

Homework Assignment 7: Diabetes

36-402, Advanced Data Analysis, Spring 2011

SOLUTIONS

1. ANSWER:

```
# Look what variables are in the data:
colnames(pima)
# Consider maximums and minimums to check that the variable values are feasible
summary(pima)
help(pima)
```

From the variable descriptions, it is not perfectly clear what feasible values would be for each variable, but we strongly suspect that body mass index should never be zero (that would imply zero weight), nor should diastolic blood pressure, blood glucose level, or the triceps skin fold thickness. This is supported both by common sense and also by the fact that the distribution for each of these variables appears to have an irregular spike at 0. Insulin also has such a spike. There is a form of diabetes (“type 1”) where insulin levels are very low or even undetectable, but this is actually rare among Native Americans, as opposed to some parts of Europe (Scandinavia, Sardinia)¹, so while these zeroes are, most likely, mistakes or missing values, there is no penalty for not removing them. Leaving them in changes the estimates in later problems, but not hugely.

Let’s remove these observations:

```
good.data = (pima$glucose != 0) & (pima$diastolic != 0) & (pima$triceps != 0) &
  (pima$bmi != 0) & (pima$insulin != 0)
pima.clean = pima[good.data,]
# Make note that we’ve deleted about 50% of the observations!
nrow(pima.clean)/nrow(pima)

# Make a scatterplot (figure 1)
plot(pima.clean, cex = 0.4, pch = 16)
```

See Figure 1 for an overview of the pairwise relationships. The data looks “healthy” now that the zero observations are removed.

¹See, e.g., *The Merck Manual*, 18th edition, ch. 158, online at <http://www.merckmanuals.com/professional/sec12/ch158/ch158b.html>.

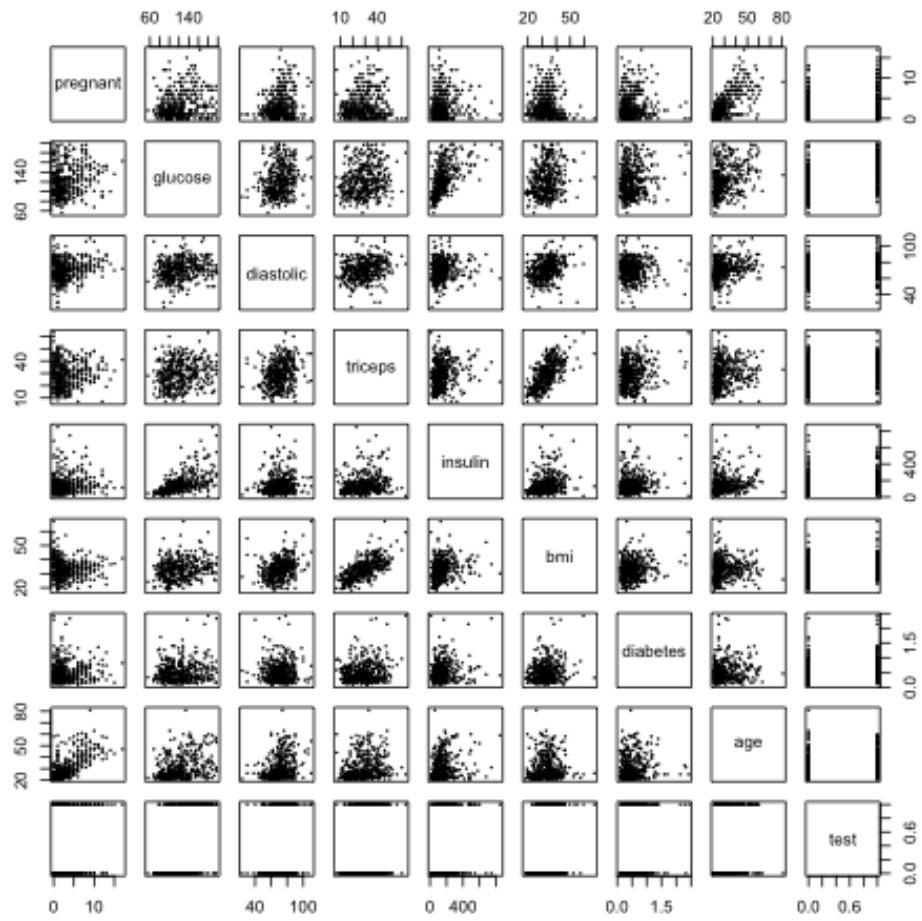


Figure 1:

It's good to keep in mind the overall frequency of testing positive for diabetes, since this gives a lower bound for model performance; the model that always predicts a negative test result will be correct about 67% of the time:

```
mean(pima.clean$test) # So about 0.33 of the test results are 1 (positive, have
# diabetes) and the rest are 0s
```

2. ANSWER:

```
# glm does logistic regression when we choose family = binomial(link = logit)
logit.model = glm(test~., data = pima.clean, family = binomial(link = logit))
summary(logit.model) #names(logit.model)
```

#output includes the coefficients:

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-1.004e+01	1.218e+00	-8.246	< 2e-16	***
pregnant	8.216e-02	5.543e-02	1.482	0.13825	
glucose	3.827e-02	5.768e-03	6.635	3.24e-11	***
diastolic	-1.420e-03	1.183e-02	-0.120	0.90446	
triceps	1.122e-02	1.708e-02	0.657	0.51128	
insulin	-8.253e-04	1.306e-03	-0.632	0.52757	
bmi	7.054e-02	2.734e-02	2.580	0.00989	**
diabetes	1.141e+00	4.274e-01	2.669	0.00760	**
age	3.395e-02	1.838e-02	1.847	0.06474	.

Actually, when using family=binomial, the logistic or logit link function is the default, but it never hurts to be explicit about it.

3. ANSWER:

```
# Create the imaginary test subject:
woman.1 = as.data.frame(matrix(
  c(2, 99, 64, 22, 76, 26, 0.25, 30, NA), nrow=1))
names(woman.1) = names(pima)
# Alternately:
woman.1 = data.frame(pregnant=2,glucose=99,diastolic=64,triceps=22,insulin=76,
  bmi=26,diabetes=0.25,age=30)
# We can leave off "test" because only variables appearing in the right-hand
# side of the model's formula are needed to make a prediction
# Make the prediction. IMPORTANT: predict return the predicted value for the
# transformed response (the log odds); get a confidence interval for that, since
# it's approximately Gaussian, and then transform back to the probability scale
pred.lin = predict(logit.model, newdata = woman.1, se.fit = T)
lower.lin = pred.lin$fit - 1.96*pred.lin$se.fit
```

```

upper.lin = pred.lin$fit + 1.96*pred.lin$se.fit
# 95% C.I. lower bound:
ilogit(lower.lin)
# Equivalently, exp(lower.lin)/(1+exp(lower.lin))
# 95% C.I. upper bound:
ilogit(upper.lin)
# Equivalently, exp(upper.lin)/(1+exp(upper.lin))

```

We obtain a lower bound of about 0.030 and an upper bound of about 0.097. Note that instead of computing the standard error for the prediction explicitly, we are using the *se.fit = T* option in the predict function.

By using `predict()` with the `type="response"` option, one can get a standard error on the untransformed, probability scale; but the sampling distribution here is not as nearly Gaussian as it is on the transformed, logit scale, so using this mechanically to get an approximate confidence interval is not very accurate, unless the standard error is very small and/or the predicted probability is zero close to 0.5. (This was discussed in lecture and in Faraway.)

4. ANSWER: Note that $\text{odds} = \frac{p}{1-p}$. If β_b is the estimated coefficient for `bmi`, the logistic regression equation is

$$\log(\text{odds}) = \beta_b * \text{bmi} + \text{other}$$

where “other” includes the intercept and all the other terms and their coefficients. Then the (multiplicative) change in odds of testing positive for diabetes for someone moving from the third quartile to the first quartile in `bmi` is the ratio

$$\frac{e^{\beta_b * \text{bmi.1st.quantile} + \text{other}}}{e^{\beta_b * \text{bmi.3rd.quantile} + \text{other}}} = e^{\beta_b * (\text{bmi.1st.quantile} - \text{bmi.3rd.quantile})}$$

We code this in R:

```

odds.change = exp(coefficients(logit.model)["bmi"]*
  (quantile(pima.clean$bmi,0.25)-quantile(pima.clean$bmi,0.75)))
> as.numeric(odds.change)
[1] 0.54

```

(We could replace the difference in quantiles with `-IQR(pima.clean$bmi)`, but the `IQR()` function is a little obscure.) Therefore, all else held constant, the model suggests that a woman at the first quartile on `bmi` has about half the odds of testing positive for diabetes compared to being at the third quartile.

But how confident can we be of this? From the table in problem 2, we see the standard error of the `bmi` coefficient β_b is about 0.027. The estimated coefficients in a logistic regression have approximately Gaussian

distributions around their true values, so a 95% confidence interval for β_b is $0.075 \pm 1.96 * 0.027$, or (0.017, 0.12). In R:

```
bmi.se = summary(logit.model)$coefficients["bmi","Std. Error"]
bmi.lower = as.numeric(coefficients(logit.model) ["bmi"] - 1.96*bmi.se)
bmi.upper = as.numeric(coefficients(logit.model) ["bmi"] + 1.96*bmi.se)
```

Plugging the lower and upper bounds into the formula for the change in odds gives a confidence interval for the change in odds:

```
odds.change.upper = exp(bmi.lower*
  (quantile(pima.clean$bmi,0.25)-quantile(pima.clean$bmi,0.75)))
> as.numeric(odds.change.upper)
[1] 0.863
```

```
odds.change.lower = exp(bmi.upper*
  (quantile(pima.clean$bmi,0.25)-quantile(pima.clean$bmi,0.75)))
> as.numeric(odds.change.lower)
[1] 0.340
```

We can be rather confident that moving to the lower `bmi` is associated with decreasing the odds of testing positive for diabetes, by a factor between 0.34 and 0.86.

Alternatively, a somewhat more accurate method of calculating confidence intervals for logistic regression coefficients is available in the package `MASS` (as illustrated by Faraway). Using that here:

```
> library(MASS)
> confint(logit.model,"bmi")
Waiting for profiling to be done...
      2.5 %      97.5 %
0.01766988 0.12534312
```

gives a very similar confidence interval for β_b , and so a very similar confidence interval for the change in odds, namely (0.336, 0.858).

5. ANSWER: We can directly compare the mean diastolic blood pressure for women who test positive for diabetes against those who don't:

```
# Diastolic blood pressure for those testing negative:
> mean(pima.clean$diastolic[pima.clean$test == 0])
[1] 68.96947
# Diastolic blood pressure for those testing positive:
> mean(pima.clean$diastolic[pima.clean$test == 1])
[1] 74.07692
```

It appears that high diastolic blood pressure is associated with an increase in diabetes risk. We can use a t -test to see whether this difference in means is significant:

```
> t.test(x=pima.clean$diastolic[pima.clean$test == 1],
         y=pima.clean$diastolic[pima.clean$test == 0],
         alternative="greater")
```

Welch Two Sample t-test

```
data: pima.clean$diastolic[pima.clean$test == 1] and pima.clean$diastolic[pima.clean$test == 0]
t = 3.761, df = 237.753, p-value = 0.0001066
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 2.865015      Inf
sample estimates:
mean of x mean of y
 74.07692  68.96947
```

The t -statistic tells us that the difference in means is large compared to the fluctuations around the global mean, and the p -value tells us that it is really unlikely that we'd get that big a test statistic just by chance.

However, looking back at the estimated coefficients from the logistic regression shown in problem 2, the coefficient on blood pressure is actually *negative* (-1.42×10^{-3}), though statistically insignificant (p -value of 0.90). Thus while we have just seen that higher blood pressure is significantly associated with diabetes, the logistic regression model says that there is no significant relation between the variables. This appears to be a contradiction.

However, note that diastolic blood pressure is positively correlated with body mass index:

```
> cor(pima.clean$diastolic, pima.clean$bmi)
[1] 0.3044034
```

Also, there is a highly statistically significant and positive relationship between `bmi` and diabetes risk as shown in the coefficients in problem 2. This gives us reason to think that the association between diabetes and diastolic blood pressure is secondary, in some sense, to the correlation between `bmi` and diabetes risk. After taking into account `bmi`, changes in `diastolic` have do not have a significant association with diabetes risk.

6. ANSWER: There are several ways of doing this, and we can't go through all valid solutions. Several of these methods are described in Section 2.9 of Faraway; there is also the generalized-additive-model approach of the lecture notes. All approaches are based on the idea that the model fits the

data well if the difference between the true probability of testing positive for diabetes and the predicted probability is small over the covariate space.

Option 1 (based on Faraway): A simple way to test that the predicted and the truth do not deviate much is as follows. Gather all observations for which the predicted value \hat{p} is in a small interval, say, $(0, 0.5)$. Count how many times the diabetes test is positive in this group. Then use the \hat{p} values to compute the number of positive diabetes test results we would *expect* to see in this group, based on the model. If the difference between observed and expected is large, we have evidence to reject that the model fits well. To be more thorough, we actually want to consider more than just the interval $(0, 0.5)$. Partitioning the full interval $(0, 1)$ into 10 pieces is standard practice, and then we can use the chi-squared statistic to compute a p-value for the goodness of fit.

Option 2 (based on lecture notes): A non-parametric model should match the training data (as measured by log-likelihood or deviance) at least as well as the logistic regression model, because the former is more flexible. When the logistic regression is correct, however, the non-parametric model ends up approximating the parametric model, and the difference should be small and shrinking with sample size. We do not have a closed-form expression for the distribution of the difference in deviances, but we can find it by simulating from the logistic regression model (parametric bootstrapping).

7. ANSWER:

Option 1:

```
# Function to compute a p-value for the goodness-of-fit
# using the Hosmer and Lemeshow goodness of fit test.
# This code is taken from http://sas-and-r.blogspot.com/
hosmerlem = function(y, yhat, g=10) {
  # divide yhat into 10 equal bins after ordering:
  cutyhat = cut(yhat,breaks = quantile(yhat, probs=seq(0,
    1, 1/g)), include.lowest=TRUE)
  # tabulate the association between y and the 10 yhat bins:
  obs = xtabs(cbind(1 - y, y) ~ cutyhat)
  # tabulate the expected association based on the predicted values:
  expect = xtabs(cbind(1 - yhat, yhat) ~ cutyhat)
  # compute a statistic that should be distributed Chi-squared under the null:
  chisq = sum((obs - expect)^2/expect)
  # The p-value:
  P = 1 - pchisq(chisq, g - 2)
  return(list(chisq=chisq,p.value=P))
}
```

Use the function to compute a p-value for goodness of fit:

```
> hosmerlem(y=pima.clean$test, yhat=logit.model$fitted)
$p.value
[1] 0.995313
```

The p-value is quite large, so this test does not provide evidence against the hypothesis that the model fits the data well.

Option 2 follows the notes: fit a GAM, compare the deviances, then simulate from the logistic regression and compare deviances on the simulations. For conformity with the notes, this uses the `gam()` function from the package `gam`, but using the one from `mgcv`, as on the midterm, wouldn't change things much.

```
pima.gam <- gam(test ~ lo(pregnant) + lo(glucose) + lo(diastolic) + lo(triceps)
               +lo(insulin) +lo(bmi) + lo(diabetes) + lo(age), data=pima.clean,
               family=binomial)
```

(See Figure 2 for the plots of the partial response functions.) Common mistakes here were to forget to enclose the terms to be smoothed in `lo()` or `s()`, and forgetting to specify `family=binomial`.

Now compare the deviances:

```
> (diff.deviance.obs <- logit.model$deviance - pima.gam$deviance)
[1] 36.72602
```

As expected, the GAM does better at fitting the data than the logistic model. Is this within the range we'd expect if the logistic model were true?

```
sim.from.logistic <- function(model=logit.model) {
  # Presume that the fitted values of the model are probabilities
  # This is true of logistic regression
  p.hat <- fitted(model)
  n <- length(p.hat)
  # Generate binary data with those probabilities
  y.new <- rbinom(n,size=1,prob=p.hat)
  return(y.new)
}

sim.diff.deviance <- function(model=logit.model,x=pima.clean[,-9]) {
  # simulate from the logistic model
  y.new <- sim.from.logistic(model)
  GLM.new <- glm(y.new ~ x[,1] + x[,2] + x[,3] + x[,4] + x[,5] + x[,6]
                + x[,7] + x[,8], family=binomial)
  GAM.new <- gam(y.new ~ lo(x[,1]) + lo(x[,2]) + lo(x[,3]) + lo(x[,4]) +
                 lo(x[,5]) + lo(x[,6]) + lo(x[,7]) + lo(x[,8]), family=binomial)
```

```

diff.dev <- GLM.new$deviance - GAM.new$deviance
return(diff.dev)
}

```

Notice that there are *eight* predictor variables in this model, not two as in the example in the notes, and since all of them were used on the real data, all of them must be used on the simulations as well. Clearly writing out formulas like this is a bit awkward, especially since we already wrote out the formulas for fitting the original models. R *has* functions for manipulating and generating formulas, but that gets more deeply into pure R programming than is really appropriate here.

As a quick check on the simulator, it should give about the same proportion of positive results as the real data:

```

> mean(sim.from.logistic())
[1] 0.3545918

```

Running `sim.diff.deviance` once show it takes about a third of a second (on my computer), so I take a few minutes to do 1000 simulations:

```

null.dist.diff.deviance <- replicate(1000,sim.diff.deviance())

```

The p -value is then approximately

```

> mean(null.dist.diff.deviance > diff.deviance.obs)
[1] 0.185

```

so there's quite a substantial chance (about 18%) of getting this big a difference when the logistic regression is right; we don't have much cause to reject the logistic regression.

To be *really* rigorous about this (more than the problem requires), we should however check that the test had reasonable power to detect departures from the logistic regression. We get a sense of this by simulating from the GAM:

```

> (critical.value = quantile(null.dist.diff.deviance,0.95))
42.28163
> alt.dist <- replicate(1000,sim.diff.deviance(pima.gam))
> mean(alt.dist> critical.value )
[1] 0.824
> mean(alt.dist <= diff.deviance.obs)
[1] 0.068

```

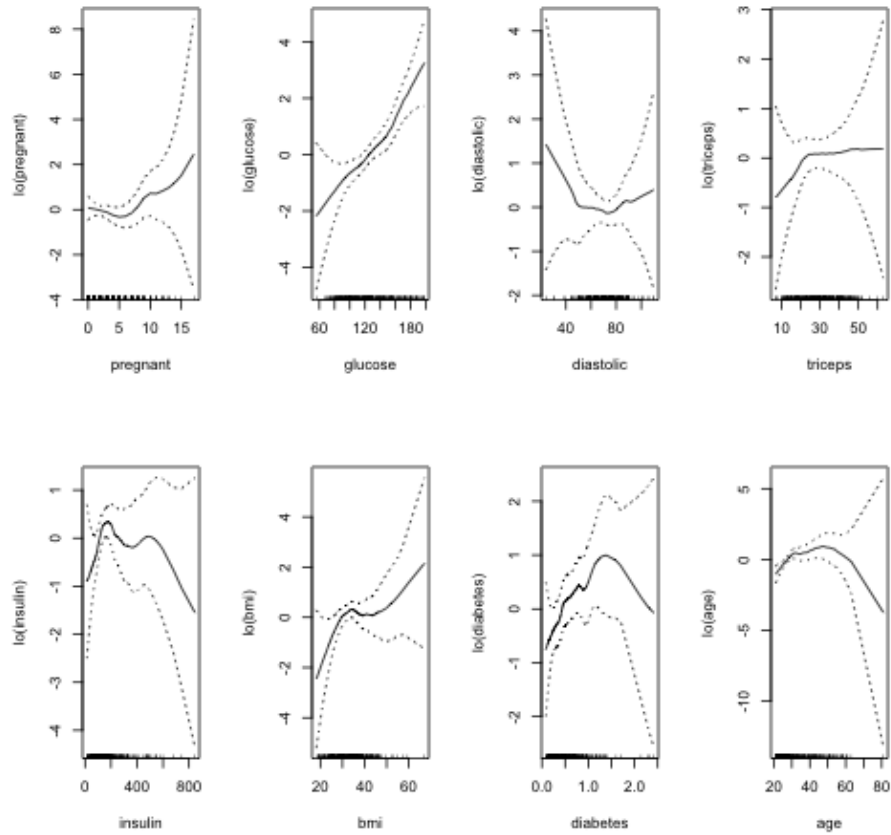
If we reject the logistic model when the p -value is below 0.05, we can work out what the critical value for the difference in deviances is from

the sampling distribution under the null. If the GAM was correct, we'd have a quite large probability (about 82%) of being over that threshold, and so of noticing that the logistic regression was wrong. Similarly, the probability of getting such a *small* difference in deviances is only about 7%, so we have pretty reasonable (though not conclusive!) evidence against the GAM. (Figure 4.) Of course there could be other nonlinear models which are closer to the logistic regression, and which we would not have good evidence to reject, but at least large nonlinearities, like in Figure 2, can be ruled out with some confidence.

8. ANSWER: See problem 7 for the code fitting the GAM.

```
plot(fitted(pima.gam), fitted(logit.model),
     xlab="Nonparametric fitted probabilities",
     ylab="Logistic fitted probabilities", cex = 0.4, pch = 16)
abline(0,1,col="grey")
```

The two models seem to agree tolerably about who has very low probability of becoming diabetic (though the logistic regression gives them somewhat higher probabilities), but the disagreements become more common and bigger at higher predicted probabilities.

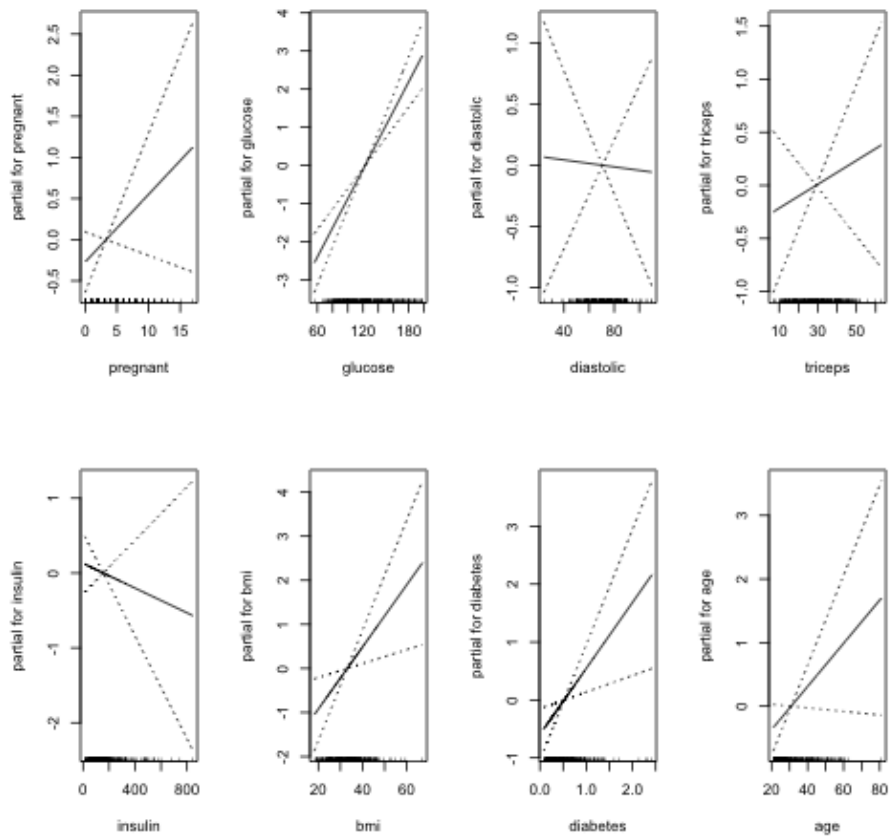


```

par(mfrow=c(2,4))
plot(pima.gam,se=TRUE)
par(mfrow=c(1,1))

```

Figure 2: Partial response functions for the generalized additive model fit in Problem 7; dotted lines show ± 2 standard errors. Note that the vertical axes have different scales across the plots.

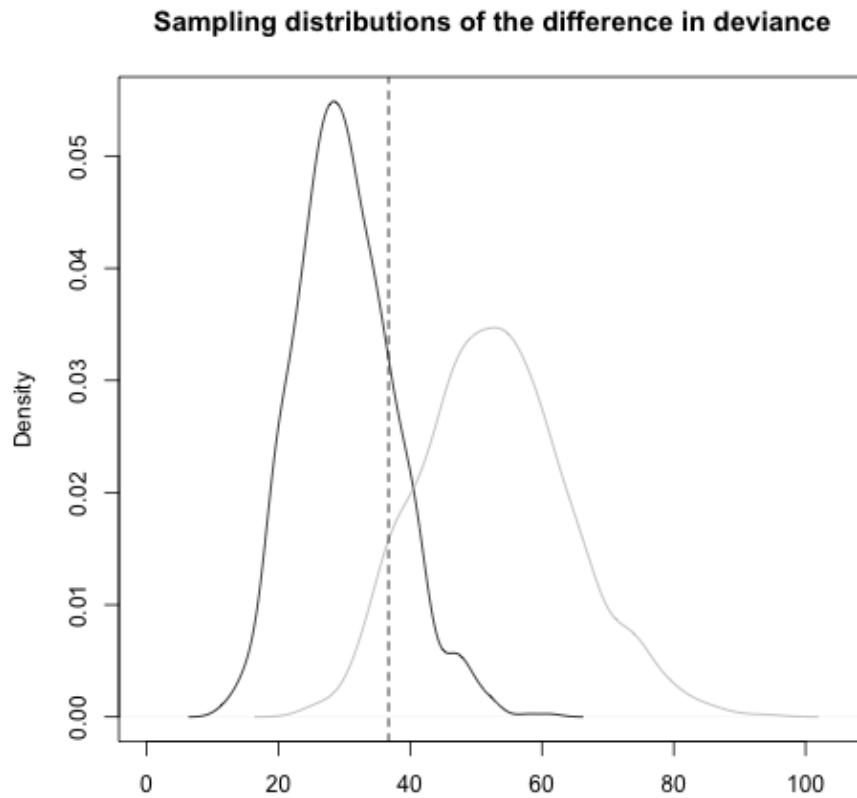


```

par(mfrow=c(2,4))
plot.gam(logit.model,se=TRUE)
par(mfrow=c(1,1))

```

Figure 3: Similar plot for the logistic regression model.



```
plot(density(null.dist.diff.deviance),xlim=c(0,105),
     main="Sampling distributions of the difference in deviance",xlab="")
lines(density(alt.dist),col="grey")
abline(v=diff.deviance.obs,lty=2)
```

Figure 4: Sampling distributions for the difference-in-deviance test statistic when simulating from the logistic regression (black curve) or the GAM (grey curve); dashed vertical line is at the value observed on the read data.

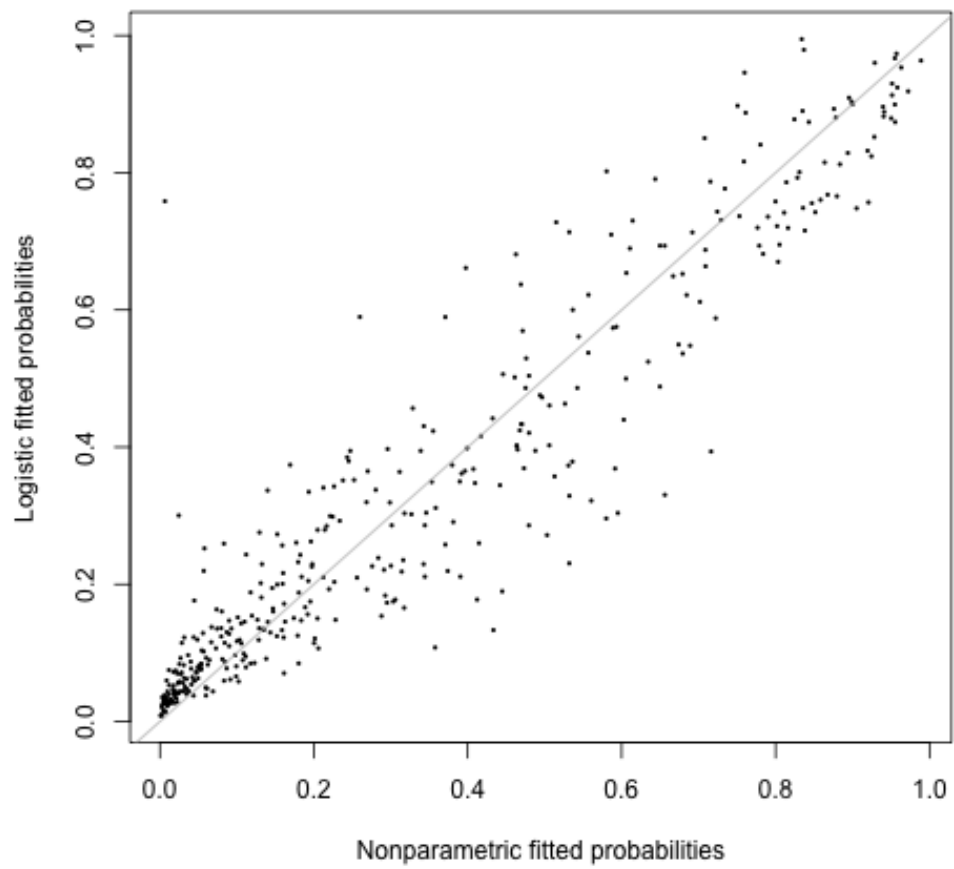


Figure 5: