STAT 526 Assignment 4

Bowen Zheng

2026-02-24

Problem 0

Bowen Zheng

Problem 1

> The data set esoph contains a case-control study of esophageal cancer. The three ordered factors are age (6 levels), tobacco consumption (4 levels), and alcohol consumption (4 levels).

a.

> Fit a binomial GLM that includes all two-factor interactions and use AIC to select the best model. State this model.

```
data("esoph")

full_model <- glm(cbind(ncases, ncontrols) ~ (agegp + alcgp + tobgp)^2, family = binomial, data = esoph
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
best_model <- step(full_model, direction = "both")
```

```
## Start:  AIC=247.88
## cbind(ncases, ncontrols) ~ (agegp + alcgp + tobgp)^2
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##               Df Deviance    AIC
## - alcgp:tobgp  9   37.535 236.59
## - agegp:tobgp 15   50.309 237.36
## - agegp:alcgp 15   56.807 243.86
## <none>             30.824 247.88
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=236.59
## cbind(ncases, ncontrols) ~ agegp + alcgp + tobgp + agegp:alcgp +
##     agegp:tobgp
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##               Df Deviance    AIC
## - agegp:tobgp 15   56.256 225.31
## - agegp:alcgp 15   62.776 231.83
## <none>             37.535 236.59
## + alcgp:tobgp  9   30.824 247.88
##
## Step:  AIC=225.31
## cbind(ncases, ncontrols) ~ agegp + alcgp + tobgp + agegp:alcgp
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##               Df Deviance    AIC
## - agegp:alcgp 15   82.337 221.39
## <none>             56.256 225.31
## + agegp:tobgp 15   37.535 236.59
## + alcgp:tobgp  9   50.309 237.36
## - tobgp        3   80.300 243.35
##
## Step:  AIC=221.39
## cbind(ncases, ncontrols) ~ agegp + alcgp + tobgp
##
##               Df Deviance    AIC
## <none>             82.337 221.39
## + agegp:alcgp 15   56.256 225.31
## + agegp:tobgp 15   62.776 231.83
## + alcgp:tobgp  9   76.886 233.94
## - tobgp        3  105.881 238.94
## - agegp        5  208.825 337.88
## - alcgp        3  210.270 343.32
```

```r
summary(best_model)
```

```
##
## Call:
## glm(formula = cbind(ncases, ncontrols) ~ agegp + alcgp + tobgp,
##     family = binomial, data = esoph)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.19039    0.20737  -5.740 9.44e-09 ***
## agegp.L      3.99663    0.69389   5.760 8.42e-09 ***
## agegp.Q     -1.65741    0.62115  -2.668  0.00762 **
## agegp.C      0.11094    0.46815   0.237  0.81267
## agegp^4      0.07892    0.32463   0.243  0.80792
## agegp^5     -0.26219    0.21337  -1.229  0.21915    
## alcgp.L      2.53899    0.26385   9.623  < 2e-16 ***
## alcgp.Q      0.09376    0.22419   0.418  0.67578    
## alcgp.C      0.43930    0.18347   2.394  0.01665 *  
## tobgp.L      1.11749    0.24014   4.653 3.26e-06 ***
## tobgp.Q      0.34516    0.22414   1.540  0.12358    
## tobgp.C      0.31692    0.21091   1.503  0.13294    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 367.953  on 87  degrees of freedom
## Residual deviance:  82.337  on 76  degrees of freedom
## AIC: 221.39
## 
## Number of Fisher Scoring iterations: 6
```

We fit the model and find the best using AIC in both directions.

Our final model is

$$\text{logit}(p) = \beta_0 + \beta_1(\text{agegp}) + \beta_2(\text{alcgp}) + \beta_3(\text{tobgp})$$

It does not include interactions terms. However, there may be good reasoning to include a quadratic term for agegp.

b.

> Convert your factors to numerical representation using the unclass function. Then construct a simplified Binomial regression model using polynomials of these representations. [HINT: The the factors in the dataset are ordinal, thus R creates orthogonal polynomial rather than binary dummy variables. You can use the significant test results in part (a) to determine up to which polynomial degree you should use for the numerical regression.]

```r
# Create the new predictors
esoph$age <- unclass(esoph$agegp) - 3.5
esoph$alc <- unclass(esoph$alcgp) - 2.5
esoph$tob <- unclass(esoph$tobgp) - 2.5

# Our simplified model from our previous part a results
simplified_model <- glm(cbind(ncases, ncontrols) ~ age + I(age^2) + alc + tob, family = binomial, data =

summary(simplified_model)
```

```
## 
## Call:
## glm(formula = cbind(ncases, ncontrols) ~ age + I(age^2) + alc + 
##     tob, family = binomial, data = esoph)
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.43917    0.13279  -3.307 0.000942 ***
## age          0.86659    0.10244   8.459  < 2e-16 ***
## I(age^2)    -0.23417    0.06402  -3.658 0.000255 ***
## alc          1.06511    0.10458  10.185  < 2e-16 ***
## tob          0.43951    0.09559   4.598 4.27e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 367.953  on 87  degrees of freedom
## Residual deviance:  93.172  on 83  degrees of freedom
## AIC: 218.23
##
## Number of Fisher Scoring iterations: 5
```

c.

> Commment on the fit of this model. Include plots to support your summary

We can first check residual deviance and estimate our dispersion

```
pchisq(93.172, 83, lower=FALSE)
```

```
## [1] 0.2087865
```

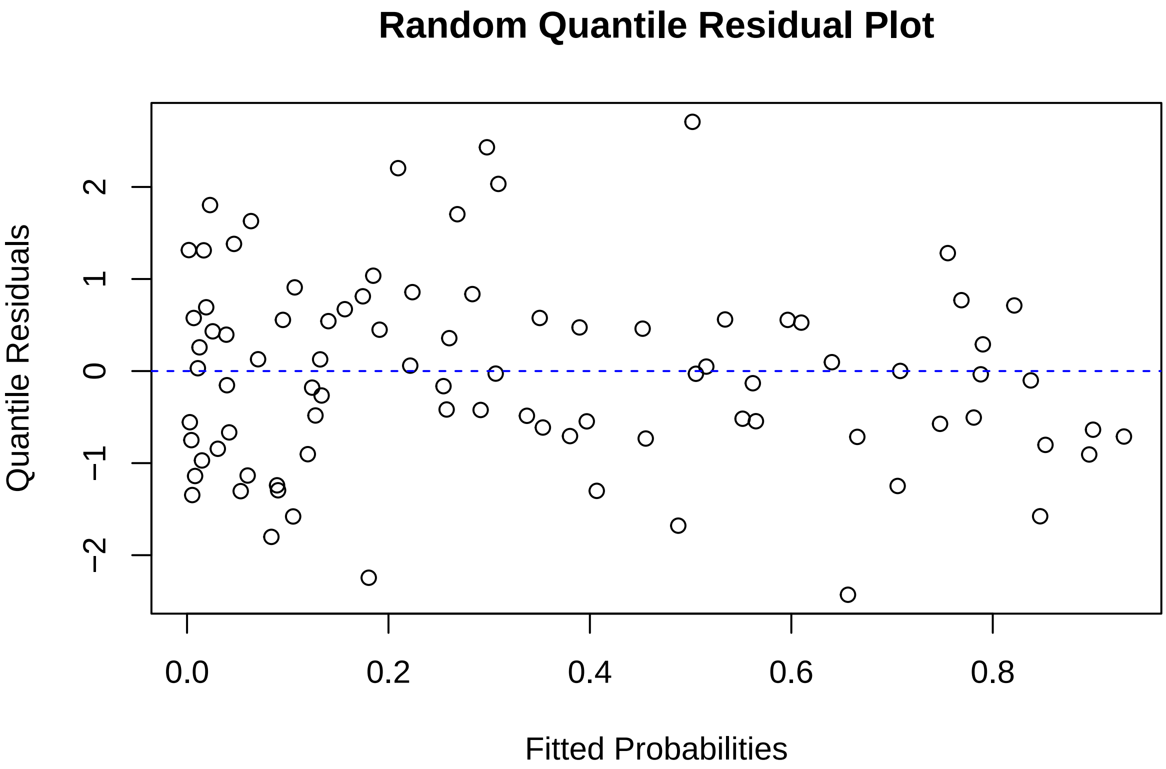
```
summary(simplified_model)$deviance / summary(simplified_model)$df.residual
```

```
## [1] 1.122548
```

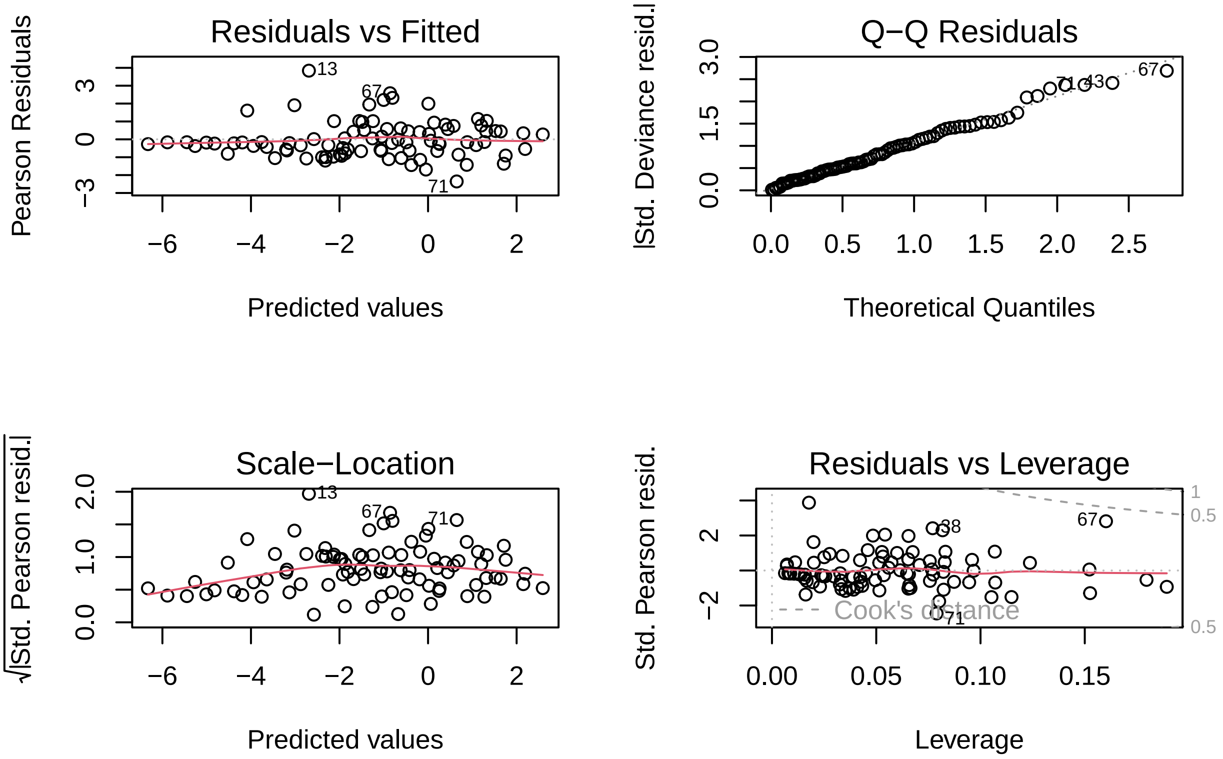
The test concludes that we fail to reject the null, indicating that the model fits the data well. Furthermore, our overdispersion parameter estimate is pretty close to 1, which is what we want.

We also check the diagnostic plots

```
library(statmod)
q_resids <- qres.binom(simplified_model)
plot(fitted(simplified_model), q_resids, xlab = "Fitted Probabilities", ylab = "Quantile Residuals", ma

abline(h = 0, lty = 2, col = "blue")
```



```
par(mfrow = c(2, 2))
plot(simplified_model)
```



The random quantile residual plot seems to have no pattern and could be generally random normal around 0.

Residual plot does not show too many strong outliers, with not very large spread and pretty random. The QQ plot follows the diagonal line. The scale-location is pretty random and doesn't seem to have extremes, especially as the predicted values get larger. The residuals vs leverage shows no high leverage observations.

The overall fit seems good. No reason to doubt it.

d.

> Construct a 95% confidence interval for the effect of moving one category higher in tobacco consumption.

```r
tob_coef <- summary(simplified_model)$coefficients["tob", "Estimate"]
tob_se <- summary(simplified_model)$coefficients["tob", "Std. Error"]
lower_log <- tob_coef - 1.96 * tob_se
upper_log <- tob_coef + 1.96 * tob_se

# We take exp to get to odds ratio
exp(tob_coef)
```

```
## [1] 1.551943
```

```r
OR_lower <- exp(lower_log)
OR_upper <- exp(upper_log)
cat(OR_lower, OR_upper)
```

```
## 1.286794 1.871727
```

```r
# Or use built in R function that I found out only later...
exp(confint(simplified_model, "tob"))
```

```
## Waiting for profiling to be done...
```

```
##    2.5 %   97.5 %
## 1.287394 1.873749
```

We expect the coefficient to be approximately normal and we use its estimate and standard deviation for the CI.

We find CI (1.286794, 1.871727) which does not include 1.

e.

> Construct a 95% confidence interval for the effect of moving from the 45-54 age group to the 55-64 age group.[HINT: You may want to use the vcov(model) function to obtain the covariance matrix of $\hat{\beta}$]

Ok, first we have to figure out which levels we are looking at because we centered it all. We did -3.5 so group 45-54 is factor level 3 and group 55-64 is level 4. They correlate to -0.5 and 0.5 respectively.

We used both linear and quadratic factors so we first have to figure out:

$$\begin{aligned}
Diff &= [\beta_{age}(L4) + \beta_{age^2}(L4^2)] - [\beta_{age}(L3) + \beta_{age^2}(L3^2)] \\
&= \beta_{age}(0.5 - (-0.5)) + \beta_{age^2}(0.5^2 - (-0.5)^2) \\
&= \beta_{age}(1) + \beta_{age^2}(0) \\
&= \beta_{age}
\end{aligned}$$

Ok, we just need a CI for just one times coefficient of age. We do this using the hint vcov

```r
beta_hat <- coef(simplified_model)
V <- vcov(simplified_model)

# Our contrast vector 'L' for the change from age group 3 to 4:
# Change in age: (0.5) - (-0.5) = 1
# Change in age^2: (0.25) - (0.25) = 0
# The vector matches the order of coefficients in our model: (Intercept, age, I(age^2), alc, tob)
L <- c(0, 1, 0, 0, 0)

# Estimate of the effect (Log-Odds)
est_effect <- t(L) %*% beta_hat

# Standard Error of the effect
# SE = sqrt( L' * V * L )
se_effect <- sqrt(t(L) %*% V %*% L)

lower_log <- est_effect - 1.96 * se_effect
upper_log <- est_effect + 1.96 * se_effect
exp(c(lower_log, upper_log))
```

```
## [1] 1.946060 2.907738
```

```r
# Or use R function to find it easily
exp(confint(simplified_model, "age"))
```

```
## Waiting for profiling to be done...
```

```
##    2.5 %   97.5 %
## 1.965098 2.940641
```

We didn't really have to do all this because in the end we just wanted the 2nd SE so again we can simply use the built in R functions.Our final CI is (1.946060, 2.907738)

Problem 2: Faraway Chapter 3 Exercise #4

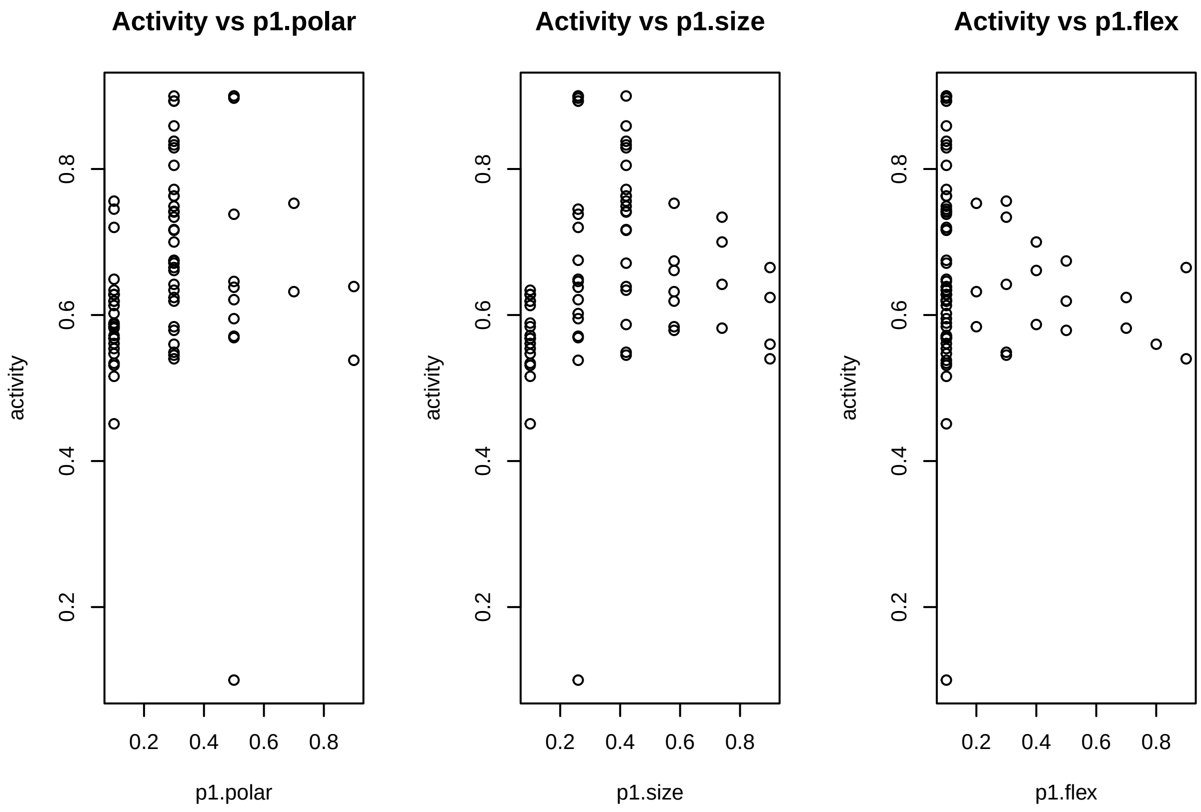
This problem concerns the modeling of the quantitative structure-activity relationships (QSAR) of the inhibition of dihydrofolate reductase (DHFR) by pyrimidines. We want to relate the physicochemical and/or structural properties as exhibited by the 26 predictors in pyrimidines with an activity level. We have structural information on 74 2,4-diamino- 5-(substituted benzyl) pyrimidines used as inhibitors of DHFR in E. coli. All the variables lie in [0,1].

```r
library(faraway)
data(pyrimidines)
```

a.

Plot the activity (response) against the first three predictors. Are any outliers in the response apparent? Remove any such cases.

```r
# Plotting activity against the first three predictors
par(mfrow=c(1,3))
plot(activity ~ p1.polar, data=pyrimidines, main="Activity vs p1.polar")
plot(activity ~ p1.size, data=pyrimidines, main="Activity vs p1.size")
plot(activity ~ p1.flex, data=pyrimidines, main="Activity vs p1.flex")
```



There is one obvious outlier that has values less than 0.2 in activity.

```r
outlier_index <- which(pyrimidines$activity < 0.2)
print(outlier_index)
```

```
## [1] 10
```

```r
pyrimidines_clean <- pyrimidines[-outlier_index, ]
```

Remove the 10th observation

b.

> Fit a Gaussian linear model for the response with all 26 predictors. How well does this model fit the data in terms of R^2? Plot the residuals against the fitted values. Is there any evidence of a violation of the standard assumptions?

```r
linear_model <- lm(activity ~ ., data = pyrimidines_clean)
summary(linear_model)
```

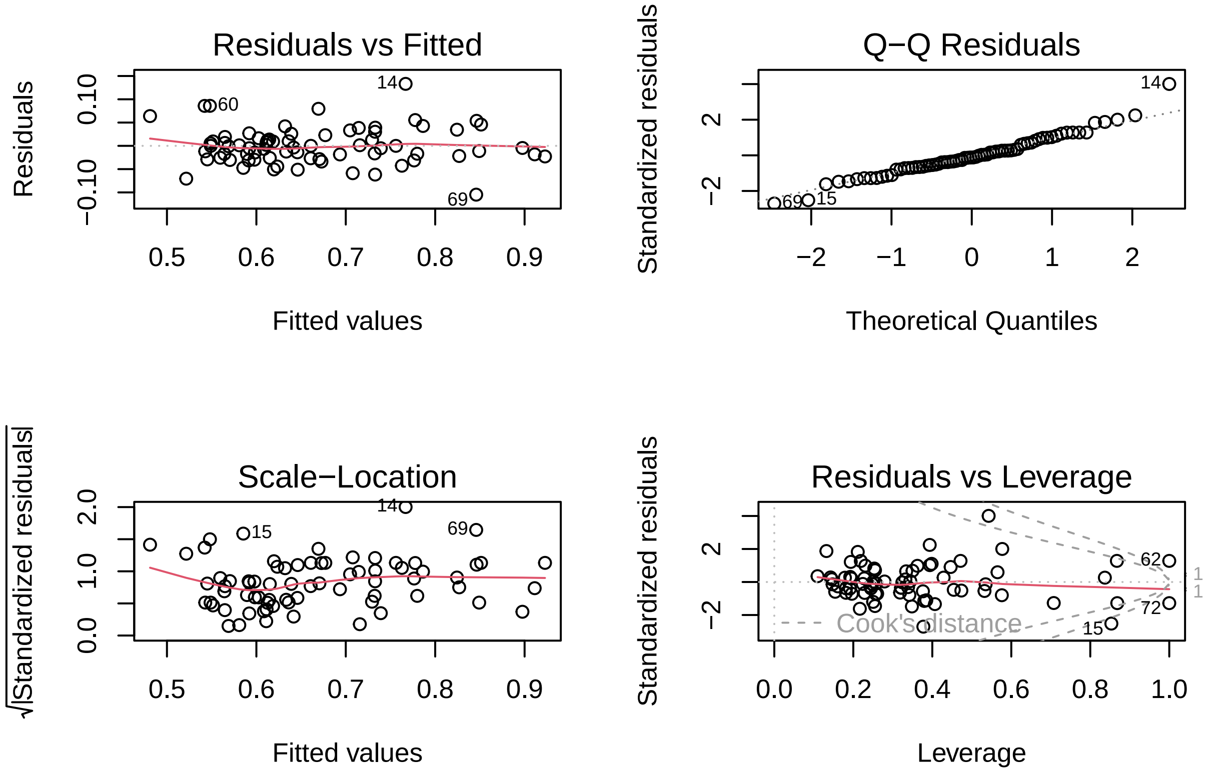
```
##
## Call:
## lm(formula = activity ~ ., data = pyrimidines_clean)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.104856 -0.025626 -0.003679  0.019134  0.133071
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.524804   0.057478   9.131 6.77e-12 ***
## p1.polar        -0.281121   0.161738  -1.738  0.08888 .
## p1.size          0.161732   0.090760   1.782  0.08136 .
## p1.flex         -0.209947   0.071895  -2.920  0.00540 **
## p1.h.doner      -0.120697   0.060931  -1.981  0.05360 .
## p1.h.acceptor    0.051868   0.058282   0.890  0.37813
## p1.pi.doner      0.044287   0.059004   0.751  0.45672
## p1.pi.acceptor   0.118463   0.091183   1.299  0.20035
## p1.polarisable   0.148911   0.065750   2.265  0.02828 *
## p1.sigma         0.276042   0.156579   1.763  0.08455 .
## p2.polar        -0.032748   0.210570  -0.156  0.87709
## p2.size          0.156501   0.080072   1.955  0.05673 .
## p2.flex         -0.235798   0.067746  -3.481  0.00111 **
## p2.h.doner       0.018055   0.057312   0.315  0.75417
## p2.h.acceptor    0.023309   0.037575   0.620  0.53810
## p2.pi.doner     -0.014209   0.063834  -0.223  0.82483
## p2.pi.acceptor  -0.009825   0.074666  -0.132  0.89589
## p2.polarisable   0.054130   0.050003   1.083  0.28465
## p2.sigma        -0.009310   0.185067  -0.050  0.96010
## p3.polar        -0.103299   0.180852  -0.571  0.57066
## p3.size          0.350239   0.131729   2.659  0.01075 *
## p3.flex         -0.100047   0.092197  -1.085  0.28351
## p3.h.doner       0.229888   0.222638   1.033  0.30721
## p3.h.acceptor  -0.284557   0.247213  -1.151  0.25565
## p3.pi.doner    -0.017501   0.367992  -0.048  0.96227
## p3.polarisable  0.021693   0.162123   0.134  0.89414
## p3.sigma        0.272428   0.218294   1.248  0.21835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04913 on 46 degrees of freedom
## Multiple R-squared:  0.8744, Adjusted R-squared:  0.8034
## F-statistic: 12.32 on 26 and 46 DF,  p-value: 3.404e-13
```

Adjusted R-squared is 0.8034. We check for violations using standard diagnostic plots.

```r
par(mfrow = c(2, 2))
plot(linear_model)
```

```
## Warning: not plotting observations with leverage one:
##   6
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



There are quite a few values with very high leverage. We can see this also in the QQ plot where they seem to not follow the diagonal line especially at the extremes of the theoretical quantiles. The residuals look ok without pattern.

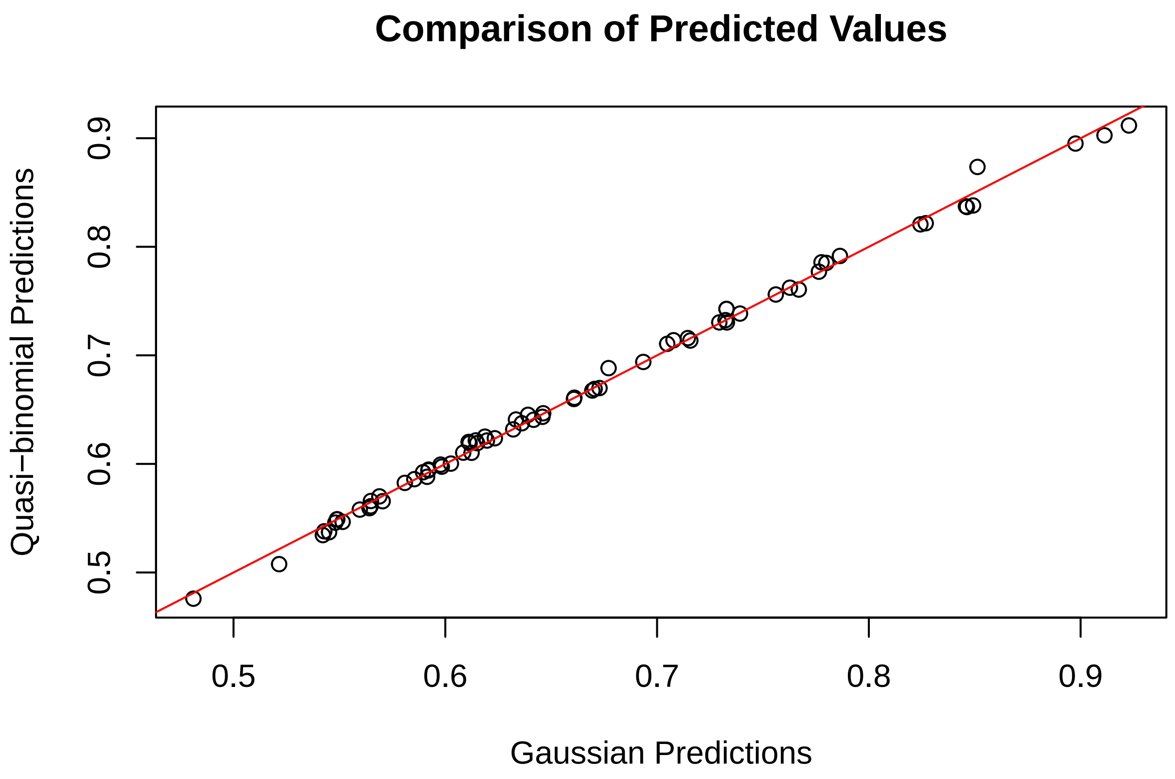
c.

> Fit a quasi-binomial model for the activity response. Compare the predicted values for this model to those for the Gaussian linear model. Take care to compute the predicted values in the appropriate scale. Compare the fitted coefficients between the two models. Are there any substantial differences?

```r
qb_model <- glm(activity ~ ., family = quasibinomial, data = pyrimidines_clean)
summary(qb_model)
```

```
##
## Call:
## glm(formula = activity ~ ., family = quasibinomial, data = pyrimidines_clean)
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.048962   0.282100   0.174  0.86297
## p1.polar       -1.329158   0.779548  -1.705  0.09493 .
## p1.size         0.705941   0.431162   1.637  0.10839
## p1.flex        -0.921347   0.341241  -2.700  0.00967 **
## p1.h.doner     -0.505915   0.285662  -1.771  0.08318 .
## p1.h.acceptor   0.245498   0.272139   0.902  0.37170
## p1.pi.doner     0.146646   0.274156   0.535  0.59530
## p1.pi.acceptor  0.506579   0.429023   1.181  0.24376
## p1.polarisable  0.689160   0.313959   2.195  0.03324 *
## p1.sigma        1.309291   0.759034   1.725  0.09125 .
## p2.polar       -0.201515   0.994763  -0.203  0.84036
## p2.size         0.768765   0.397785   1.933  0.05945 .
## p2.flex        -1.177539   0.339639  -3.467  0.00115 **
## p2.h.doner      0.063053   0.279551   0.226  0.82255
## p2.h.acceptor   0.122573   0.184027   0.666  0.50870
## p2.pi.doner    -0.069150   0.308130  -0.224  0.82343
## p2.pi.acceptor -0.046187   0.352325  -0.131  0.89627
## p2.polarisable  0.246687   0.241786   1.020  0.31294
## p2.sigma        0.001547   0.868954   0.002  0.99859
## p3.polar       -0.674515   0.910651  -0.741  0.46264
## p3.size         1.424434   0.690750   2.062  0.04487 *
## p3.flex        -0.375631   0.479780  -0.783  0.43768
## p3.h.doner      1.400621   1.102696   1.270  0.21041
## p3.h.acceptor  -1.615232   1.247061  -1.295  0.20170
## p3.pi.doner     0.391883   1.920217   0.204  0.83919
## p3.polarisable  0.319654   0.840840   0.380  0.70558
## p3.sigma        1.726704   1.237534   1.395  0.16963
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.01216917)
##
##     Null deviance: 4.32379  on 72  degrees of freedom
## Residual deviance: 0.57852  on 46  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```r
# Plot
preds_gaussian <- predict(linear_model)
preds_quasi <- predict(qb_model, type = "response")
plot(preds_gaussian, preds_quasi,
     xlab = "Gaussian Predictions",
     ylab = "Quasi-binomial Predictions",
     main = "Comparison of Predicted Values")
abline(0, 1, col = "red") # 45-degree line for perfect agreement
```



```r
coef_gaussian_logit <- coef(linear_model)
coef_quasibinomial <- coef(qb_model)
data.frame(
  linear_model = coef_gaussian_logit,
  Quasi_Binomial = coef_quasibinomial,
  Difference = coef_gaussian_logit - coef_quasibinomial
)
```

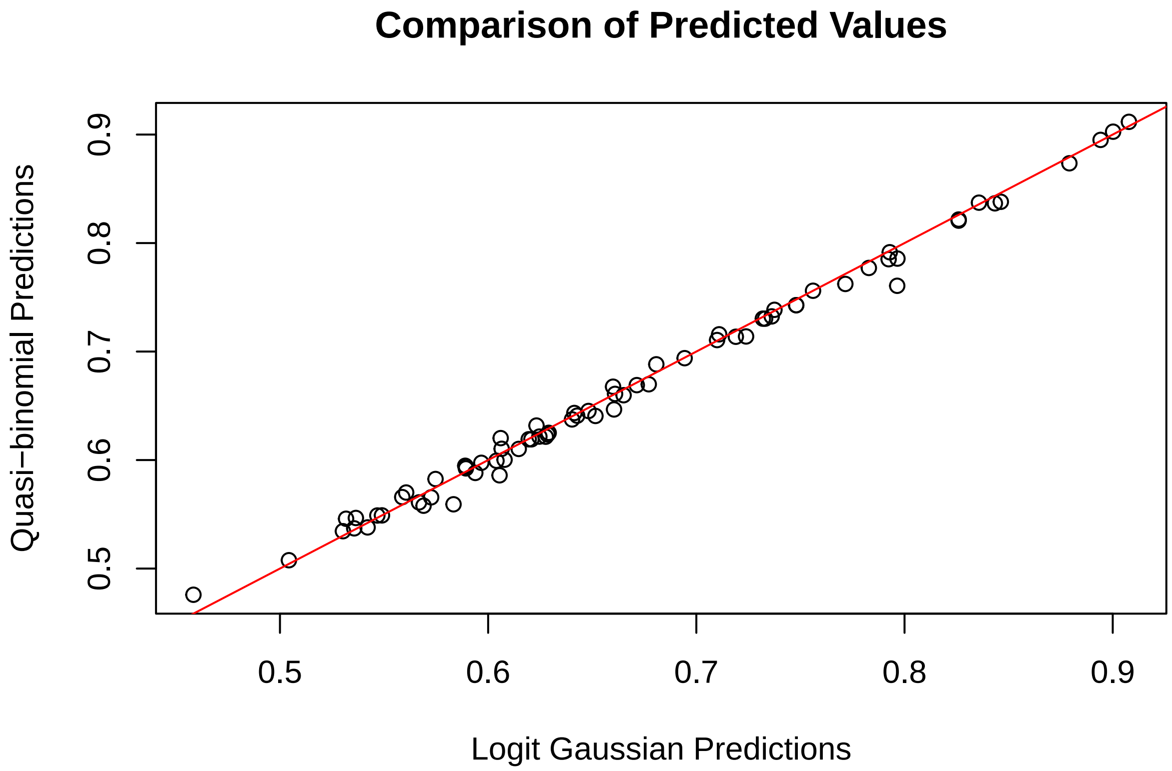
```
##                 linear_model Quasi_Binomial  Difference
## (Intercept)      0.524804275    0.048962369  0.47584191
## p1.polar        -0.281120741   -1.329157573  1.04803683
## p1.size          0.161731505    0.705940724 -0.54420922
## p1.flex         -0.209946799   -0.921346748  0.71139995
## p1.h.doner      -0.120697426   -0.505915258  0.38521783
## p1.h.acceptor    0.051867751    0.245498439 -0.19363069
## p1.pi.doner      0.044287497    0.146645765 -0.10235827
## p1.pi.acceptor  0.118463392    0.506578638 -0.38811525
## p1.polarisable  0.148911120    0.689160205 -0.54024908
## p1.sigma        0.276041636    1.309291083 -1.03324945
## p2.polar       -0.032747677   -0.201515157  0.16876748
## p2.size         0.156501145    0.768765406 -0.61226426
## p2.flex        -0.235797963   -1.177539001  0.94174104
## p2.h.doner      0.018054855    0.063053225 -0.04499837
## p2.h.acceptor   0.023308568    0.122573178 -0.09926461
## p2.pi.doner    -0.014209303   -0.069149820  0.05494052
## p2.pi.acceptor -0.009824673   -0.046186977  0.03636230
## p2.polarisable  0.054130072    0.246686843 -0.19255677
## p2.sigma       -0.009309822    0.001547109 -0.01085693
## p3.polar       -0.103298938   -0.674514823  0.57121589
## p3.size         0.350239140    1.424433709 -1.07419457
## p3.flex        -0.100047216   -0.375630879  0.27558366
## p3.h.doner      0.229887903    1.400620707 -1.17073280
## p3.h.acceptor  -0.284557236   -1.615232376  1.33067514
## p3.pi.doner    -0.017500839    0.391883385 -0.40938422
## p3.polarisable  0.021692990    0.319654140 -0.29796115
## p3.sigma        0.272428334    1.726704482 -1.45427615
```

The two almost predict exactly the same activity levels. Generally, the coefficients are not close at all. However, that is probably due to the model needing transformation to align properly with the output / coefficients of the general linear model.

d.

> Fit a Gaussian linear model with the logit transformation applied to the response. Compare the coefficients of this model with the quasi-binomial model.

```r
pyrimidines_clean$logit_activity <- log(pyrimidines_clean$activity / (1 - pyrimidines_clean$activity))
lmod_logit <- lm(logit_activity ~ . - activity, data = pyrimidines_clean)

# Plot
inv_logit <- function(x) exp(x) / (1 + exp(x))
preds_gaussian <- inv_logit(predict(lmod_logit))
preds_quasi <- predict(qb_model, type = "response")
plot(preds_gaussian, preds_quasi,
     xlab = "Logit Gaussian Predictions",
     ylab = "Quasi-binomial Predictions",
     main = "Comparison of Predicted Values")
abline(0, 1, col = "red") # 45-degree line for perfect agreement
```



```r
coef_gaussian_logit <- coef(lmod_logit)
coef_quasibinomial <- coef(qb_model)
data.frame(
  Gaussian_Logit = coef_gaussian_logit,
  Quasi_Binomial = coef_quasibinomial,
  Difference = coef_gaussian_logit - coef_quasibinomial
)
```

```
##                  Gaussian_Logit Quasi_Binomial   Difference
## (Intercept)         -0.01165804    0.048962369 -0.060620408
## p1.polar            -1.16892366   -1.329157573  0.160233913
## p1.size              0.63486884    0.705940724 -0.071071884
## p1.flex             -0.87269328   -0.921346748  0.048653467
## p1.h.doner          -0.58419889   -0.505915258 -0.078283637
## p1.h.acceptor        0.21863907    0.245498439 -0.026859371
## p1.pi.doner          0.24248577    0.146645765  0.095840009
## p1.pi.acceptor       0.42622674    0.506578638 -0.080351897
## p1.polarisable       0.60747443    0.689160205 -0.081685779
## p1.sigma             1.25706592    1.309291083 -0.052225165
## p2.polar             0.26019234   -0.201515157  0.461707494
## p2.size              0.85376463    0.768765406  0.084999219
## p2.flex             -1.25710634   -1.177539001 -0.079567339
## p2.h.doner           0.10220813    0.063053225  0.039154901
## p2.h.acceptor        0.06255480    0.122573178 -0.060018374
## p2.pi.doner         -0.03791027   -0.069149820  0.031239549
## p2.pi.acceptor      -0.18163092   -0.046186977 -0.135443941
## p2.polarisable     0.13935297    0.246686843 -0.107333870
## p2.sigma          -0.38654234    0.001547109 -0.388089446
## p3.polar          -0.48697852   -0.674514823  0.187536304
## p3.size            1.66320786    1.424433709  0.238774156
## p3.flex           -0.52463213   -0.375630879 -0.149001250
## p3.h.doner         1.37135815    1.400620707 -0.029262558
## p3.h.acceptor     -1.62290582   -1.615232376 -0.007673446
## p3.pi.doner        0.33670940    0.391883385 -0.055173982
## p3.polarisable     0.32654096    0.319654140  0.006886823
## p3.sigma           1.30913514    1.726704482 -0.417569338
```

The coefficients between the two are generally quite close. This is apparent when we see that the predicted values by both models are very similar also.

e.

> Fit a Beta regression model. Compare the coefficients of this model with that of logit response regression model.

```r
# Most of the code is kind of guesswork bc Prof Song didn't provide example code for this package
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.9-3. For overview type 'help("mgcv-package")'.
```

```r
# Get all predictor names (excluding activity and any temporary logit columns)
predictors <- names(pyrimidines)[!names(pyrimidines) %in% c("activity", "logit_activity")]
# Construct the formula: "activity ~ p1 + p2 + ... + p26"
gam_formula <- as.formula(paste("activity ~", paste(predictors, collapse = " + ")))
bgam <- gam(gam_formula,  family = betar(link="logit"), data = pyrimidines_clean)

beta_gauss <- coef(lmod_logit)[-1]
beta_gam   <- coef(bgam)[-1]
data.frame(
  Predictor = names(beta_gauss),
  Logit_Gaussian = round(beta_gauss, 4),
  Beta_GAM = round(beta_gam, 4),
  Difference = round(beta_gauss - beta_gam, 4)
)
```

```
##                      Predictor Logit_Gaussian Beta_GAM Difference
## p1.polar             p1.polar        -1.1689  -1.3008     0.1318
## p1.size               p1.size         0.6349   0.6820    -0.0471
## p1.flex               p1.flex        -0.8727  -0.8989     0.0262
## p1.h.doner         p1.h.doner        -0.5842  -0.5196    -0.0646
## p1.h.acceptor   p1.h.acceptor         0.2186   0.2439    -0.0253
## p1.pi.doner       p1.pi.doner         0.2425   0.1675     0.0750
## p1.pi.acceptor p1.pi.acceptor         0.4262   0.4857    -0.0594
## p1.polarisable p1.polarisable         0.6075   0.6759    -0.0684
## p1.sigma             p1.sigma         1.2571   1.3244    -0.0673
## p2.polar             p2.polar         0.2602  -0.2529     0.5131
## p2.size               p2.size         0.8538   0.7689     0.0848
## p2.flex               p2.flex        -1.2571  -1.1634    -0.0937
## p2.h.doner         p2.h.doner         0.1022   0.0630     0.0392
## p2.h.acceptor   p2.h.acceptor         0.0626   0.0876    -0.0251
## p2.pi.doner       p2.pi.doner        -0.0379  -0.0180    -0.0199
## p2.pi.acceptor p2.pi.acceptor        -0.1816  -0.0205    -0.1612
## p2.polarisable p2.polarisable         0.1394   0.2268    -0.0874
## p2.sigma             p2.sigma        -0.3865   0.0465    -0.4331
## p3.polar             p3.polar        -0.4870  -0.6459     0.1589
## p3.size               p3.size         1.6632   1.5301     0.1331
## p3.flex               p3.flex        -0.5246  -0.5169    -0.0077
## p3.h.doner         p3.h.doner         1.3714   1.2382     0.1332
## p3.h.acceptor   p3.h.acceptor        -1.6229  -1.4416    -0.1813
## p3.pi.doner       p3.pi.doner         0.3367   0.0555     0.2812
## p3.polarisable p3.polarisable         0.3265   0.4011    -0.0746
## p3.sigma             p3.sigma         1.3091   1.6794    -0.3702
```

The coefficients of this model is actually quite a bit similar to the logit response model.

f.

> What property of the response leads to the similarity of the models considered thus far in this question?

The similarities is probably because of the response variable. We are largely looking at responses as a ratio that is centered around 0.5 mostly, with rarely any variation below 0.2 (which we remove from our dataset). At the middle of this range, predicted activity level shouldn't be varying too much and with the fact that logit is also relatively linear around this area, we get that all our models (that use logit) do a very similar job as the linear one.

Problem 3

> Simulation study. Let’s now look at the two goodness of fit measures under the binomial distribution. For each setting, you will generate N = 1000 data sets, fit each using the correct binomial logistic regression model, and save the deviance and Pearson Xˆ2 measures. You will then compare the distribution of each measure against the appropriate asymptotic Xˆ2 density. You will use the following model in all simulations: y|x ~ Binomial(m, p) where x ~ N(0, 1) and logit(p)= 0.35 + x. For each setting and goodness-of-fit measure, create a histogram and overlay the appropriate Xˆ2 density. Summarize what you find, making sure to address whether the distributions of deviance and Pearson Xˆ2 better fit the Xˆ2 density as the number of trials and/or sample size increases.

```r
library(ggplot2)
library(tidyr)
library(dplyr)
```

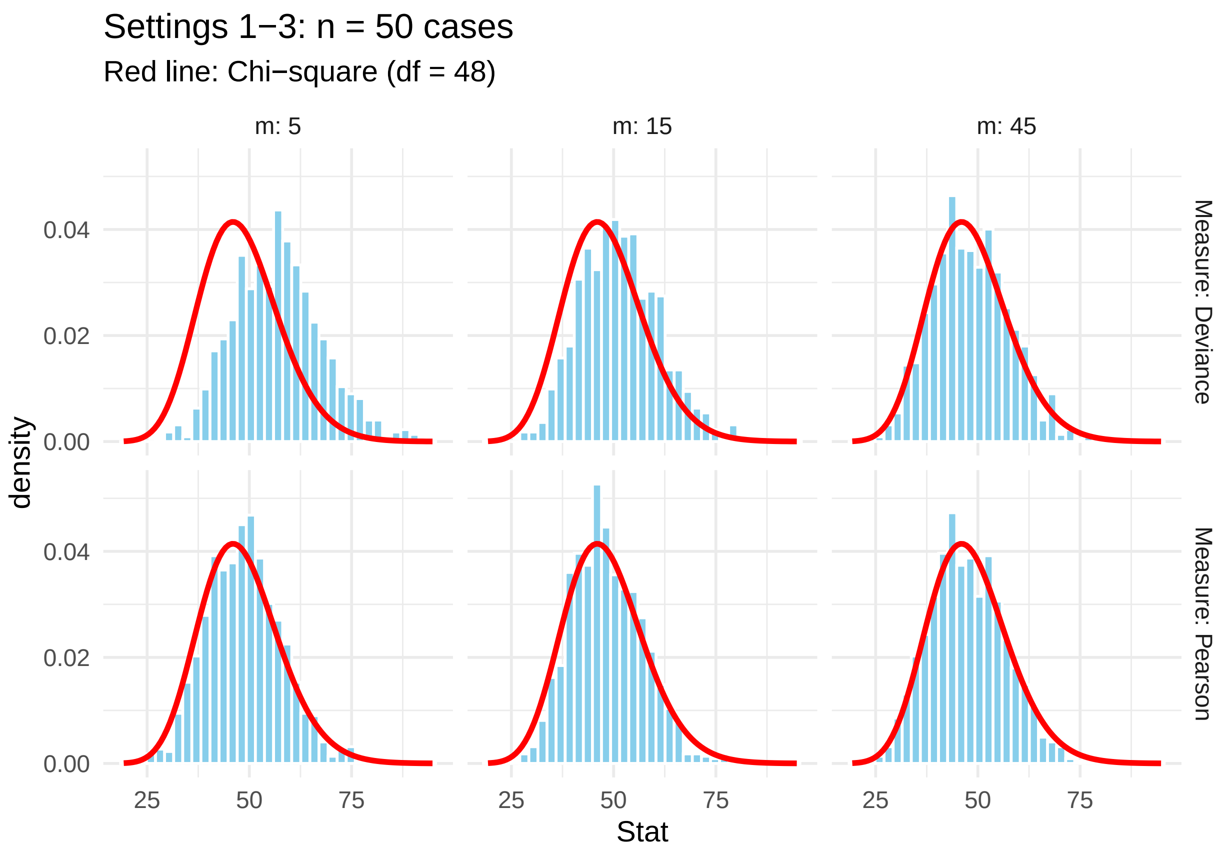
```
##
## Attaching package: ’dplyr’

## The following object is masked from ’package:nlme’:
##
##     collapse

## The following objects are masked from ’package:stats’:
##
##     filter, lag

## The following objects are masked from ’package:base’:
##
##     intersect, setdiff, setequal, union
```

```r
set.seed(42)

N_sims <- 1000
settings <- expand.grid(m = c(5, 15, 45), n = c(50, 200, 800))
results_list <- list()

# Loop through our 9 settings
for (i in 1:nrow(settings)) {
  m_val <- settings$m[i]
  n_val <- settings$n[i]

  deviance_vals <- numeric(N_sims)
  pearson_vals <- numeric(N_sims)

  for (j in 1:N_sims) {
    # 1. Generate Data
    x <- rnorm(n_val, 0, 1)
    logit_p <- 0.35 + x
    p <- 1 / (1 + exp(-logit_p))
    y <- rbinom(n_val, size = m_val, prob = p)
    # 2. Fit Model
    model <- glm(cbind(y, m_val - y) ~ x, family = binomial)

    # 3. Save Measures
    deviance_vals[j] <- deviance(model)
    pearson_vals[j] <- sum(residuals(model, type = "pearson")^2)
  }

  # Store results in a data frame
  results_list[[i]] <- data.frame(
    setting = paste0("m=", m_val, ", n=", n_val),
    m = m_val,
    n = n_val,
    Deviance = deviance_vals,
    Pearson = pearson_vals
  )
}

# Combine all results
all_results <- bind_rows(results_list)
```

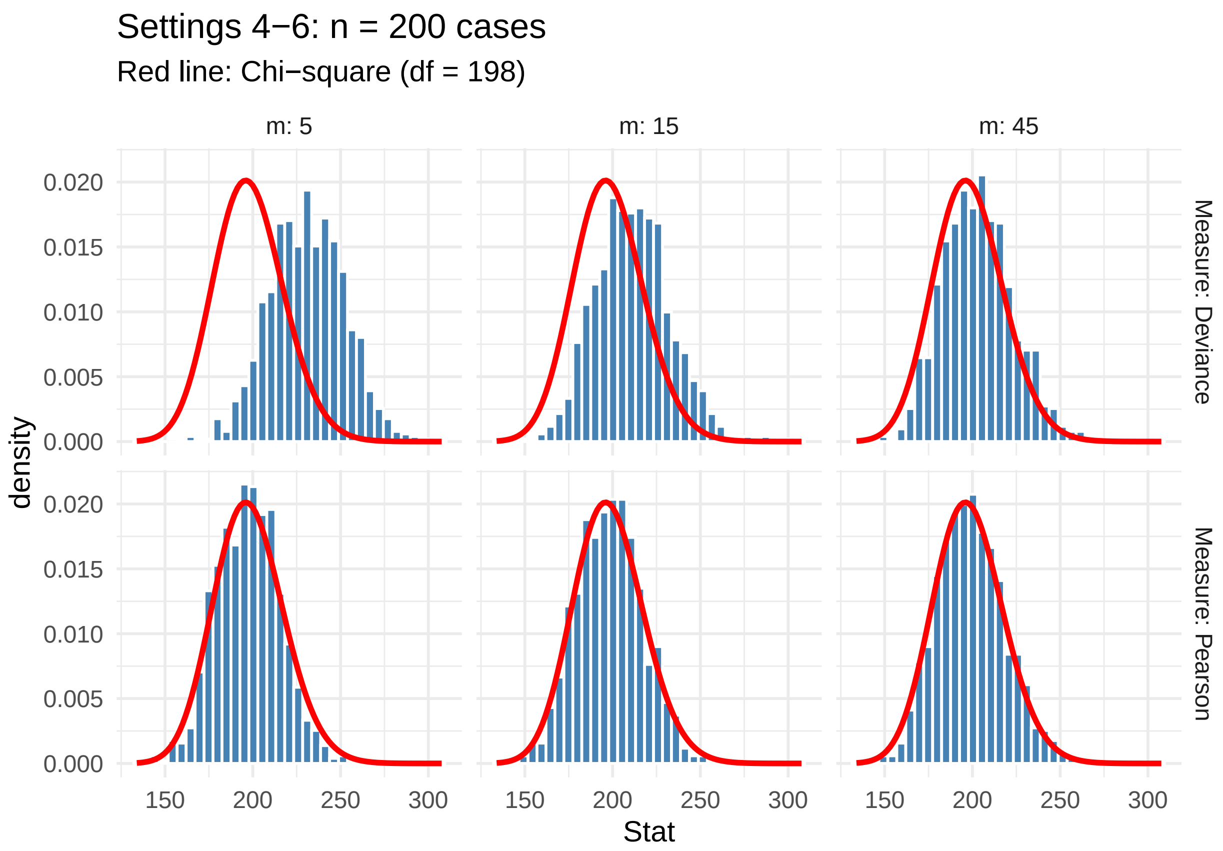
```r
# Reshape for plotting
all_data <- bind_rows(all_results) %>%
  pivot_longer(cols = c(Deviance, Pearson), names_to = "Measure", values_to = "Stat")

df_50 <- 50 - 2
ggplot(filter(all_data, n == 50), aes(x = Stat)) +
  geom_histogram(aes(y = ..density..), bins = 35, fill = "skyblue", color = "white") +
  stat_function(fun = dchisq, args = list(df = df_50), color = "red", size = 1) +
  facet_grid(Measure ~ m, labeller = label_both) +
  labs(title = "Settings 1-3: n = 50 cases", subtitle = "Red line: Chi-square (df = 48)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

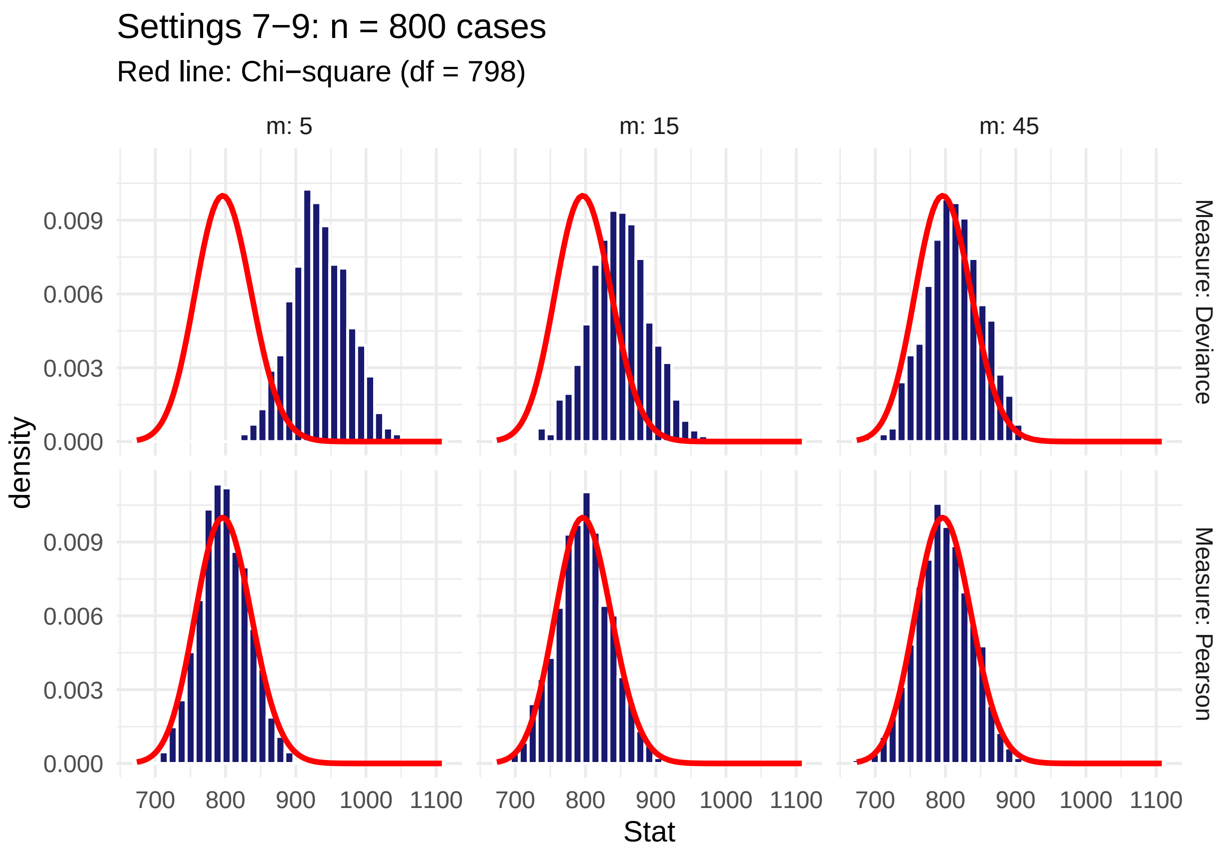
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
df_200 <- 200 - 2
ggplot(filter(all_data, n == 200), aes(x = Stat)) +
  geom_histogram(aes(y = ..density..), bins = 35, fill = "steelblue", color = "white") +
  stat_function(fun = dchisq, args = list(df = df_200), color = "red", size = 1) +
  facet_grid(Measure ~ m, labeller = label_both) +
  labs(title = "Settings 4-6: n = 200 cases", subtitle = "Red line: Chi-square (df = 198)") +
  theme_minimal()
```



```
df_800 <- 800 - 2
ggplot(filter(all_data, n == 800), aes(x = Stat)) +
  geom_histogram(aes(y = ..density..), bins = 35, fill = "midnightblue", color = "white") +
  stat_function(fun = dchisq, args = list(df = df_800), color = "red", size = 1) +
  facet_grid(Measure ~ m, labeller = label_both) +
  labs(title = "Settings 7-9: n = 800 cases", subtitle = "Red line: Chi-square (df = 798)") +
  theme_minimal()
```



It took a really long time to run all those loops. As we increase m, we better fit the Chi-squared distribution. Deviance is much worse at fitting the distribution at low m, almost shifted way off, but seems to do better as m increases. The squared pearson value matches pretty well, although at low m it seems to be a little squeezed or sharp. As we increase n, we see this effect much more.

Problem 4

An experiment analyzes imperfection rates for two processes used to fabricate silicon wafers for computer chips. For treatment A applied to 10 wafers, the numbers of imperfections are 8, 7, 6, 6, 3, 4, 7, 2, 3, 4. Treatment B applied to 10 other wafers has 9, 9, 8, 14, 8, 13, 11, 5, 7, 6 imperfections. The following model was fit to the data: (in hw)

a.

State the model and the assumptions. Denote the expected number of imperfections in treatment A as μ_A, and the expected number of imperfections in treatment B as μ_B. Provide the numeric estimate of $\frac{\mu_A}{\mu_B}$ based on the model fit above. Interpret the estimate.

Our model is a Poisson glm model, which usually uses the log link function to relate the mean / expectation of the number of imperfections to a linear model with one intercept and one treatment variable.

We can represent it as:

$$\ln(\mu_i) = \beta_0 + \beta_1 \cdot x_i$$

For a poisson distribution, we expect the expectation and variance of the response variable to be the same, iid of the wafers, and that the log of the counts can be predicted linearly.

Treatment variable x_i is either 0 or 1 depending on if treatment was provided. Then $\mu_A = e^{\hat{\beta}_0}$ and $\mu_B = e^{\hat{\beta}_0 + \hat{\beta}_1}$. We can then find:

$$\frac{\mu_B}{\mu_A} = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1}}{e^{\hat{\beta}_0}} = e^{\hat{\beta}_1} = e^{0.5878} \approx 1.80$$

Since it is a ratio, our interpretation is that the expected number of imperfections for treatment B is 1.8 times that of the expected number of imperfections for treatment A.

b.

Test H0: $\frac{\mu_A}{\mu_B} = 1$ against Ha: $\frac{\mu_A}{\mu_B} \neq 1$ using both the Wald test and the likelihood ratio test.

$$\begin{aligned}
\frac{\mu_A}{\mu_B} &= 1 \\
\ln(\frac{\mu_A}{\mu_B}) &= ln(1) = 0 \\
\ln(\mu_A) - \ln(\mu_B) &= 0 \\
\ln(\mu_A) &= \ln(\mu_B) \\
\beta_0 &= \beta_0 + \beta_1 \cdot x_i \\
0 &= \beta_1
\end{aligned}$$

We want to test if the coefficient for treatment B is 0:

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_a : \beta_1 \neq 0$$

We first do the Wald test. We use estimate ($\hat{\beta}_1$): 0.5878 and Standard Error (SE): 0.1764 to find Z score of

$$z = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} = \frac{0.5878}{0.1764} = 3.332$$

The associated p score to that z score is 0.000861 so at a 95% confidence level, we choose to reject the null hypothesis.

Next, we do the likelihood ratio test. We know "Null deviance: 27.857 on 19 degrees of freedom" and "Residual deviance: 16.268 on 18 degrees of freedom". We find that

$$\text{Null Deviance} - \text{Residual Deviance} = 27.857 - 16.268 = 11.589$$

The degrees of freedom for our test will be 19-18 = 1. We will test on a chi-squared distribution with 1 degree of freedom. We find $P(\chi_1^2 > 11.589) \approx 0.000663$ so we reject the null hypothesis at 95% confidence level.

Both tests reject the null hypothesis so there is strong evidence that the expected number of imperfections is not the same between treatment A and B.

c.

Construct a 95% confidence interval for $\frac{\mu_A}{\mu_B}$

We already did some math above to show that we are actually trying to solve for $\frac{\mu_B}{\mu_A} = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1}}{e^{\hat{\beta}_0}} = e^{\hat{\beta}_1}$. We first find a confidence level for β_1. We use the info from the problem $\hat{\beta}_1 = 0.5878$ and $SE(\hat{\beta}_1) = 0.1764$.

We use the Wald method $\hat{\beta}_1 \pm z_{0.025} \times SE(\hat{\beta}_1) = 0.5878 \pm 1.96 \times 0.1764$.

This gives us CI of [0.2421, 0.9335]. We take exp to these values and find [1.274, 2.544] is our 95% confidence interval for $\frac{\mu_A}{\mu_B}$