains examples of how to control the number of bars, scale and tick marks of the axes. Notice that ***Age, ped*** (Diabetes pedigree) and ***npreg***, (number of pregnancies) are all positively skewed, whilst the other variables are almost symmetrical. There does appear to be the odd extreme value especially for ***skin*** (Triceps skin thickness).

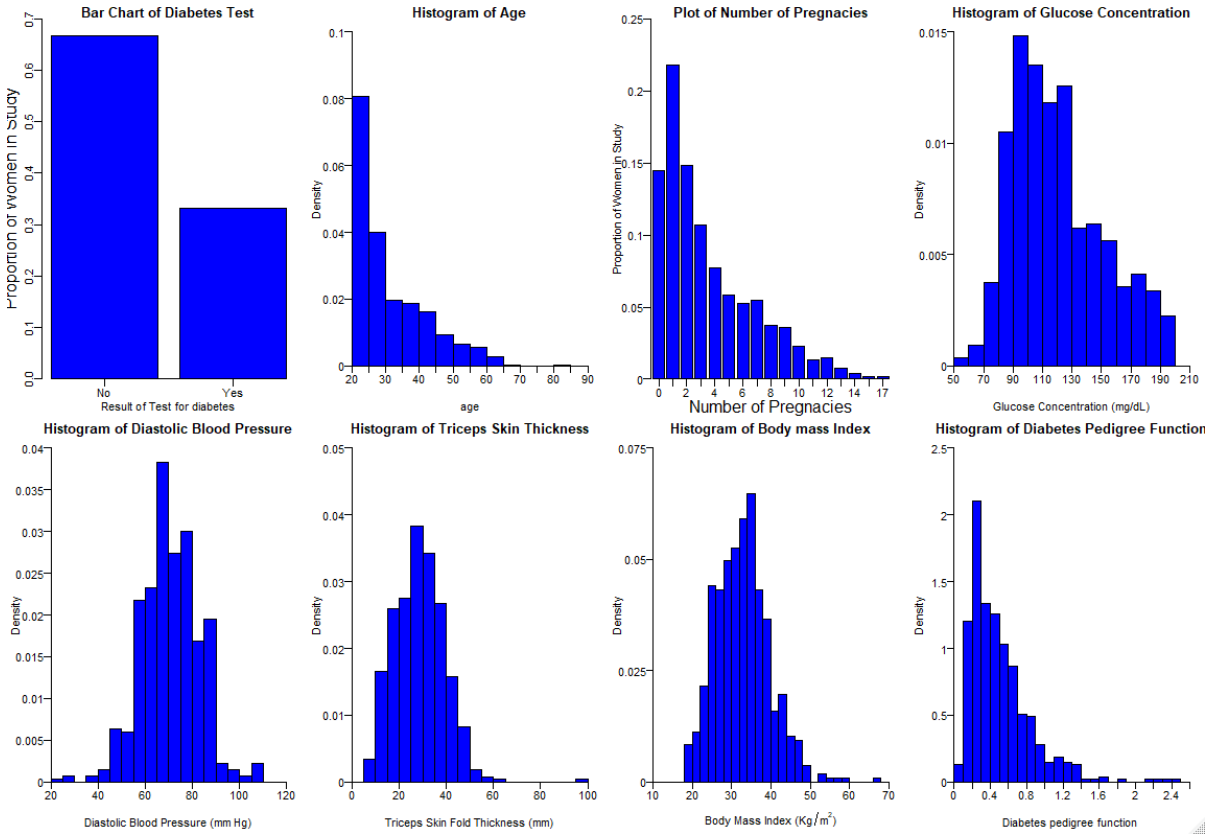


Figure 1: Initial Bar charts and Histograms of Variables in Pima data set

Notice that ***npreg*** has been plotted as a bar chart. If we fit it as a histogram then strange things can happen. Think about what would happen if the bar widths were 1.5 for example, some bars would contain two numerical values (e.g. 0 & 1) whilst others would contain just 1 (e.g. 2). This would yield a strange plot such as the one below (Figure 2). Watch out for this and change the default settings if this happens (it can often be a problem is data are rounded, for example, and we will see more examples below). In this case, the code provided produces a bar chart (rather than a histogram) as the number of pregnancies only takes a limited number of integer values. In other situations, you have to experiment with the number of bars and where the bar ends should be.

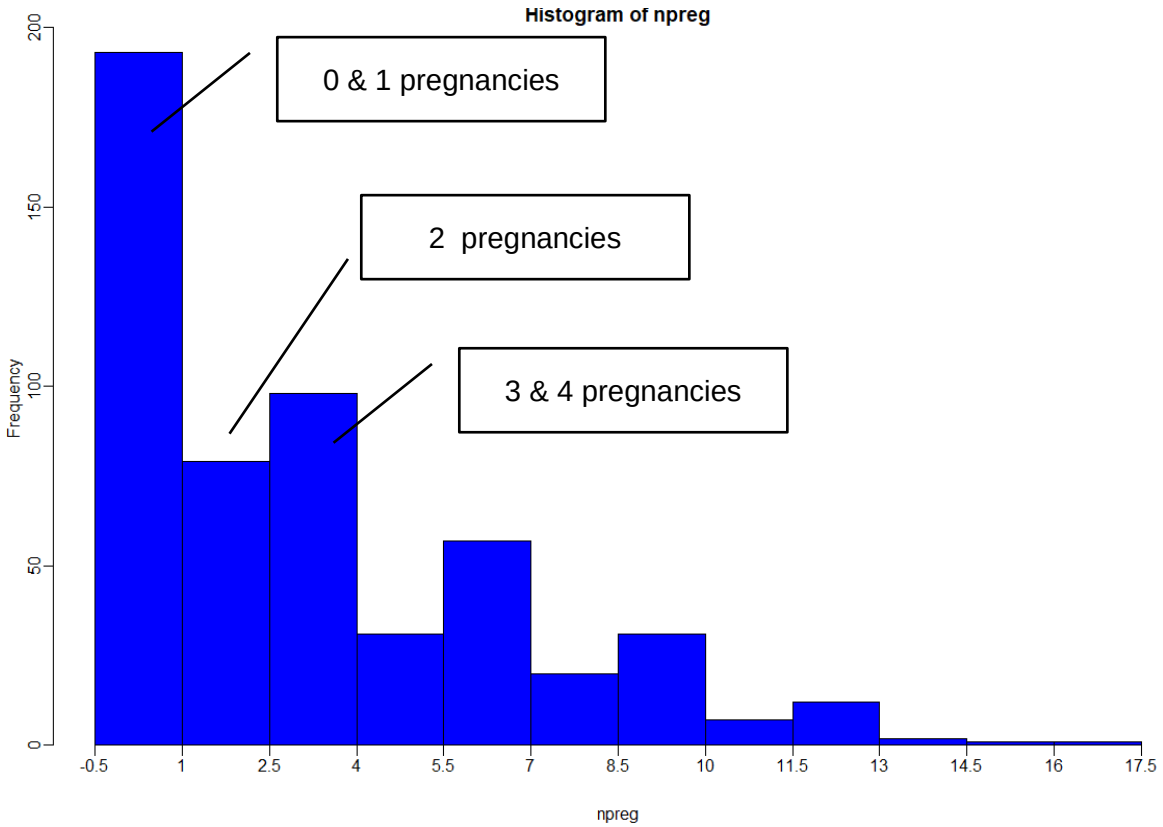


Figure 2: Histogram of Number of Pregnancies with inappropriate bar widths; results in a crenelated shape.

(If you are interested, I have included the code that produced this in BLOCK XIX)
A more appropriate plot is shown below:

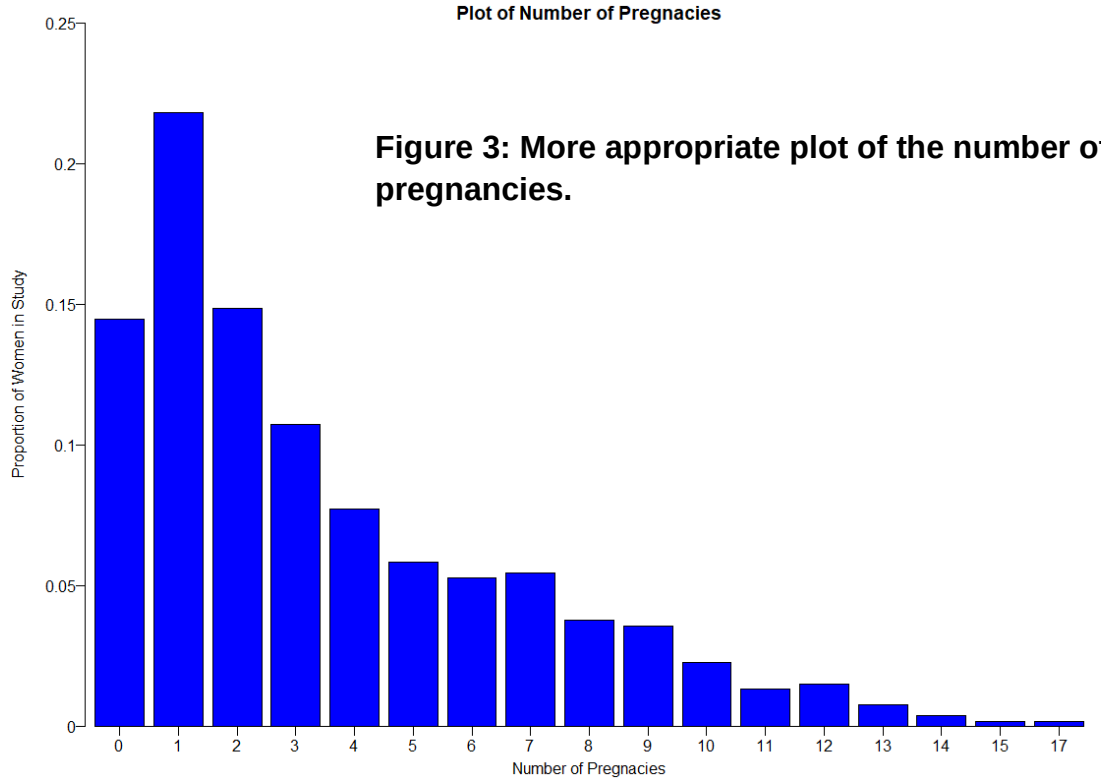


Figure 3: More appropriate plot of the number of pregnancies.

3.2 EXPLORING (UNIVARIATE) OUTLIERS

In the recent assignment there was a lot of confusion about the reason for the initial univariate EDA and why we might transform an individual variable at this early stage.

- These plots are telling us something about our data. Some variables are skewed,
  - But there is no requirement for them to be normal (or symmetrical) in our model (if you think about dummy variables in a linear model, they are not normal).
  - Even a dependent variable in a linear model may not appear normal because it is a mixture of normals (each with a different mean, depending on the values of the explanatory variables.) i.e.: $y_i \sim N(\mu_i, \sigma^2);\ i = 1, \dots, n$ note that μ_i is different for each y_i; $\mu_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i}\ ;\ i = 1, \dots, n$
  - A similar thing applies to the distribution for a generalised linear model; the independent (regressors) variables do not have to be normal and the dependent variable (response) follows a distribution (for logistic that is a binomial distribution) but with the location parameter dependent on the explanatory variables. Here $y_i \sim B(n_i, p_i);\ \eta_i = log\left(\frac{p_i}{1-p_i}\right)\ ; p_i = \frac{e^{\eta_i}}{1+e^{\eta_i}} = \frac{1}{1+e^{-\eta_i}}\ ;$
    $$\eta_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i}\ ,\qquad i = 1, \dots, n$$
- <u>For a linear model we need to check the linearity and homoscedasticity assumption by producing scatter plots; it is these that should form the basis for whether a transformation is needed or not</u>.
- <u>For a generalised linear model we need to find an equivalent means to check linearity</u>.
- We might transform individual variables for other techniques, for example prior to cluster analysis or prior to fitting a neural network.
- In this context (fitting a linear model) we would NOT generally transform on the basis of skewness prior to fitting the model.
- However, investigating transformations prior to modelling to make univariate distributions symmetrical can help you identify outliers.
  - For example, are the extreme values of age or diabetes pedigree function just part of the general skewed shape or are they actually outliers?

4) Run the code in Block IV (line by line); this plots a variety of transformations of ***age***, working down the ladder of transformations (given it was positively skewed). Eventually the reciprocal square seems to give something that looks symmetrical. A few histograms are repeated as there are some tall bars indicating we have the same problem as in Figure 2, so the end points of the histogram have been specified. (This took a bit of trial and error). Finally, the histogram is re-drawn with lines indicating the mean, and $\pm 2sd$. The histogram fits within the two dotted lines, so there are no outliers with respect to ***age***.

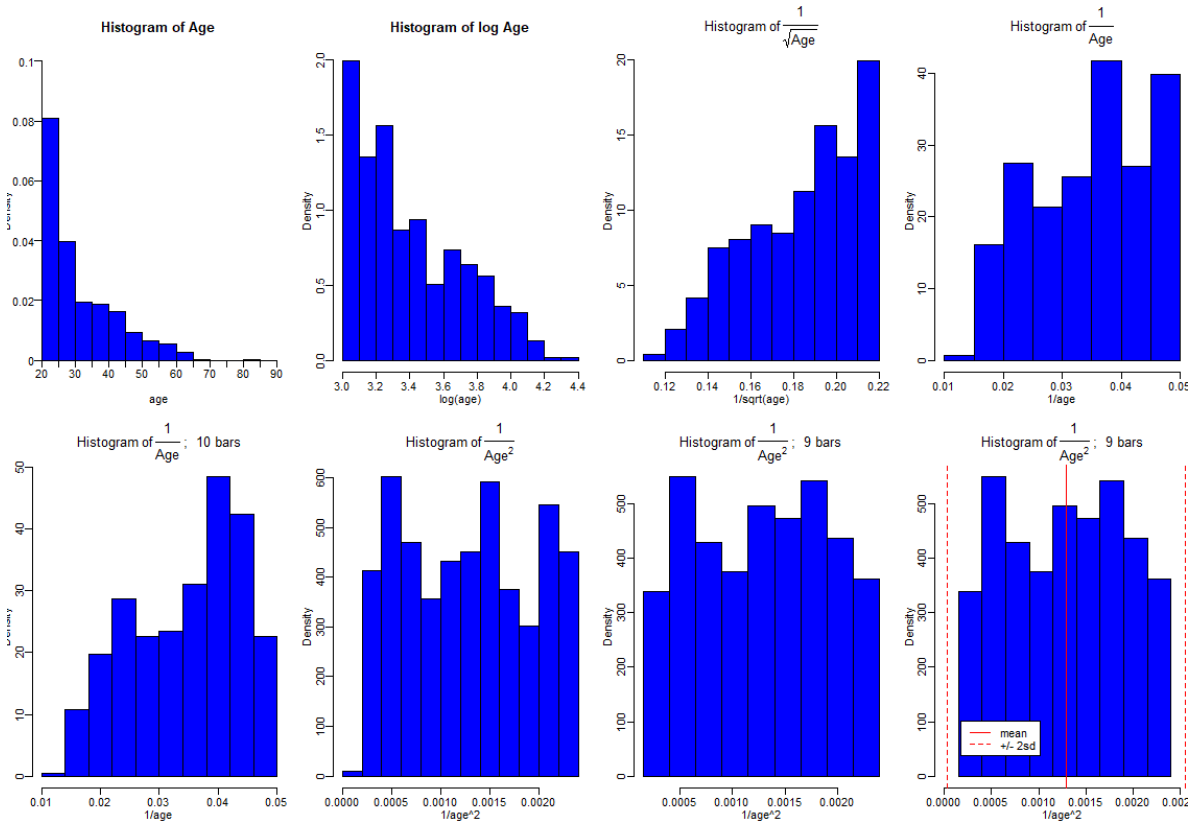


Figure 4: Exploring Transformations of Age, $\frac{1}{Age^2}$ seems most appropriate but bar widths adjusted to prevent uneven categories.

5) Run the code in Block V (line by line). The process above is repeated for pedigree function, but now the log transformation is sufficient. This time some of this histogram falls outside $\pm 2sd$ so the $\pm 3sd$ lines are also included. No points fall outside of these. Remember we would expect about 5% outside $\pm 2sd$ and about 0.25% outside $\pm 3sd$ which for our data of 532 observations is about 27 observations and 1 or 2 observations respectively. So as the histogram sits within this, it would suggest there are no outliers for pedigree function.

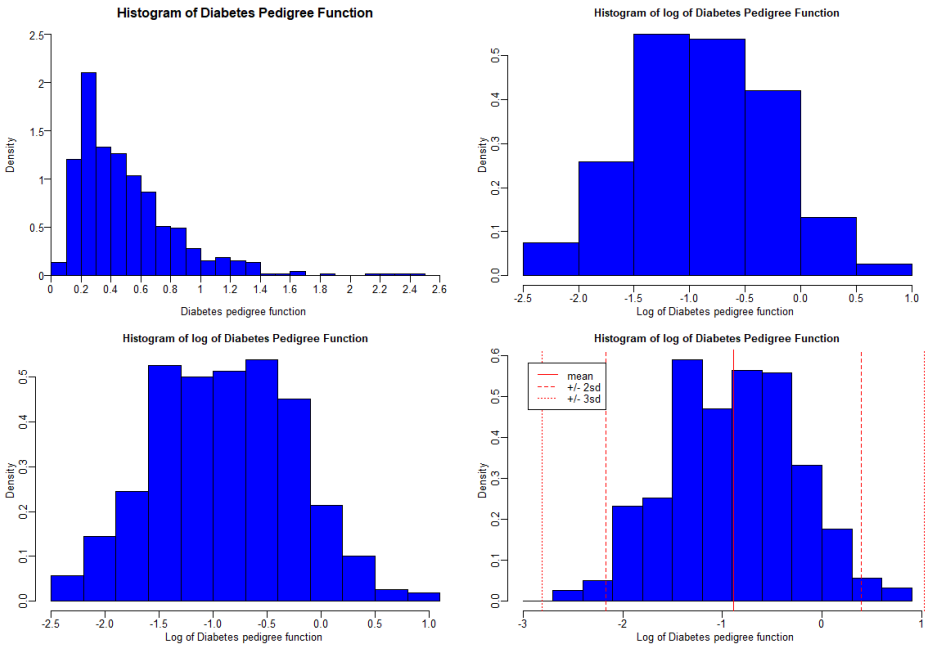


Figure 5: Transformation of Diabetes Pedigree Function showing that once transformed distribution within +/- 3 standard deviations of mean.

3.3 UNIVARIATE BOX-PLOTS

Rather than draw lines on a histogram we could instead check for outliers by producing some box plots of the transformed variables. Also, we can draw box-plots of the other variables where we have not transformed.

6) Run BLOCK VI which will produce some univariate box-plots. Note that this code uses `Boxplot` rather than `boxplot`.

7)

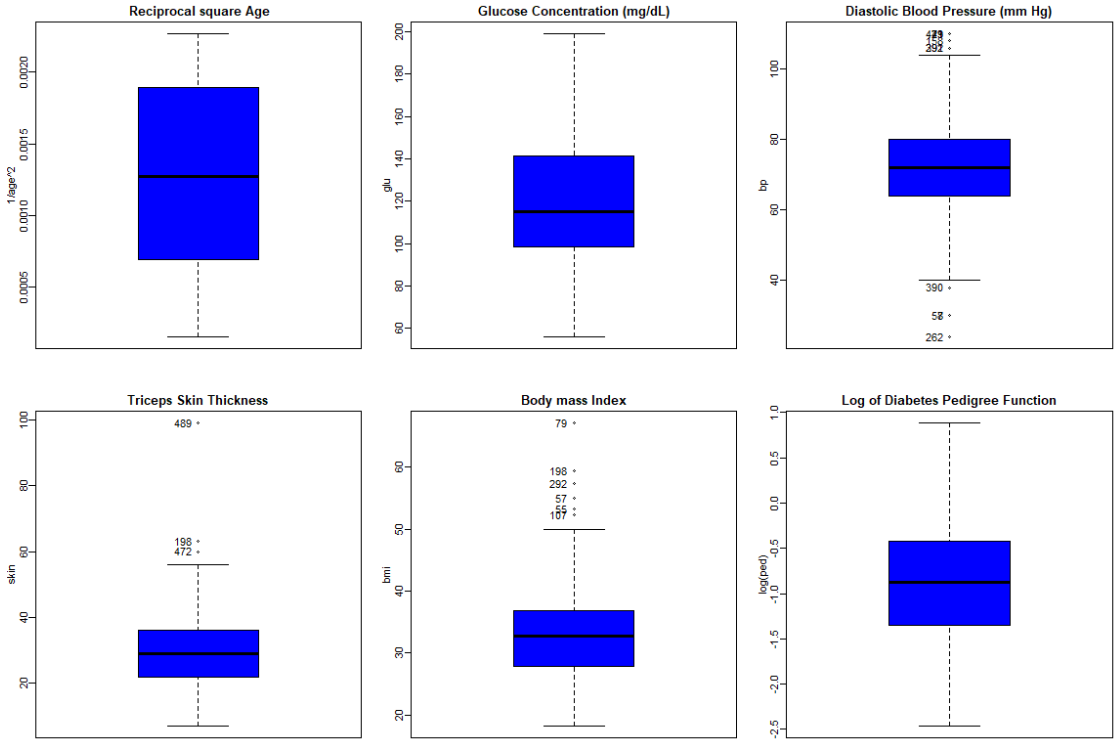


Figure 6: Box Plots of continuous variables where Age and Diabetes Pedigree Function are Transformed. Points outside the whiskers are labelled and these may be potential outliers.

The one for ***age*** and ***ped***, both show no outliers. However, we also produce them for the variables that looked more-or-less symmetrical but had potential outliers. We find that Triceps Skin fold thickness (***skin***), Body mass Index (***bmi***)and Diastolic Blood pressure (***bp***) have some outliers. If we use `Boxplot` (from the `car` package) rather than `boxplot` (note capital "B") these are labelled with their observation number. Potential outliers are observation 489, 198 and 472 for skinfold thickness (the first looks very extreme); 79,198,292,57,55 and 107 for body mass index. These do not look quite as extreme as 489 did in the previous box plot. Finally there are some for blood pressure, but they are only just outside the whiskers and given the size of the data set are probably not a cause of concern.

3.4 Conclusions of Univariate Analysis

- Some variables are skewed,
    - this may not be an issue at all.
    - If we need to transform these variables knowing they are skewed may give us a clue as to what transformations to consider, BUT for now we ignore this and consider if we need a transformation to make the linear term in our model linear (see below).
- We have some potential outliers, but the only really large ones seem to be 489 for Skin thickness and 79 for body mass index.
    - We should check these prior to modelling to make sure they are correct.
    - Neither of these seem so large that we should omit them at this stage.
    - Once we have fitted our model we should again check for outliers (which will be in a more multi-variate sense). If the same observations are flagged, we will already have identified the potential variable responsible.

3.5 Some Practical Tips

In this exercise the code has been extensive. If I were doing this and not having to demonstrate how this worked I may not even have considered transforming.

- With practice I can see that the extreme values in the skewed distributions are in keeping with the shape of these distributions.
- I would note the extreme values in e.g. skin thickness. But note that none of the extreme values are so large that they need to be removed at this stage.
- I might not adjust the bar widths to avoid the crenelations if they occur, unless I were writing this up as an assignment, a paper or some other published report. For my own quick view I would be happy to put up with it.

So for an analysis with a quick view I would probably only have produced Figure 1 on page 6! The rest is here as feedback and to demonstrate how we might proceed if we were unsure about outliers or preparing the data for some procedure that does require symmetry. (Which is NOT linear models!)

4 Initial Plotting: Bivariate Plots

4.1 Box Plots

8) Run BLOCK *VII* which will produce some plots of the explanatory variables as box plots, plotted against the binary dependent variable. Notice that there does appear to be some difference in the overall distribution of each variable for those diagnosed as being diabetic compared to those who are not. Generally, in each case the box-plot is higher for those who are diabetic, the largest difference is for ***glu*** (glucose concentration). Notice that the range shown by the plots does still overlap. So, knowing the value of the explanatory variable can help identify those which are diabetic, but it is not perfect. However, perhaps knowing a combination of these might be useful. The second set of plots repeats this but uses `Boxplot` (from the `car` package) this time rather than `boxplot`. This now labels the points outside of the whiskers.

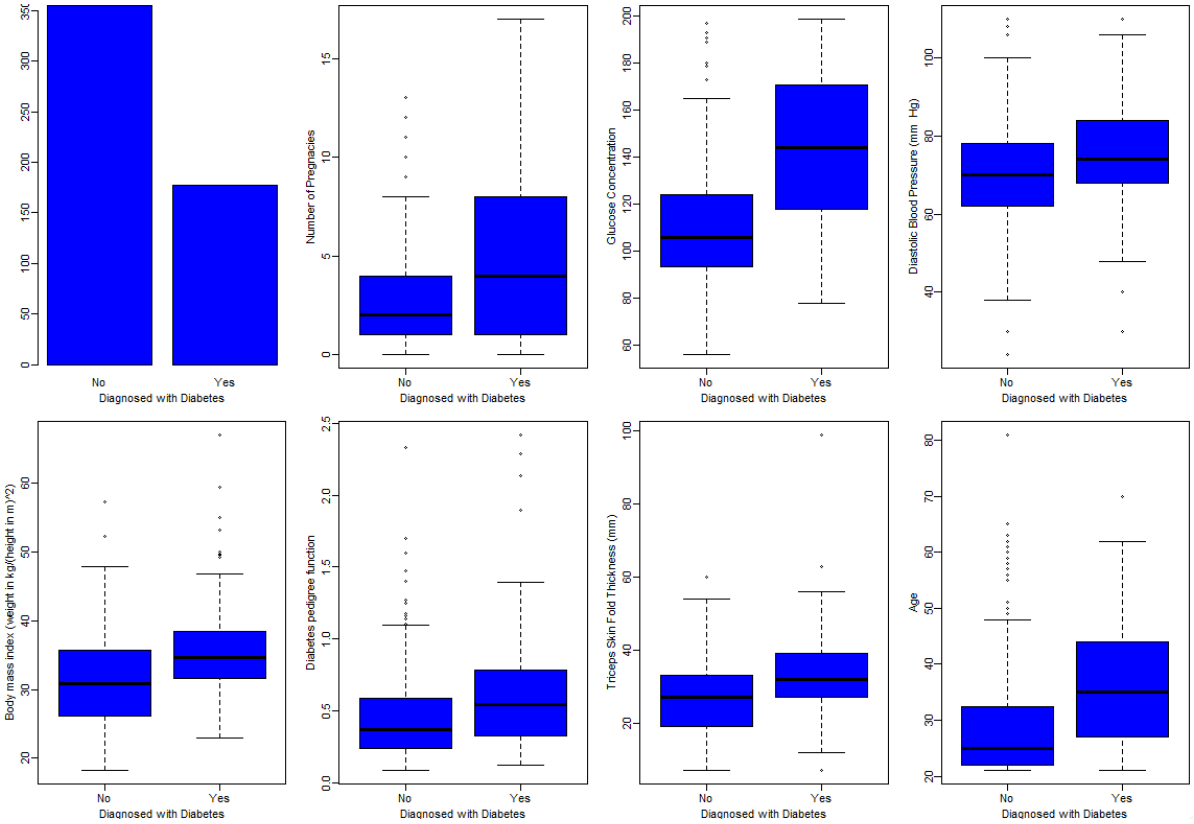


Figure 7: Box Plots against *type* showing that most variables have higher values on average for individuals who are diabetic.

Whilst these plots show some relationship, perhaps more useful are plots that will enable us to see if the linear predictor term in the logistic regression model is in fact linear.

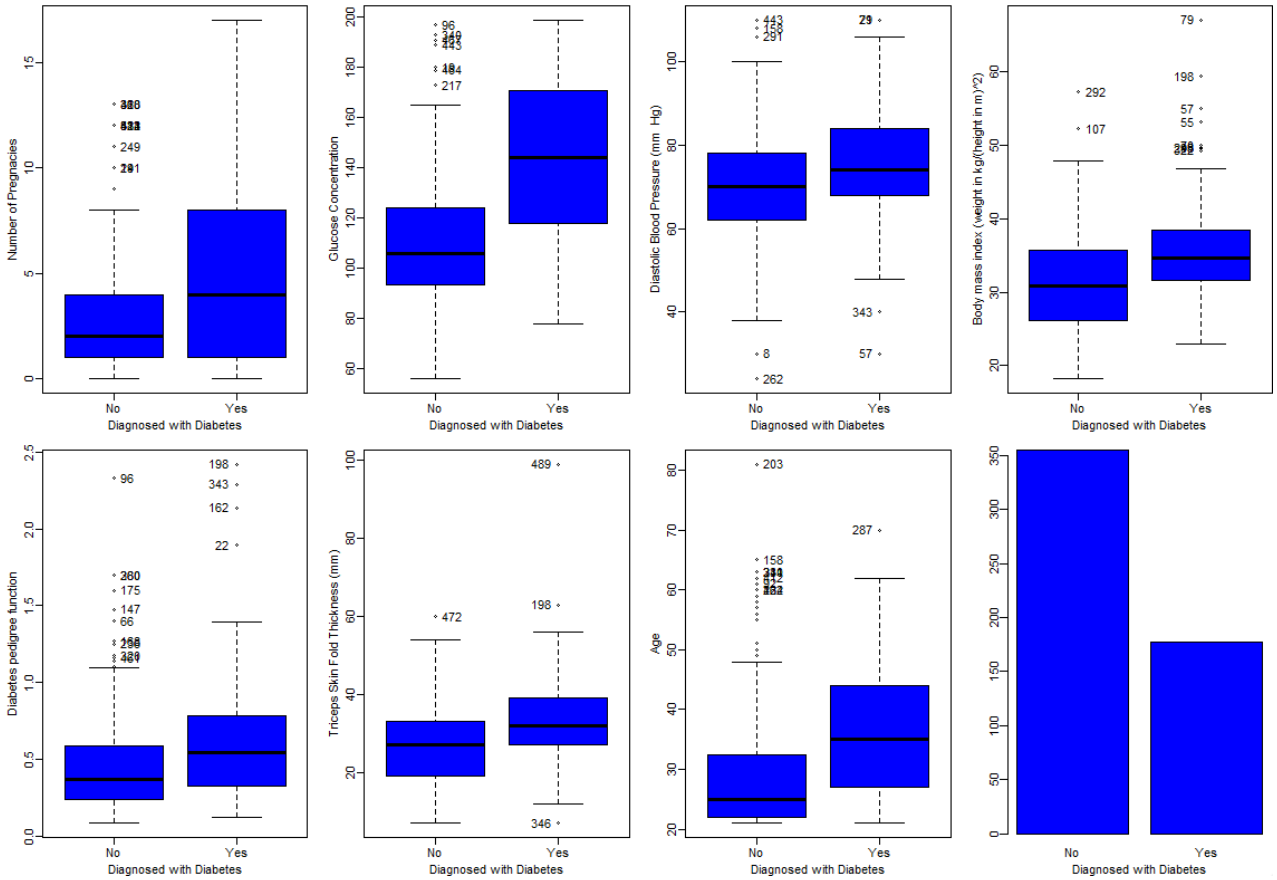


Figure 8: Box Plots against *type* showing that most variables have higher values on average for individuals who are diabetic but with outliers labelled.

4.2 Grid of Scatter Plots

Some of the above plots can be summarised using for example the `ggpairs` plot. This will re-do the box-plots against the dependent variable, but in addition it will produce scatter plots for each of the variables against each other, produce correlations and produce the density (in the diagonals) and histograms (bottom row) of each variable for each diabetes ***type.***

9) Load and run the code in **ggpairsPima.R**

10) Run the code in BLOCK $VIII$.

   a) What do these plots show?

   b) Are there any new issues that you can spot?

Much of the information shown is the same as above. We can see the skewed plots for ***age*** and ***ped*** again, but notice they are not as skewed for the diabetic group.

Also, we can see that some of the explanatory variables are highly correlated with each other. In particular ***skin*** and ***bmi*** have a correlation of 0.647, which is not surprising as the body mass index and triceps skinfold thickness are both measures of being overweight. Similarly ***age*** and ***npreg*** are highly correlated. Clearly to have had a large number of pregnancies a woman has to be older. We should be careful to look for models that perhaps do not include both variables from these pairs. However, we will let the model selection process pick the best variables.

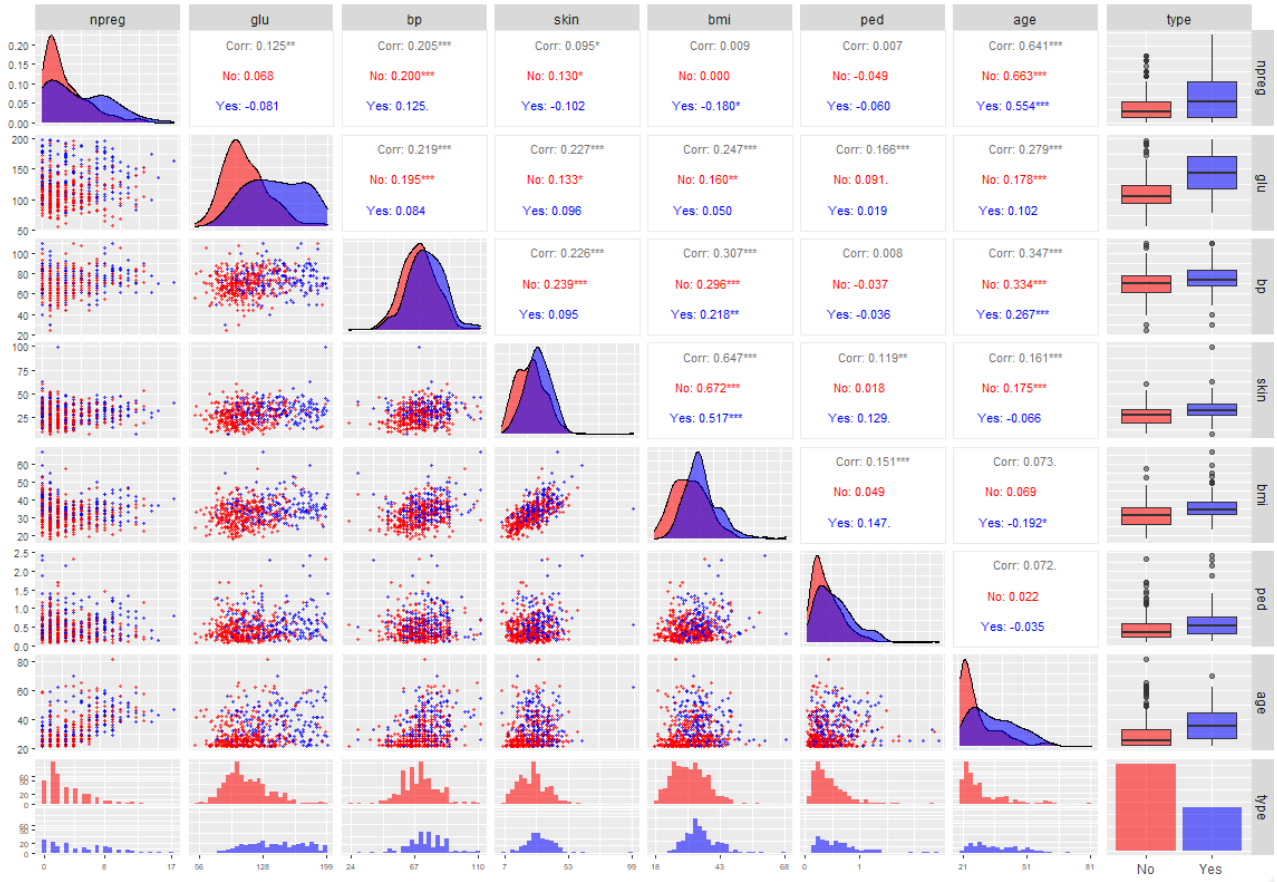


Figure 9: `ggpairs` plot of Pima Data

4.3 Empirical Logit Plots

For data already in bins (i.e. grouped data) the type of plot we are going to produce is straightforward. For single observation data where each row corresponds to one individual this is not so straightforward. In this situation it is necessary to put the data into "bins" e.g. 0-10,10-20 etc. The package, **`Stat2Data`** contains a function to produce these, but it requires us to identify the breaks at which we want the data binned; ideally these should be evenly spaced. Consequently, I have supplied you with a function `myemplogit` that will do this for you and then call the function `emplogitplot1` within the **`Stat2Data`** package. This function is in BLOCK IX. It is a slightly different function seen in the frogs example (last video produced by Professor Plummer). These are the differences:

- It plots bubbles to depict the size of the bin (i.e. the number of observations being used to produce each point).
- It does not join the points up (so that perhaps when there are many point plotted it easier to see the "shape" the points make).
- It adds a regression line by default so this can provide a bench mark to see if the relationship of the regressor to the logit of the dependent variable is linear.

It does not really matter which version of this function you use, but you should produce this plot if you are going to fit a logistic regression model. It is equivalent to producing the standard scatter plots when you fit a normal linear model.

4.3.1 An Explanation of the function provided (optional reading)

Within the code for `myemplogit`, the `quantile` function divides the data into as equal sized bins as possible, although the result is not always quite what we wish for. `quantile` requires the probabilities that the quantiles should be calculated for. In our case we create evenly spaced probabilities. For example suppose we want 14 bins then quantiles are required at 0/14, 1/14, 2/14,…, 14/14; in R code this would be `0:14/14`. The function `myemplogit` takes in the argument `maxbins` and finds the quantiles for `0:maxbins/maxbins` (rather than 14). If two quantiles turn out to be the same (e.g. if we ask for $k^{th}/20$ quantiles and over 5% of the data has the same value – the variable `pregnant`, is one such example see below.) then this will result in an error. The `unique` function is used to remove any duplicates:

```
> breaks<<- unique(quantile(xvar, probs=0:maxbins/maxbins))
```

When this is called for e.g. the `npreg` variable with `maxbins` = 30 `breaks` contains:

```
[1]  0  1  2  3  4  5  6  7  8  9 11 17
```

Notice that although we requested 30 bins we do not have sufficient unique values in a variable to split it into that many and consequently `breaks` contains fewer values. In the pregnant example it only contains 12 values. The resulting values in `breaks` are later passed to the `emplogitplot1` function. Before this, `cut` is used to create the range for each bin (see for example (endmemo.com 2016). By default, `cut` removes the lowest range, which we do in fact want but if included, it seems to create the first bin as the two smallest categories together. Consequently, in my function I have left it as the default (to not include the lowest range). As a result, the `cut` function creates a bin containing the minimum values classes as <NA> as the last row. The count function is used to create the frequency for each bin. As an example, the resulting table (`c.tab`) for the variable `pregnant` is as follows:

	levs	freq
1	(0,1]	93
2	(1,2]	64
3	(2,3]	45
4	(3,4]	27
5	(4,5]	21
6	(5,6]	19
7	(6,7]	20
8	(7,8]	14
9	(8,9]	11
10	(9,11]	11
11	(11,17]	11
12	<NA>	56

Figure 10: Result of `cut` function

The last row is then replaced with the label (e.g. in this case [0]) and this is moved to the first row:

```
c.tab$levs <- factor(c.tab$levs,levels = levels(addNA(c.tab$levs)),
labels = c(levels(c.tab$levs),paste("[",min(xvar),"]",sep="")),
exclude = NULL)
c.tab <- c.tab[c(nrow(c.tab),1:nrow(c.tab)-1),]
```

Resulting in the correct table (for `pregnant`):

```
> c.tab
```

```
       levs freq
12      [0]   56
1     (0,1]   93
2     (1,2]   64
3     (2,3]   45
4     (3,4]   27
5     (4,5]   21
6     (5,6]   19
7     (6,7]   20
8     (7,8]   14
9     (8,9]   11
10   (9,11]   11
11  (11,17]   11
```

Figure 11: Reconfigured table

The frequency of each bin in this table is used to work out the relative bubble size of the points in the plot.

Notice the use of the "superassignment" operator (`<<-`) in places throughout this function; this causes results, for example `num,` to still exist once the function has been executed. It is used here for `num` because the call to the `count` function will not recognise `num` as existing; `count` is a function outside of the `myemplogit` function. In fact `count`uses the **`plyr`** package to count the number of observations in each of the `levels` created. This will be printed out when the function `myemplogit` is called. (e.g. resulting in the output above). Similarly, the `emplogitplot1` function will not recognise the variable `breaks` unless it is "super-assigned".
Below the function are the calls to the function that will create the plots. (BLOCK IX)

4.3.2 Using the `myemplogit` (essential reading if you wish to use this code)

If was found that the function `emplogitplot1` within `myemplogit` produced some errors for some variable/`maxbins` combinations (it was usually an error about length of variables – which will be something to do with where the bin breaks are). Also, sometimes the bubbles were too big or too small. As a consequence of the latter, the `sc` variable was added to the function to enable adjustment of the size of the points in the resulting plots. You should therefore use some trial and error in using this function:

`maxbins:` you want sufficient bins to see some details of the pattern of points but not too many bins; otherwise the bin is too small to be representative of that data group. If you get an error such as that suggested above, try requesting fewer bins.

`sc:` if the bubbles are too big (if the plot is all black they are very very big!) reduce this, if too small increase this. Perhaps at a later stage I will modify this function to be less sensitive.

4.3.3 Recap on what this function does (recommended reading)

The function takes a variable that is in single observation form, bins it into `maxbins` bins and then plots the empirical logit against the mid-point of the bin range. e.g. for the Pima Indian data we have each row of the data being one individual. We then count the number of individuals with and without diabetes in a bin for a particular bin. For example, for the variable ***age*** the diagram on the next page (Figure 12) shows a selection of the individual data and then shows the bins and the numbers with and without diabetes in each age bin.

The total numbers in each age group (n) can be found by adding the two resulting columns (called y and n-y respectively in Figure 12) together.

The estimate of the logit for each age group (bin) would be:

$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = log\left(\frac{{}^{y}/_{n}}{1-{}^{y}/_{n}}\right) = log\left(\frac{y}{n-y}\right)$$

However, to avoid log(0) or a divide by 0 in the quotient we use:

$$log\left(\frac{y+0.5}{n-y+0.5}\right)$$

This is then plotted against the mid-point of age. For the example the calculations are shown in Figure 13. In my function the bubble size represents n, the number of observations in the bin. E.g. for age 21 the bin size is 31+2= 33 whilst for (37,39] the bin size is 7+3 = 10. The (37,39] bin will therefore be plotted with a smaller dot (bubble) than for age group 21.

Age	type
21	0
33	1
26	1
53	1
59	1
51	1
31	1
33	0
32	1
27	0
51	1
41	1
22	0
57	0
28	1
22	0
33	0
.	.

Etc.

26	1
37	1
22	0
63	0
30	0

Age group (x)	Number without diabetes (n-y)	Number with diabetes (y)
[21]	31	2
(21,22]	38	5
(22,23]	25	3
(23,24]	25	6
(24,25]	21	9
(25,26]	19	5
(26,27]	11	3
(27,28]	14	7
(28,29]	6	8
(29,30.6]	7	3
(30.6,32]	8	10
(32,33]	4	7
(33,35]	9	5
(35,37]	8	7
(37,39]	7	3
(39,41]	5	6
(41,43]	6	10
(43,46]	4	7
(46,51]	7	9
(51,55]	1	8
(55,81]	6	7

Figure 12: Illustration of binning age: individual data is on the left, binned data on the right.

Age (x)	Not Diabetic (n-y)	Diabetic (y)	Diabetic+0.5 y+0.5	Not+0.5 n-y+0.5	"Empirical" probability $\frac{y+0.5}{n-y+0.5}$	Emperical Logit $log\left(\frac{y+0.5}{n-y+0.5}\right)$
[21]	31	2	2.5	31.5	0.07936508	-2.5336968
(21,22]	38	5	5.5	38.5	0.14285714	-1.9459101
(22,23]	25	3	3.5	25.5	0.13725490	-1.9859155
(23,24]	25	6	6.5	25.5	0.25490196	-1.3668763
(24,25]	21	9	9.5	21.5	0.44186047	-0.8167611
(25,26]	19	5	5.5	19.5	0.28205128	-1.2656664
(26,27]	11	3	3.5	11.5	0.30434783	-1.1895841
(27,28]	14	7	7.5	14.5	0.51724138	-0.6592456
(28,29]	6	8	8.5	6.5	1.30769231	0.2682640
(29,30.6]	7	3	3.5	7.5	0.46666667	-0.7621401
(30.6,32]	8	10	10.5	8.5	1.23529412	0.2113091
(32,33]	4	7	7.5	4.5	1.66666667	0.5108256
(33,35]	9	5	5.5	9.5	0.57894737	-0.5465437
(35,37]	8	7	7.5	8.5	0.88235294	-0.1251631
(37,39]	7	3	3.5	7.5	0.46666667	-0.7621401
(39,41]	5	6	6.5	5.5	1.18181818	0.1670541
(41,43]	6	10	10.5	6.5	1.61538462	0.4795731
(43,46]	4	7	7.5	4.5	1.66666667	0.5108256
(46,51]	7	9	9.5	7.5	1.26666667	0.2363888
(51,55]	1	8	8.5	1.5	5.66666667	1.7346011
(55,81]	6	7	7.5	6.5	1.15384615	0.1431008

Figure 13: Table showing calculation of empirical logit for age variable

11) Run the code (BLOCK IX) for the `myemplogit` function, highlight the full function including the closing curly bracket. This will load the function.

12) Now run each line in BLOCK X, first reset the graphic area to fit just one plot in it and then each line that calls the `myemplogit` function (). This will produce a binned empirical logit of ***type*** against each explanatory variable in the data.
   a) Do you think any are suitable for inclusion in our model?
   b) Does the plot look linear? (The interpretation is essentially the same as for a scatter plot when we do normal regression).

If we wish to transform a variable here, the only option would be to transform the explanatory variables. The dependent variable is tied in with the logistic function. Other options are to include additional terns such as the squared term, or to bin the variable and treat it as a factor.

13) Experiment and reproduce these plots with a different number of bins (you may need to change the scale factor as well).

14) Try plotting these plots with and without the regression line. (Option is `line =FALSE/TRUE`)

15) To produce the plots on one page run the code in BLOCK XI (which sets the plotting area into a grid) and then go back and run the code in BLOCK X without the first line of code (which resets the grid back to 1×1).

16) Try some transformations for any variables that do not look linear.

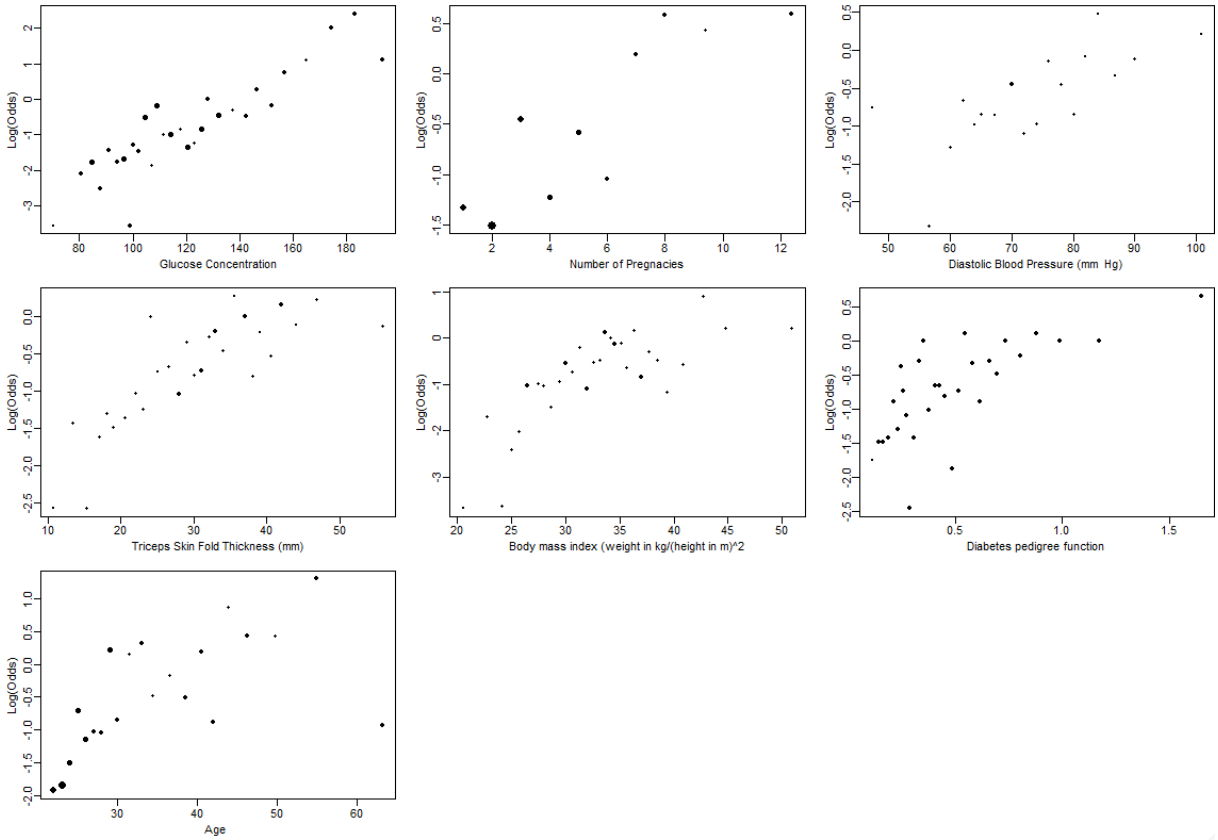


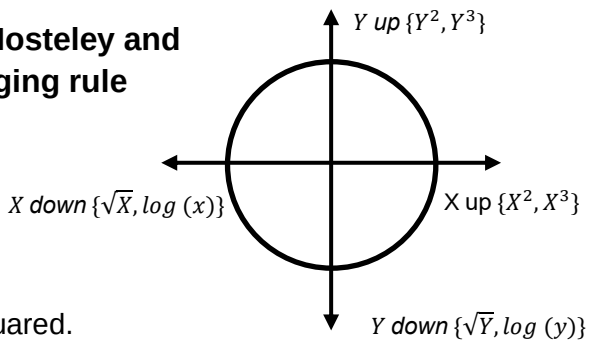
Figure 14: Initial Empirical Logit Plots

Most of the plots look approximately linear. However, two that do not quite look linear are the body mass index (***bmi***) and the diabetes pedigree function (***ped***). Possibly Triceps skin fold thickness (***skin***) is non-linear but it is hard to tell if the point on the far right and the two points on the bottom are just a bit low. ***Age*** also either has a large outlier on the lower right-hand side or needs a quadratic term. Remember that although this may be due to just one point, the point itself is made up of several individuals. The point is not particularly small so it might be worth trying a squared term.

5 Transformations

We tried some transformations earlier but remember there is no requirement that the variables be symmetrical. In finding transformations to fit a linear model we need the relationship to be linear. For a logistic model we need the logit of the predictor to be linear with respect to the explanatory variables. From the plots above we need to try some transformations of three variables, ***bmi***, ***ped*** and ***skin***. We can make use here of the bulging rule seen in lectures:

Figure 15: Mosteley and Tukeys' bulging rule



Our plots all curve up and we cannot transform y. Hence, we should try transformations going down the ladder of x. We will start with log and then try reciprocal. If log is too strong, we can try square root. If reciprocal is not strong enough, we can try reciprocal squared.

5.1 Transforming the Explanatory Variables

17) Run the code in BLOCK *XII* This will reproduce the original plot of each of the three variables we are going to transform and in addition try both the log and reciprocal transformation. Notice the `sc=` parameter in these is changed to make the dots bigger.

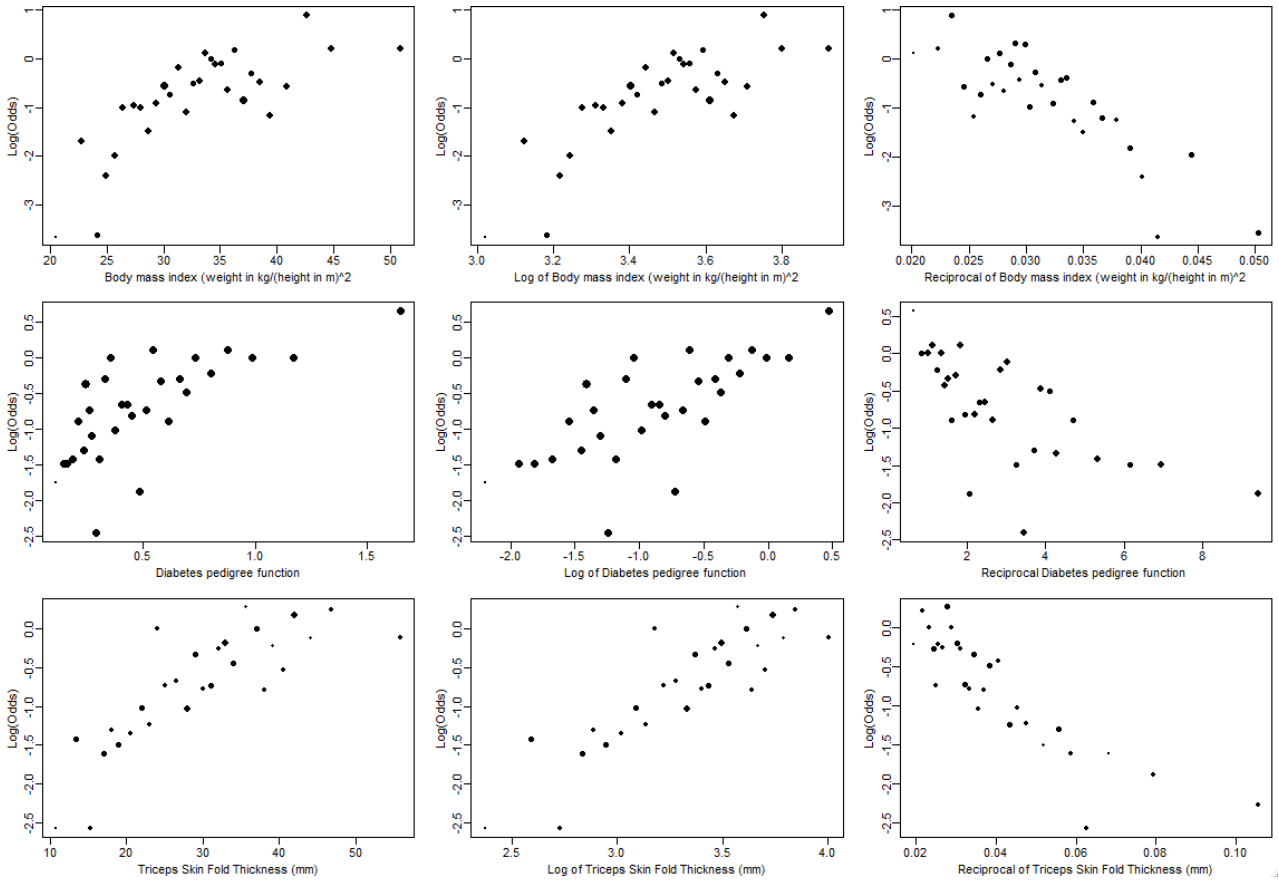


Figure 16: Transformations of bmi(top row), ped (middle row) and skin (bottom row): original (left column), log(centre) and reciprocal (right)

The log transformation does seem to improve all three variables but for ***bmi*** the reciprocal seems to be a further improvement. For both ***ped*** and ***skin***, the log transformation seems to be the best. For ***ped***, there are still two points lower than the rest.

6 Model Fitting

6.1 Full Model

18) Now run the next section of code in block *XIII*, this will fit the null model with only the intercept (`fit0`), full model (`fit1`) which includes all terms without any transformation. Then it fits a model using 1/***bmi***, log(***skin***), log(***ped***) and a quadratic in ***age*** (`fit2`). Use the `summary` and the `Anova` functions to examine the results.

a) Does it appear all terms are significant? Can any be removed?

b) Which of the two models do you prefer?

You should notice that the model using the original variables has an AIC of 482.32 whilst the one with transformed variables has an AIC of 464.43, suggesting this is the preferred model. We could of course try swapping the untransformed variable for transformed variables and see if some different combinations can improve things. However, given the plots we will examine the output from the second model.

```
Call:
glm(formula = type ~ npreg + glu + bp + I(1/bmi) + log(ped) +
    poly(age, 2) + log(skin), family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7238  -0.6280  -0.3186   0.6155   2.5005

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -1.225069   2.264870  -0.541 0.588576
npreg           0.060191   0.046769   1.287 0.198103
glu             0.035256   0.004313   8.175 2.95e-16 ***
bp             -0.010408   0.010429  -0.998 0.318276
I(1/bmi)      -99.923076  27.996866  -3.569 0.000358 ***
log(ped)        0.740269   0.195822   3.780 0.000157 ***
poly(age, 2)1  11.227412   3.816141   2.942 0.003260 **
poly(age, 2)2 -10.721150   3.305776  -3.243 0.001182 **
log(skin)       0.063285   0.420784   0.150 0.880451
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 676.79  on 531  degrees of freedom
Residual deviance: 446.43  on 523  degrees of freedom
AIC: 464.43
```

Figure 17: Summary output from the Model fitted to all the variables with some transformed.

A global test (a bit like the F-test) can be found by taking the difference between the Null deviance and the residual deviance to give: `676.79 - 446.43 = 230.36` on 8 degrees of freedom (either count the number of terms other than the intercept or note that `531-523=8`). We therefore compare 230.36 to a χ^2_8 we can either use the `pchisq` function to find the p-value: `pchisq(230.36,8,lower.tail=FALSE)` which gives 2.48×10^{-45} which is virtually 0. Alternative we could use the `anova` function to compare the models `fit0` with `fit1`.:

```
> anova(fit0,fit2,test="Chisq")
Analysis of Deviance Table

Model 1: type ~ (1)
Model 2: type ~ npreg + glu + bp + I(1/bmi) + log(ped) + poly(age, 2) + log(skin)
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1       531     676.79
2       523     446.43  8   230.36 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 18: Deviance test to compare full model to null model (equivalent to the F-test in normal regression)

Either way the test is highly significant. We reject the null hypothesis that all the parameters for the variables are zero. This suggests we need at least some of these variables in the model.

6.2 Model Reduction

Returning to the `summary` output we notice that not all terms are significant. The term with the largest p-value is `log(skin)`, so we could fit a model with that removed. However, before we do so, we should first check whether ***age*** is required. In the `summary` output ***age*** is entered as two terns, a linear term and a quadratic. If we want to test simultaneously if both ***age*** terms are significant, we could do so by fitting the model without ***age*** and comparing the deviances as we just did for our model compared to the null model. However, if we use the `Anova` function this lists all such difference in deviance tests.

```
> Anova(fit2)
Analysis of Deviance Table (Type II tests)

Response: type
             LR Chisq Df Pr(>Chisq)
npreg           1.675  1  0.1956020
glu            84.009  1  < 2.2e-16 ***
bp              0.994  1  0.3188464
I(1/bmi)       13.544  1  0.0002330 ***
log(ped)       15.187  1  9.738e-05 ***
poly(age, 2)   16.952  2  0.0002084 ***
log(skin)       0.023  1  0.8803461
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 19: Analysis of Deviance table, change in deviance for each term if it is the last to be entered (or first to be removed).

This table shows the same p-values as in Figure 17, but now ***age*** appears as one item with 2 degrees of freedom. We notice that the ***age*** term is highly significant and this suggests we should not remove it. Instead, we proceed therefore to remove the term for ***skin***.

19) Continue removing the least significant term, using `summary()` and `Anova()` to check which terms are significant. The code is in BLOCK $XIII$.

    a) Finally check the model by carrying out a test using the `anova` function to compare the final model with the original `fit2`. You should have found that the change is not significant so we were OK to remove the terms that we did based on this. Also notice that the final model has the lowest AIC so far.

This used a backwards approach. Given the small number of variables this can be a suitable method to use. We could have automated this by using the `step` function, or we could use a LASSO. Another alternative is the grouped LASSO if we want to force both terms of ***age*** in, or both terms out together (i.e. treat both terns of ***age*** as one variable). This is certainly how we might proceed with a factor term. For a quadratic, we test to see if we need the squared term and only consider removing the linear term if the squared term has been removed. For the purpose of this example the method we have used should illustrate some of the ideas. We could also have used just the training data to fit this and validated it with the test data. (See later for an example of this.)

7 Residuals and Diagnostics

Many of the residual plots, added variable plots etc.can be carried out for logistic regression. The code will be the same in most cases. However, we mentioned that for single observation data this is not the case with our basic residual plot.

7.1 Binned Residual Plot

The code in the next block produces a grouped residual plot. This works on the same principle as the empirical logit plots in that the residuals are put into bins. By default, this adds the confidence interval for where the points should be, but this should really be some sort of smoothed interval and is susceptible to being too narrow if a section of the predictions are all under or all over predicted. However, if you change the option to leave out the confidence interval the plot can still be useful.

20) Run the code (in block XIV) to produce the Binned residual plot, experiment and try this with some other sorts of residuals and for a different number of bins. Examples are given already of plotting these against the predicted values and against each x-variable. Notice that we plot the residuals even against variables we have removed. This can be a good way to further check we were OK to remove them.

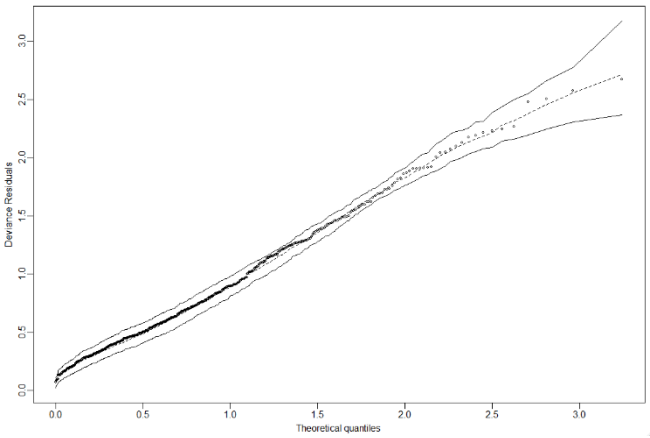
Most the plots seem OK (bear in mind these are all types of standardised residuals) but the plot of the deviance residuals or the student residuals against the fitted values does show some trend. This suggests some consistent over and under prediction at either end. Given the Pearson residuals seem fine it is not clear why this occurs in this case.

7.2 Half Normal Plot

There is no requirement that the residuals be normal, however it turns out that they are often approximately so. A good way to check if the model is adequate is to produce a half normal plot. This can often highlight outliers. We will use the function `hnp` available with the `hnp` package (de Andrade Moral, Hinde and Demetrio 2018). This produces a half normal plot which is a bit like a qqplot but is based on the idea of plotting the absolute residual against the absolute normal quantiles. The function in addition uses simulation to estimate a simulation envelope (rather like a confidence interval). If the model is adequate the points should fall in this envelope.

21) Run the code in BLOCK XV to produce the half normal plot of the Pearson and deviance residuals. Notice that the residuals fall within the simulated envelope and also lie on the dotted line, suggesting the model is adequate.

Figure 20: Half Normal Plot of Deviance Residuals



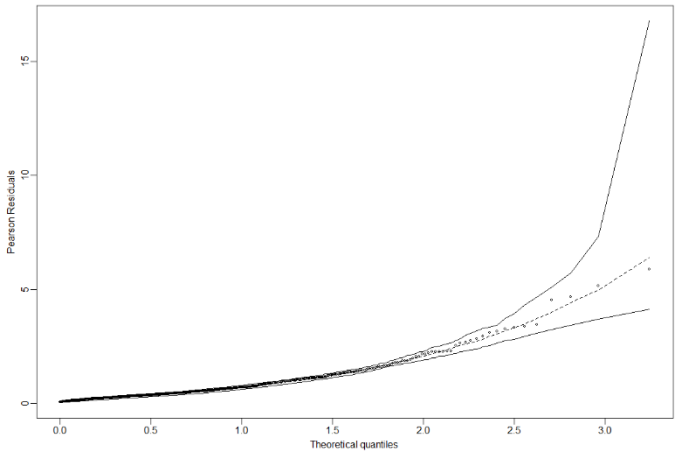


Figure 21: Half Normal Plot of Pearson Residuals

7.3 Other Diagnostic Plots

Many of the residual plots, added variable plots etc.can be carried out for logistic regression.

22) Run the code (in block *XVI*) to produce a variety of diagnostic plots:

a) The added variable plots are good ways to spot unusual observations. We run this on the full model `fit2` and then the reduced model `fit5`. The added variable plots, as with linear models, illustrate if there are any unusual observations. Observations 96 and 217 are spotted as potential issues.

b) The partial residual plots are a good way to check if the model specification is reasonable. In this case the pink line is reasonably close to the fitted line (blue), so it would seem a reasonable transformation has already taken place. The plots are dominated by the outliers so perhaps we should try refitting the models and reproducing the plots with these points removed. In this particular example it turns out that this does not really alter things that much. In these plots observation 96 is often flagged. Also notice observation 203 on the far left in the age squared plot. This might suggest that it is only down to this observation that we seemed to need the age^2 term (remember it was just one point in the residual plot). Perhaps we should try removing this point and seeing if age^2 is still required. In the partial residual or "component+residual" plot the pink line seems to match the blue line suggesting all is well.

c) Next, we draw the dfbetas, these are the change in the parameter estimate if an observation is removed, so there is a plot for every parameter in the model. There are two versions, one plots the raw dfbetas (`dfbetaPlots`), the other the standardised dfbetas (`dfbetasPlots`). In the latter we have requested that the largest three observations be

labelled, but none are outside +/- 1 so there are no changes of more than one standard error when an observation is removed.

d) Now, we produce the dffits, these show the change in the fitted values. We notice one high one. As way of illustration to identify this point we make use of the `identify()` function. Once you have produced the plot run the next line. Then in your plot with your mouse click on the high point and any others you want to identify (e.g. there is a low one on the right hand side). When finished click on the "finish" button as shown in Figure 22 . You should find that the high value was observation 287.

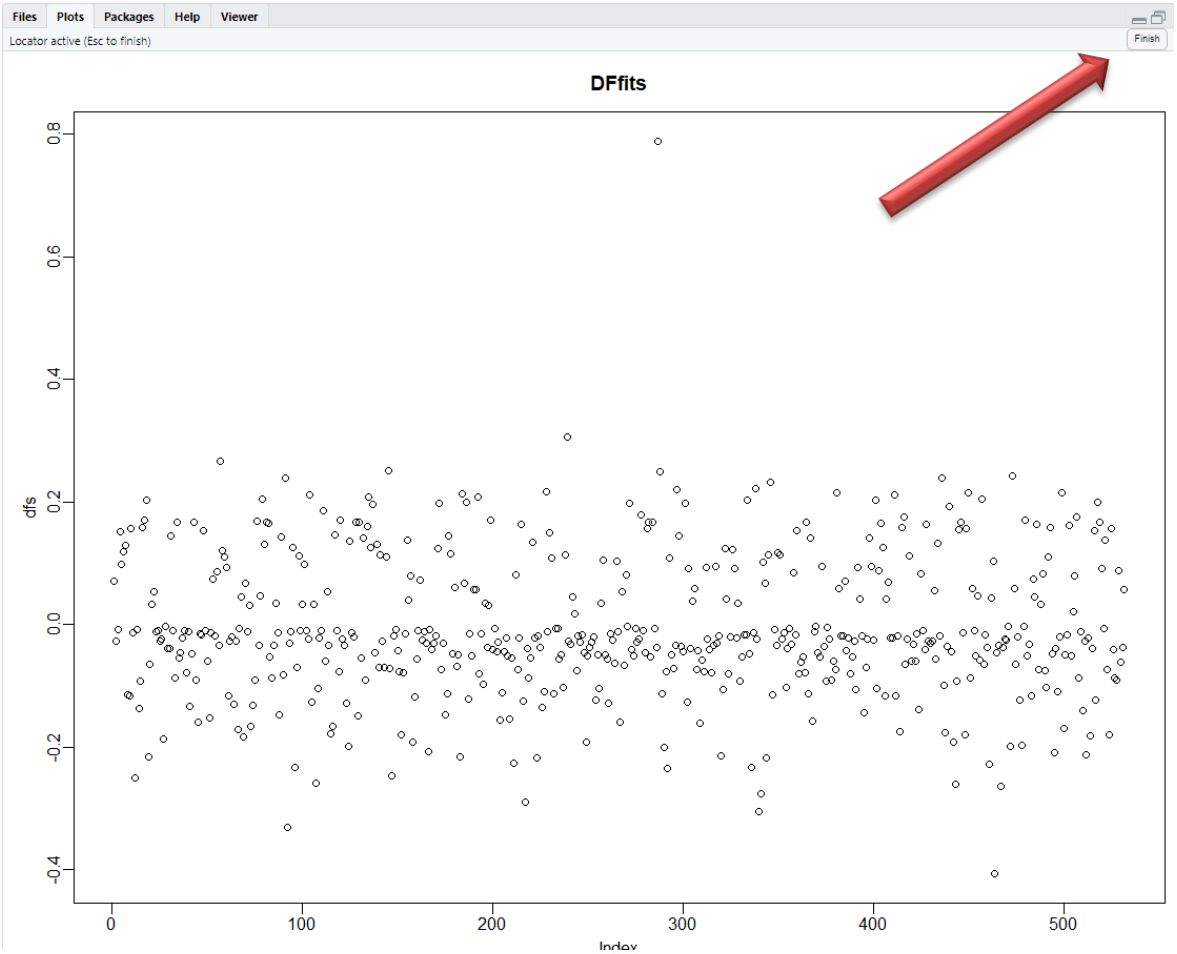


Figure 22: dffits plotted showing locator active with "finish button".

e) Next, we run the `influenceinIndexPlot` seen in the frogs example. This again shows observation 287 as being a point of concern. (Note that dffits and Cook's distance are very similar measures).

f) We can look at the default set of diagnostic plots, but for logistic regression they are difficult to interpret. They do again highlight observation 96 as an outlier and observation 287 as having high leverage. Notice however that 287 does not actually cross the threshold for the Cook's distance in the Residual vs leverage plot. So although it does have high Cook's distance it is not necessary a problem.

7.3.1 Multicollinearity

g) We can check that there are no issues of multicollinearity by using the vif() function. This gives no high variance inflation factors for the model so we can continue. If we did some we could use the same diagnostics as for linear models.

8 Interpretation and Model Performance

We seem to have a reasonable model even though some observations have been flagged in various plots such as 96 and 287. But these did not seem to alter the parameter estimates (which we can tell without having to remove them and refit from the dffbetas). We will therefore proceed to interpret this model and then think about how well it performs.

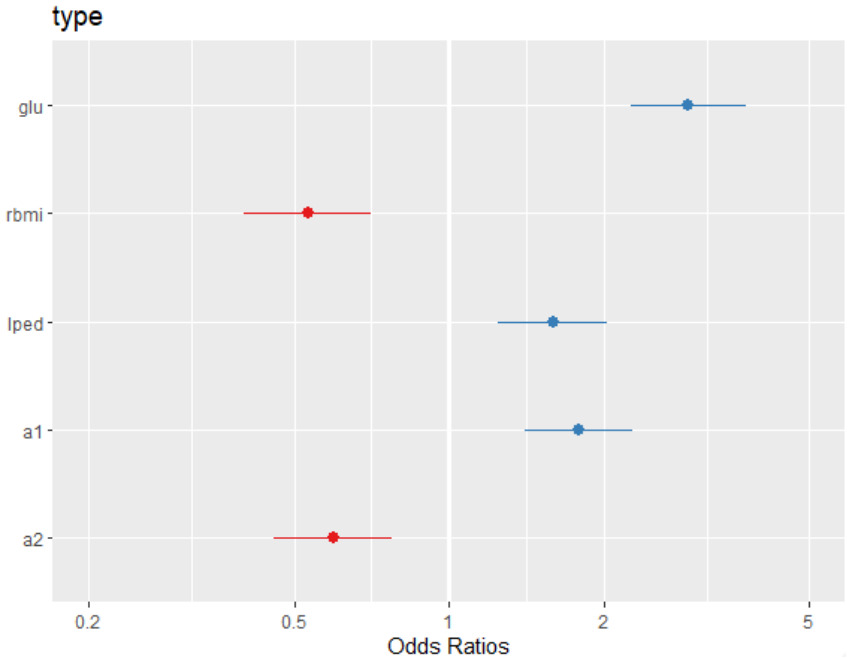
8.1 Understanding the Model

23) Run the first line of the code in BLOCK *XVII*, this first produces the effect plot (J. Fox, et al. 2020) which shows the dose response. We can see that unlike the frogs example the dose response does not quite level off for either ***bmi*** or ***ped***, but there are some observations still on the rug plot on the bottom that are reasonably high. Similarly, for ***age***, the relationship matches the quadratic fitted, but there are plenty of observations beyond the maximum point, suggesting this relationship may not in fact be just due to one value of ***age***.

24) The usual way to interpret the parameters of a logistic regression is via odds ratios. As with the parameters of a linear model the meaning of parameters is dependent on the scale of the variables. Hence it can be useful to think about the standardised coefficients. One way to do this is to make use of the `plot_model()` function from the `sjPlot` package (Lüdecke, et al. 2021). We wish to use the standardised form but unfortunately this function does not work with the transformations we have used. So, we need to create some new variables that contain these transformed variables and then refit the model. It will be the same model but the variables are not transformed within the model statement itself but prior to fitting the model. Run the rest of the code in BLOCK *XVII* which will produce this plot. In addition, it will produce a table of the odds ratios together with their confidence intervals.

			type		
Predictors	*Odds Ratios*	*std. Beta*	*CI*	*standardized CI*	*p*
(Intercept)	0.19	0.33	0.03 – 1.06	0.25 – 0.42	0.059
glu	1.04	2.91	1.03 – 1.04	2.27 – 3.81	**<0.001**
rbmi	0.00	0.53	0.00 – 0.00	0.40 – 0.70	**<0.001**
lped	2.06	1.59	1.42 – 3.04	1.25 – 2.04	**<0.001**
a1	629708.65	1.79	2620.18 – 163724694.15	1.41 – 2.27	**<0.001**
a2	0.00	0.59	0.00 – 0.00	0.45 – 0.76	**<0.001**
Observations	532				
R^2 Tjur	0.389				

Figure 23: Table of coefficients and confidence intervals for both the raw and standardised odds ratios.



From this plot we can see that an increase in glucose concentration increases the probability of having diabetes, whilst increasing the reciprocal of the body mass index (hence decreasing the body mass index) reduces the chance of having diabetes. In particular, a 1 standard error increase in glucose will mean the odds of having diabetes to not having diabetes will increase by a factor of 2.91 on average, keeping all other variables fixed.

Figure 24: Plot of Standardised odds ratios and 95% confidence intervals

The confidence intervals give us a range of values for these odds ratios.

We can read this off the table that appears in the viewer (Figure 23 above), which also shows the raw values, but the graphical display is perhaps an easier way to visualise this for a non-specialist.

The odds ratios are found by exponentiating the original coefficients, to list these in this format we add the option that the transformation should be “NULL”.

	type				
Predictors	*Log-Odds*	*std. Beta*	*CI*	*standardized CI*	*p*
(Intercept)	-1.66	-1.11	-3.41 – 0.06	-1.38 – -0.86	0.059
glu	0.03	1.07	0.03 – 0.04	0.82 – 1.34	**<0.001**
rbmi	-93.65	-0.63	-136.89 – -52.75	-0.92 – -0.36	**<0.001**
lped	0.73	0.46	0.35 – 1.11	0.23 – 0.71	**<0.001**
a1	13.35	0.58	7.87 – 18.91	0.34 – 0.82	**<0.001**
a2	-11.96	-0.52	-18.31 – -6.20	-0.79 – -0.27	**<0.001**
Observations	532				
R^2 Tjur	0.389				

Figure 25: Table of raw coefficients and confidence intervals for both the raw and standardised estimates.

8.2 Model Performance

8.2.1 Classification

25) Run the code in the first two lines of BLOCK *XVIII*, this produces similar diagnostics to those you have seen in the frogs example. The first produces box-plots of the predicted probabilities against the ***type*** - it can be seen that the model does give a higher predicted probability for those with diabetes than those without. However, both whiskers extend the full range so there are plenty of false positives and false negatives.

8.2.2 Confusion Matrix and associated statistics

The next part of the code creates the confusion matrix, but it includes an option to also display the margin totals.

a) Run this next part of code. Also calculate the accuracy. You should find that it is 0.797.

b) We did not have a separate training and test set (we merged these at the start, partly because with this size of data set we did not really have enough observations to warrant splitting them.) The next code therefore uses cross-validation to re-estimate the accuracy. To do so the algorithm has to be able to find the data in the R model object which it can't because we `attach`ed the data rather than specified it when we fitted the model. So we quickly refit the models but specify the data set. For comparison we try the earlier models. Running this code you should find you get similar results for the cross-validation estimate. The selected model also seems to give the highest accuracy.

c) As explained in the last lecture on the frogs example, accuracy has its limitations. Consequently, we also calculate the false positive and false negative rates. We can see that the false negative rate is rather high. This means that 37.9% of individuals with diabetes are incorrectly predicted to not have diabetes.

8.2.3 ROC curves

8.2.3.1 Basic ROC

d) Next we produce a plot of the ROC curve in the same way as in the frogs example. This is a plot of the false negative rate against the false positive rate. These rates are changed if we change the cut-off from 0.5. To produce the confusion matrix we did so by running this code:

```
ypred <- predicted > 0.5
addmargins(table(type, ypred))
```

If we change the cut-off in this code to other values such as 0.1, 0.015, etc. we will generate a different confusion matrix each time and hence a different accuracy, true negative rate, true positive rate, false positive rate and false negative rate etc.. The functions `predicted` and `performance` creates all of the possible values of these rates which then allows us to plot them.

The first plot produced is similar to that seen in the frogs example.

8.2.3.2 Traditional ROC

It is more usual with ROC curves to plot the true positive rate (also known as the sensitivity) against the true negative rate on a reverse axis (also know as the specificity) or the false positive rate on the usual 0-1 axis (all these latter three things are just the same!). Unfortunately, different disciplines use different terms. (I have worked with those studying health and they use sensitivity and specificity, whilst colleagues in text classification have used precision and recall) Have a look at the Wikipedia article on the Receiver Operating Characteristic curve (Wikipedia contributors 2021) for the whole list of definitions of statistics calculated from the confusion matrix. (Perhaps that is why it is called the confusion matrix!).

e) The next part of the code produces this traditional ROC curve. Notice that it appears in the top upper segment. Both forms are valid. For the first one, better models have a ROC that is as low as possible to the left corner of the plot, whilst the traditional one is as high as possible to the left.

8.2.3.3 Traditional ROC plus confidence intervals and AUC

One way of comparing models is to calculate the area under the ROC curve. There are even different versions of this (of course there are!). One is just the AUC (which stand for "Area Under the Curve) but some versions take this as the area to the diagonal line so is just 0.5 lower in value. To calculate this, we need a package that produces a more sophisticated version of these. We installed the required package at the start: `pROC` (Robin, et al. 2021) and the final selection of code uses this to reproduce the traditional ROC.

f) Run this last section of code in BLOCK XVIII that reproduces the traditional ROC and then plots it. The next lines calculate the AUC and the confidence interval for the AUC. Next a bootstrap is used to add error bars to the ROC and print the AUC onto the ROC chart itself.

g) Next, to illustrate how we may produce some of this if we did have two data sets we fit the model with the training data (which for some reason is smaller than the test set!!?) and then plot the ROC and the AUC. We then plot the ROC for the test data but using the model fitted with the training data and finally we plot both on one graph. Notice that both curves match. If the test set gave a substantially lower ROC that would be a potential cause for concern. It would suggest the model did not perform as well on unseen data.

h) Finally, to compare some of our models we produce these ROC charts for three of our earlier models. We find that there is very little to choose between them. Taking transformations does increase the AUC slightly, whilst then reducing the model leaves the AUC with a very similar value.

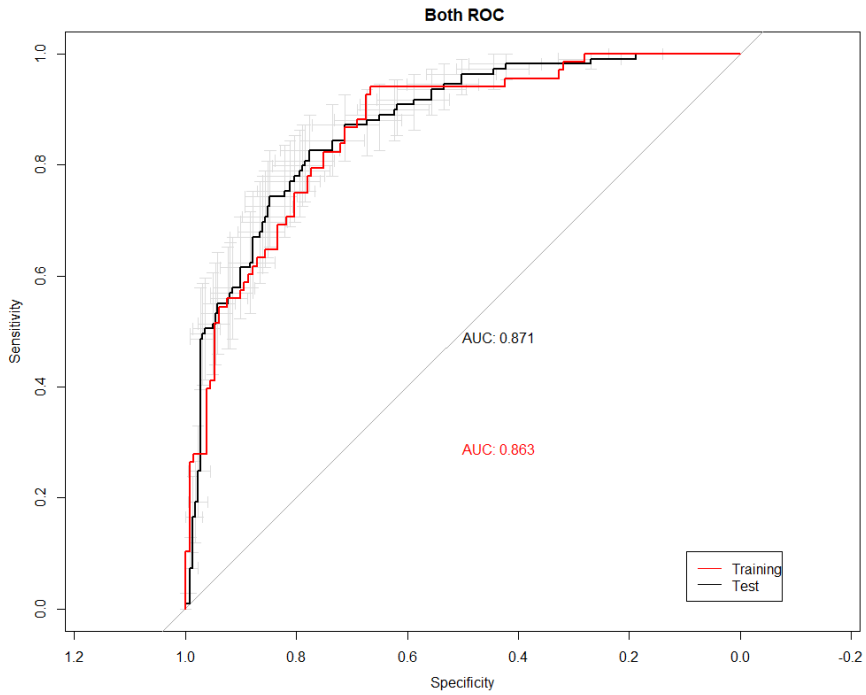


Figure 26: Comparison of ROC for training and test data

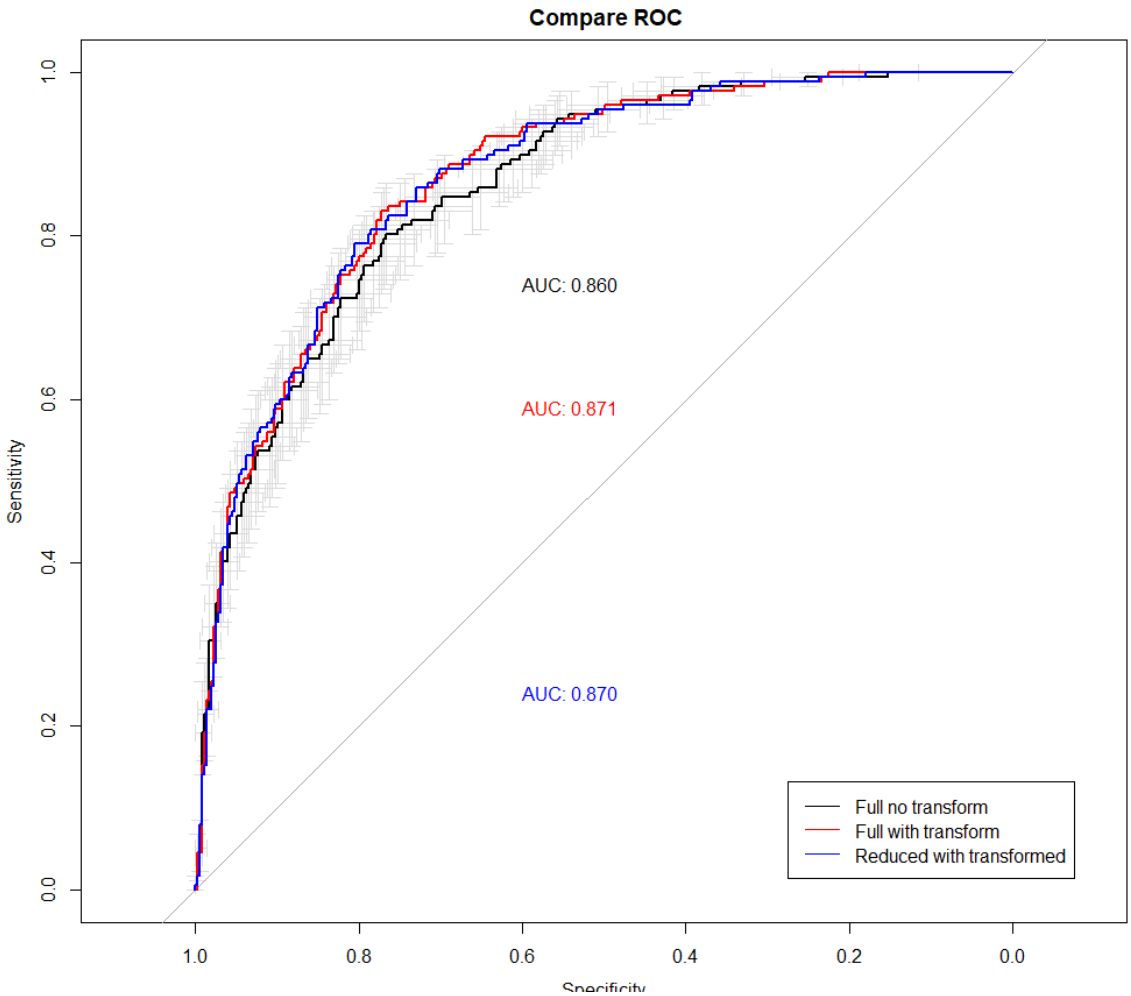


Figure 27: ROC curves comparing three possible models

The ideal AUC is of course 1, but for real data these AUC are not too bad.

9 References

Cannon, Ann, George Cobb, Bradley Hartlaub, Julie Legler, Robin Lock, Thomas Moore, Alan Rossman, and Jeffrey Witmer. 2019. “Package 'Stat2Data'.” 4 January. Accessed November 25, 2019. https://cran.r-project.org/web/packages/Stat2Data/Stat2Data.pdf.

de Andrade Moral, Rafael [aut, cre], John [aut] Hinde, and Clarice [aut] Garcia Borges Demetrio. 2018. *Half-Normal Plots with Simulation Envelopes.* Accessed 3 22, 2021. https://cran.rstudio.com/web/packages/hnp/hnp.pdf.

endmemo.com. 2016. “R cut Function.” Accessed November 2019. http://www.endmemo.com/program/R/cut.php.

Fox, John [aut, cre], Sanford [aut] Weisberg, Brad [aut] Price, Michael [aut], Friendly, Jangman [aut] Hong , Robert [ctb] Andersen, David [ctb] Firth, Steve [ctb] Taylor, and R Core Team [ctb]. 2020. *Package ‘effects’.* 11 8. Accessed 3 23, 2021. https://cran.r-project.org/web/packages/effects/effects.pdf.

Fox, John, and Sanford Weisberg. 2019. *An R Companion to Applied Regression.* Third. Los Angeles: SAGE.

Gelman , Andrew [aut], Yu-Sung [aut, cre] Su , Masanao [ctb] Yajima , Jennifer [ctb] Hill , Maria [ctb] Grazia Pittau , Jouni [ctb] Kerman, Tian [ctb] Zheng , and Vincent [ctb] Dorie. 2018. “Package ‘arm’.” 13 April. Accessed November 25, 2019. https://cran.r-project.org/web/packages/arm/arm.pdf.

Jeppson, Haley [aut, cre], Heike [aut] Hofmann, Di [aut] Cook, and Hadley [ctb] Wickham. 2021. *Mosaic Plots in the 'ggplot2' Framework.* 23 2. Accessed 3 22, 2021. https://github.com/haleyjeppson/ggmosaic.

Lüdecke, Daniel [aut, cre], Alexander [ctb] Bartel , Carsten [ctb] Schwemmer , Chuck [ctb] Powell, Amir [ctb] Djalovski, and Johannes [ctb] Titz . 2021. *Package ‘sjPlot’: Data Visualization for Statistics in Social Science.* 10 1. Accessed 3 23, 2021. https://cran.r-project.org/web/packages/sjPlot/sjPlot.pdf.

Maindonald , John [aut,mtr] H, and John [aut] W. Braun. 2020. *DAAG: Data Analysis and Graphics Data and Functions.* 10 3. Accessed 3 23, 2021. https://CRAN.R-project.org/package=DAAG .

Ripley, Brian, Bill Venables, Douglas M. Bates, Kurt Hornik, Albrecht Gebhardt, and David Firth. 2021. “Package 'MASS'.” 12 2. Accessed 3 17, 2021. https://cran.r-project.org/web/packages/MASS/MASS.pdf.

Robin, Xavier [cre, aut], Natacha [aut] Turck, Alexandre [aut] Hainard, Natalia [aut], Tiberti, Frédérique [aut] Lisacek, Jean-Charles [aut] Sanchez, Markus [aut] Müller, Stefan [ctb] Siegert , and Matthias [ctb] Doering. 2021. *Package ‘pROC’.* 13 1. Accessed 3 23, 2021. https://cran.r-project.org/web/packages/pROC/pROC.pdf.

Schloerke, Barret [aut, cre], Di [aut, ths] Cook, Joseph [aut] Larmarange, Francois [aut] Briatte, Moritz [aut] Marbach, Edwin [aut] Thoen, Amos [aut] Elberg, et al. 2021. *GGally: Extension to 'ggplot2'.* 08 3. Accessed 3 22, 2021. https://CRAN.R-project.org/package=GGally ; https://ggobi.github.io/ggally/.

Sing, Tobias [aut], Oliver [aut] Sander, Niko [aut] Beerenwinkel, Thomas [aut] Lengauer , Thomas [ctb] Unterthiner , and Felix [cre] G. M. Ernst. 2020. *Package 'ROCR'.* 2 5. Accessed 3 23, 2021. https://cran.r-project.org/web/packages/ROCR/ROCR.pdf.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis.* New York: Springer-Verlag. Accessed 3 22, 2021. https://ggplot2.tidyverse.org.

Wickham, Hadley. 2011. "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software* 40 (1): 1-29. Accessed November 25, 2019. https://cran.r-project.org/web/packages/plyr/index.html; http://www.jstatsoft.org/v40/i01/.

Wikipedia contributors. 2021. *Receiver Operating Characteristic.* 19 3. Accessed 3 23, 2021. https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=1012918905.

If you notice any typos please let me know via email: teresa.brunsdon@warwick.ac.uk