20MAP501_CW Assignment

1. Preamble

a. Load in any packages you will use to undertake this coursework, including AmesHousing. Use make_ames() command to create a clean version of the Ames Housing dataset called Ames. (5 points)

```
# import relevant packages
library(rio)
library(dplyr)
library(tidyr)
library(ggplot2)
library(pROC)
library(car)
library(nnet)
library(AmesHousing)
library(tidyverse)
library(lme4)
library(caret)
```

```
# create clean ames housing dataset
ames <- make_ames()
```

b. Import the england-premier-league-players-2018-to-2019-stats.csv dataset as "playerstats", ensuring that all variables are treated correctly. Where variable names have a space, rename the variables without these. Remove the cases with age<10 Create a new variable, "played", in playerstats that indicates if a player played greater than 0 minutes or not. Create a new dataset called "football" consisting of only those players that played greater than 0 minutes. Create a new dataset called "footballngk" consisting of those players in "football" who are not goalkeepers. Drop the unused level of "position". (5 points)

```
# load in football data
playerstats <- import("england-premier-league-players-2018-to-2019-stats.csv")
```

```
# remove spaces in variables
playerstats$club <- playerstats$`Current Club`
playerstats$`Current Club` <- NULL
```

```
# remove rows where age < 10
playerstats <- playerstats %>%
  filter(age > 10)
```

```
# create 'played' variable
playerstats$played <- (playerstats$minutes_played_overall > 0)
```

```r
# create new dataset with only players who played
football <- playerstats %>%
  filter(played==TRUE)
```

```r
# create new dataset without goalkeepers
footballngk <- football %>%
  filter(position != "Goalkeeper")
```

```r
# chaneg "position" to factor
footballngk$position<-as.factor(footballngk$position)

# drop the unused level of "position"
footballngk %>%
  droplevels()
```

```r
# check level of goalkeeper has been dropped
levels(footballngk$position)
```

```
## [1] "Defender"   "Forward"    "Midfielder"
```

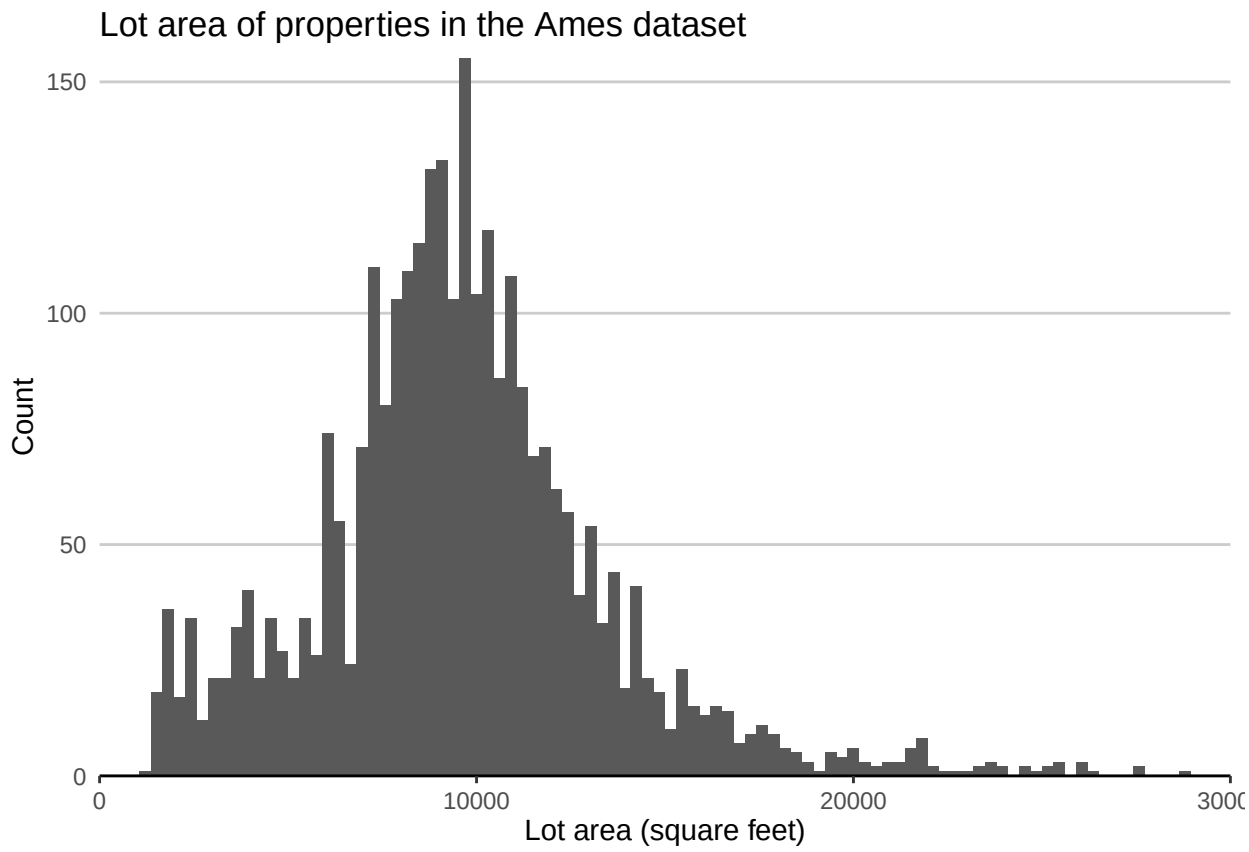
2. Linear Regression

a. By adjusting x axis range and number of bars, create a useful histogram of Lot_Area on the full dataset. Ensure that plot titles and axis labels are clear. Comment on why deleting properties with Lot_Area> 30000 makes sense. Create a new dataset Ames2 consisting only of those properties with lot area <30000. (5 points)

```r
# set global plot theme
old <- theme_set(theme_classic())
theme_set(old)
theme_update(
  panel.grid.major.x = element_blank(),
  panel.grid.minor.x = element_blank(),
  panel.grid.major.y = element_line(colour = "grey80"),
  panel.grid.minor.y = element_blank(),
  panel.background   = element_blank(),
  axis.line.y  = element_blank(),
  axis.line.x  = element_line(colour = "black"),
  axis.ticks.y = element_blank(),
  axis.text.x  = element_text(
                    angle = 0,
                    hjust = 0.5,
                    vjust = 0.5))
```

```r
# create plot object
p1 <- ames %>%
  ggplot(
    aes(
      x = Lot_Area
    ))
```

```r
# make histogram
p1 + geom_histogram(bins=100) +
  # add titles etc
  labs(
    title = "Lot area of properties in the Ames dataset",
    x = "Lot area (square feet)",
    y = "Count"
  ) +
  # improve scaling
  scale_x_continuous(limits = c(0,30000), expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))
```



Removing properties with lot area < 30000 makes sense because there are few properties with lot area this large, and some of these larger properties have lot areas much larger than 30000. The bulk of the data lies roughly between 0 and 30000, and so including the data out of this range is likely to skew results.

```r
# create a new dataset Ames2 consisting only of those properties with lot area <30000
ames2 <- ames %>%
  filter(Lot_Area < 30000)
```

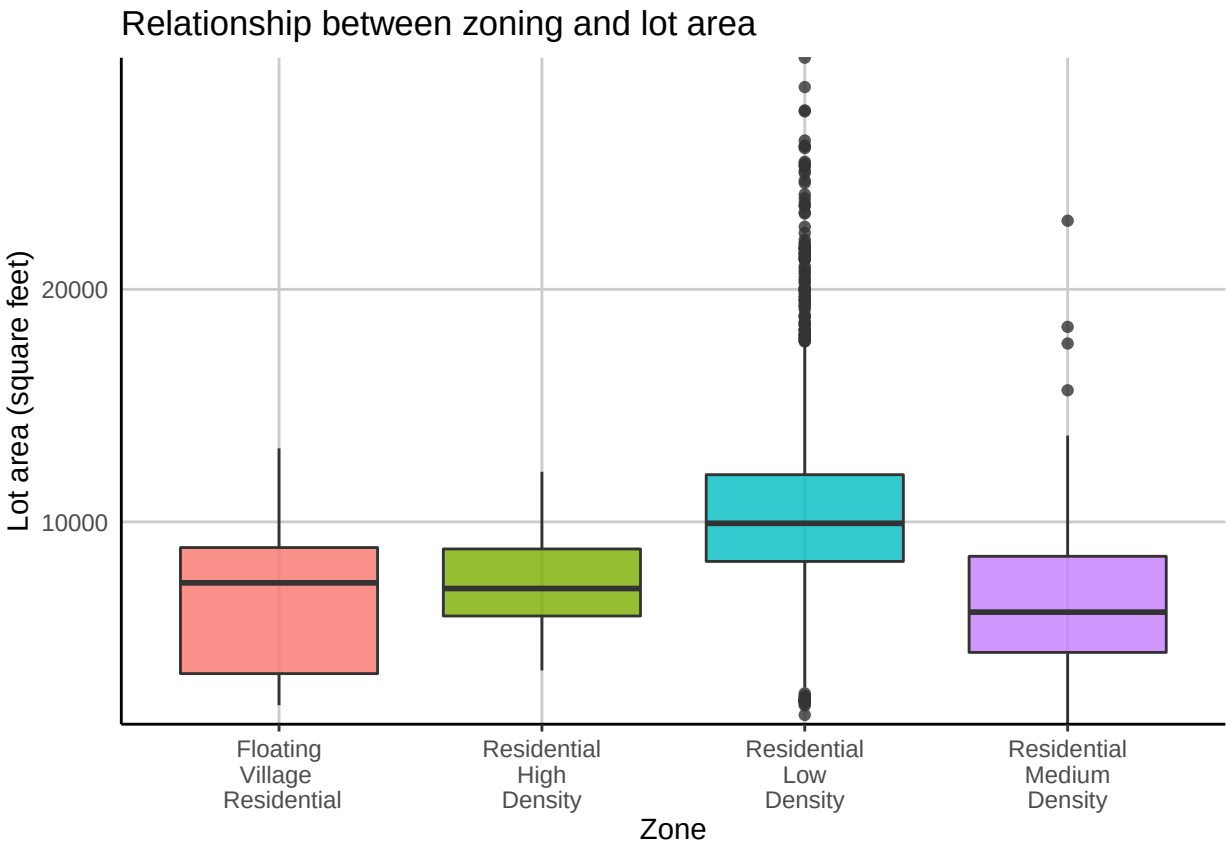
b. Now remove all cases corresponding to MS_Zoning categories of A_agr (agricultural), C_all (commercial) and I_all (industrial) from the Ames2 dataset. Drop the unused levels from the MS_Zoning variable. (2 points)

```r
# filter cases
ames2 <- ames2[!(ames2$MS_Zoning %in% c("A_agr","C_all","I_all")),]

# drop the unused levels
ames2 <- ames2 %>%
  droplevels()

# check levels have been dropped
levels(ames2$MS_Zoning)
```

```
## [1] "Floating_Village_Residential" "Residential_High_Density"
## [3] "Residential_Low_Density"      "Residential_Medium_Density"
```

c. Choose an appropriate plot to investigate the relationship between MS_Zoning and Lot_Area in Ames2. (2 points)

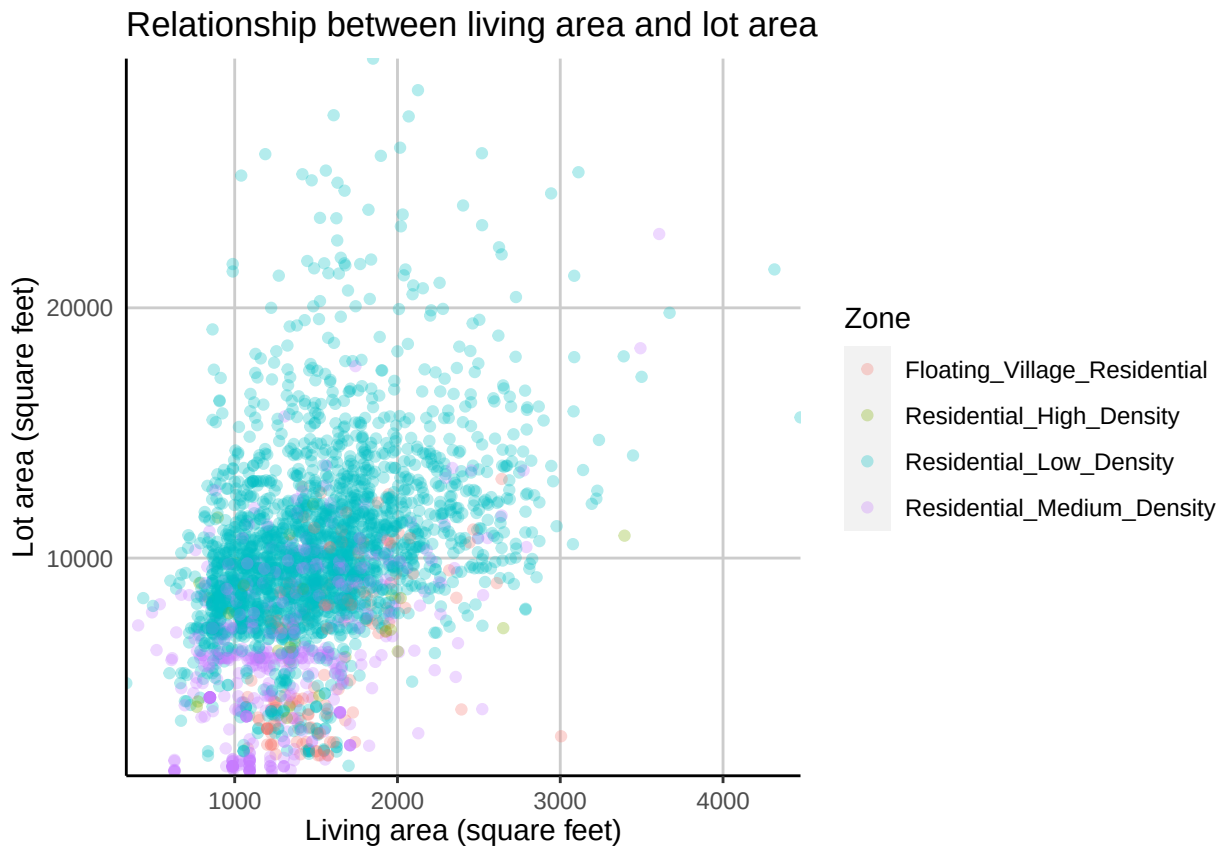
```r
# create plot object
p2 <- ames2 %>%
  ggplot(
    mapping = aes(
      x = MS_Zoning,
      y = Lot_Area,
    ),
    width= 2,
    height =2
  )

# add points
p2 +
  # add box plot
  geom_boxplot(mapping=aes(fill=MS_Zoning), alpha=0.8) +
  # add titles etc
  labs(
    title = "Relationship between zoning and lot area",
    x = "Zone",
    y = "Lot area (square feet)"
  ) +
  # improve scaling
  scale_y_continuous(expand = c(0, 0)) +
  # improve aesthetics
  theme(
    panel.grid.major.x  = element_line(colour = "grey80"),
    axis.line.y  = element_line(colour = "black"),
    legend.position = "none"
  ) +
  # change axis labels
  scale_x_discrete(labels=c("Floating_Village_Residential" = "Floating\nVillage \n Residential",
                            "Residential_High_Density" = "Residential\nHigh\nDensity",
                            "Residential_Low_Density" = "Residential\nLow\nDensity",
                            "Residential_Medium_Density" = "Residential\nMedium\nDensity"))
```



d. Choose an appropriate plot to investigate the relationship between Gr_Liv_Area and Lot_Area in Ames2. Color points according to the factor MS_Zoning. Ensure your plot has a clear title, axis labels and legend. (4 points)

```
# create plot object
p3 <- ames2 %>%
  ggplot(
    mapping =(aes(
      x = Gr_Liv_Area,
      y = Lot_Area,
      col = MS_Zoning,
    ))
  )

# make histogram
p3 + geom_point(alpha = 0.3) +
  # add titles etc
  labs(
    title = "Relationship between living area and lot area",
    x = "Living area (square feet)",
    y = "Lot area (square feet)"
  ) +
  # improve scaling
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  # improve aesthetics
```

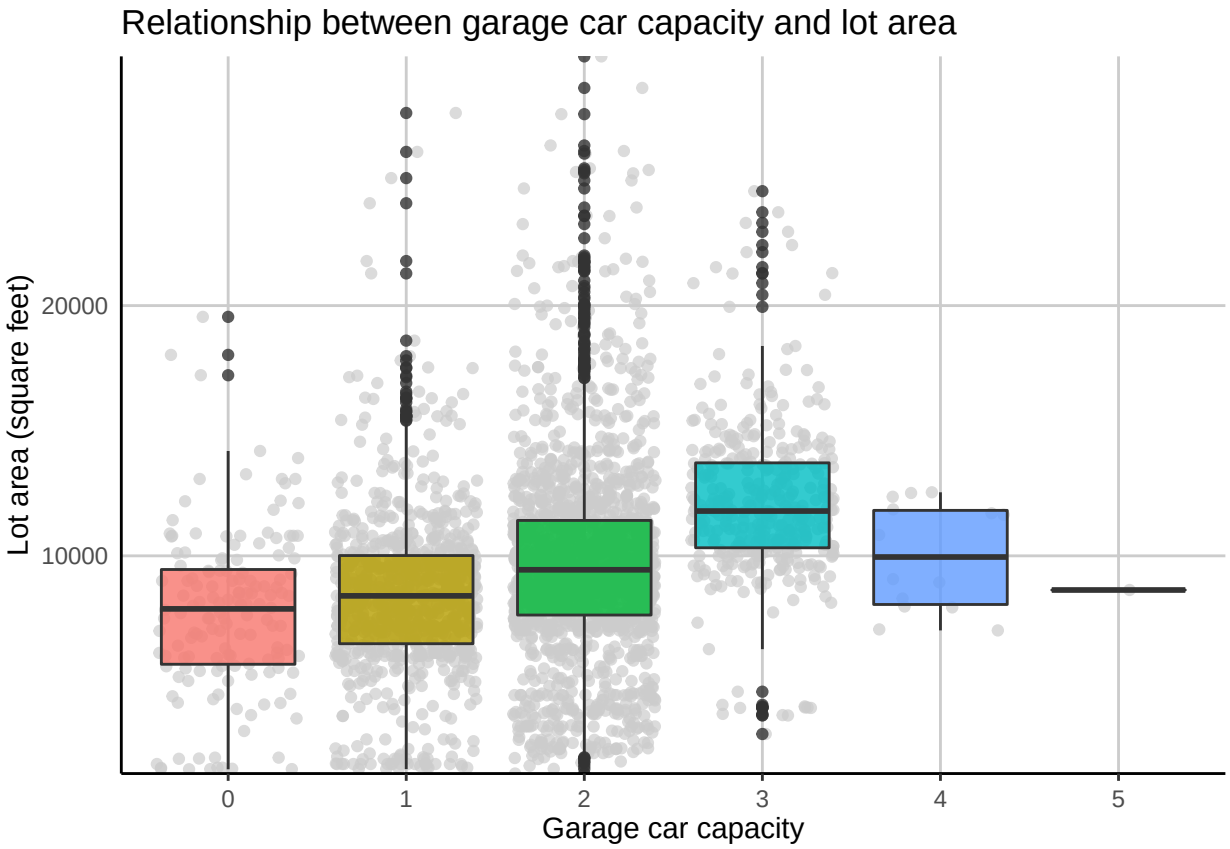
```
  theme(
    panel.grid.major.x  = element_line(colour = "grey80"),
    axis.line.y  = element_line(colour = "black")
  ) +
  labs(color='Zone')
```



e. Choose an appropriate plot to investigate the relationship between Garage_Cars and Lot_Area in Ames2. Use the "jitter" command to make your plot more clear. (3 points)

```
# change garage_cars to factor
#ames2$Garage_Cars <- as.factor(ames2$Garage_Cars)

# create plot object
p4 <- ames2 %>%
  ggplot(
    mapping = aes(
      x = as.factor(Garage_Cars),
      y = Lot_Area,
    )
  )

# add points
p4 + geom_jitter(alpha=0.7, colour = "grey80") +
  # add box plot
  geom_boxplot(mapping=aes(fill=as.factor(Garage_Cars)),alpha=0.8) +
  # add titles etc
```

```
  labs(
    title = "Relationship between garage car capacity and lot area",
    x = "Garage car capacity",
    y = "Lot area (square feet)"
  ) +
  # improve scaling
  scale_y_continuous(expand = c(0, 0)) +
  # improve aesthetics
  theme(
    panel.grid.major.x  = element_line(colour = "grey80"),
    axis.line.y  = element_line(colour = "black"),
    legend.position = "none"
  )
```



f. Why do we make these plots? Comment on your findings from these plots (1 sentence is fine). (2 points)

These plots are useful for exploring the data, so that we can see any outliers as well as seeing if there are any relationships that are worth investigating.

g. Use the lm command to build a linear model, linmod1, of Lot_Area as a function of the predictors MS_Zoning and Gr_Liv_Area for the Ames2 dataset. Evaluate the assumptions of the model. (5 points)

```
# build model
linmod1 <- lm(Lot_Area ~ MS_Zoning + Gr_Liv_Area,
```

```
              family=gaussian,
              data=ames2)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

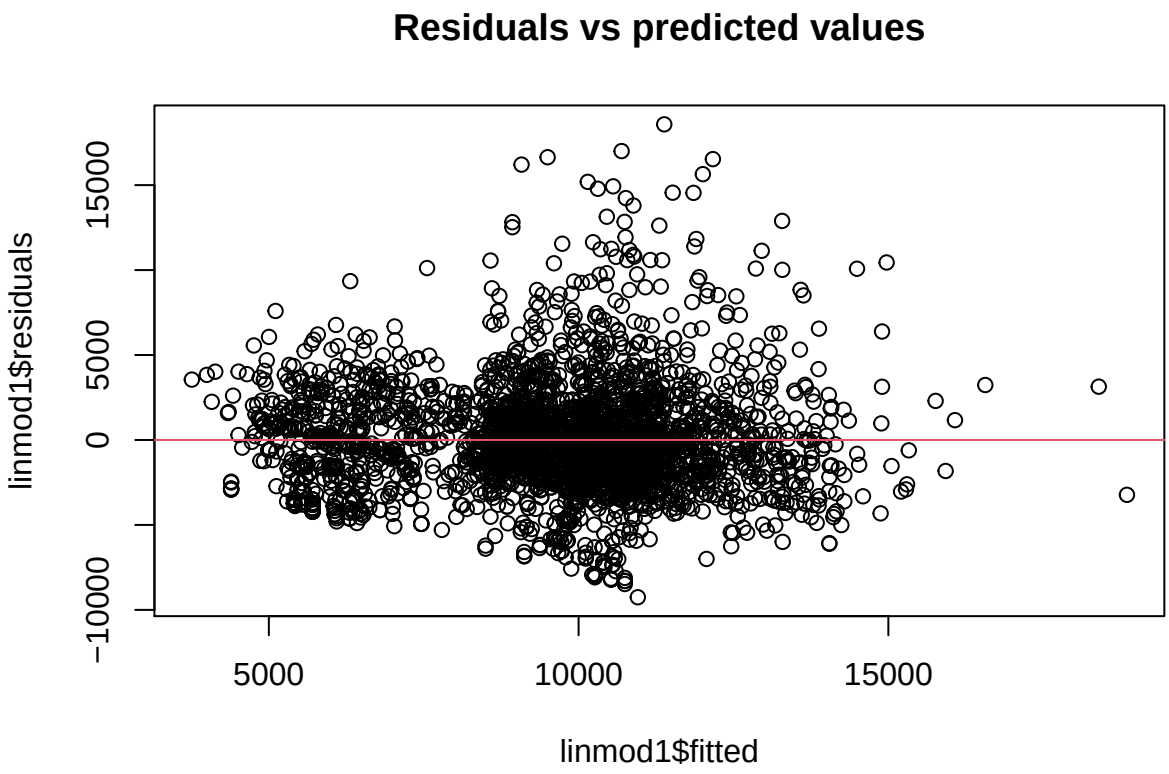
```
# view model
linmod1
```

```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area, data = ames2,
##     family = gaussian)
##
## Coefficients:
##                           (Intercept)    MS_ZoningResidential_High_Density
##                             2051.053                             1123.023
##    MS_ZoningResidential_Low_Density  MS_ZoningResidential_Medium_Density
##                             4069.855                              548.298
##                           Gr_Liv_Area
##                                 2.844
```

```
summary(linmod1)
```

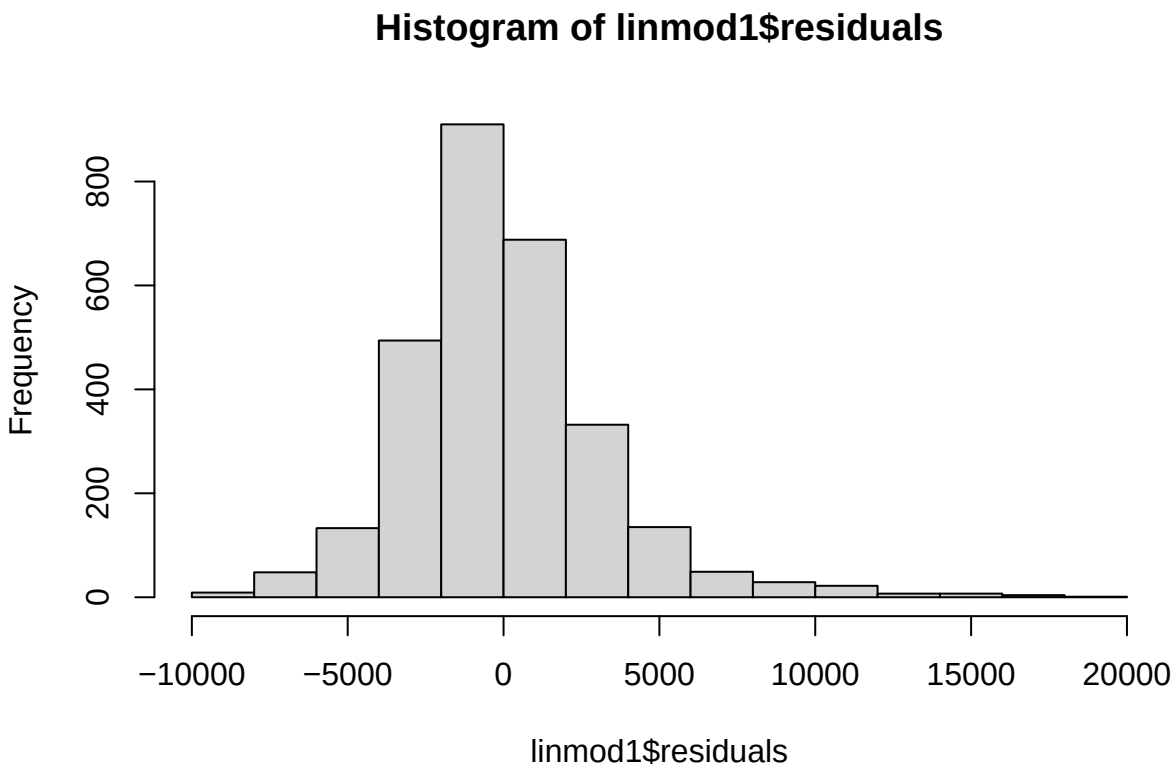
```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area, data = ames2,
##     family = gaussian)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -9256  -1905   -362   1508  18576
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        2051.0533   337.5074   6.077 1.39e-09 ***
## MS_ZoningResidential_High_Density  1123.0227   677.8742   1.657   0.0977 .
## MS_ZoningResidential_Low_Density   4069.8548   281.7687  14.444  < 2e-16 ***
## MS_ZoningResidential_Medium_Density  548.2983   313.5425   1.749   0.0804 .
## Gr_Liv_Area                            2.8442     0.1252  22.719  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3223 on 2863 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.3081
## F-statistic: 320.2 on 4 and 2863 DF,  p-value: < 2.2e-16
```

```
# evaluate suitability of model

# plot residuals vs lot area
plot(linmod1$fitted,linmod1$residuals, main="Residuals vs predicted values")
abline(h=0,col=2)
```



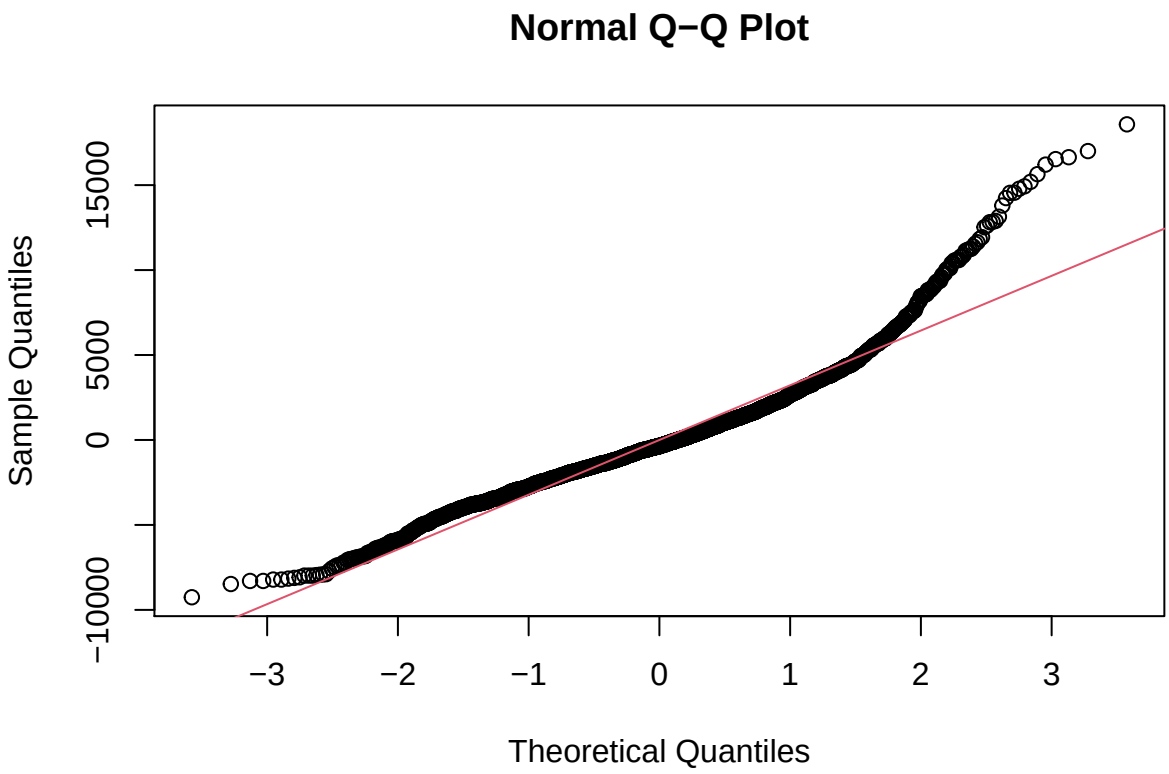
This plot shows that the residuals are spread pretty evenly around the zero line.

```
# further evaluation

# do histogram of resisuals
hist(linmod1$residuals)
```



The histogram of the residuals does look roughly like a guassian curve. It is a bit skewed though, with a longer tail on the right hand side.

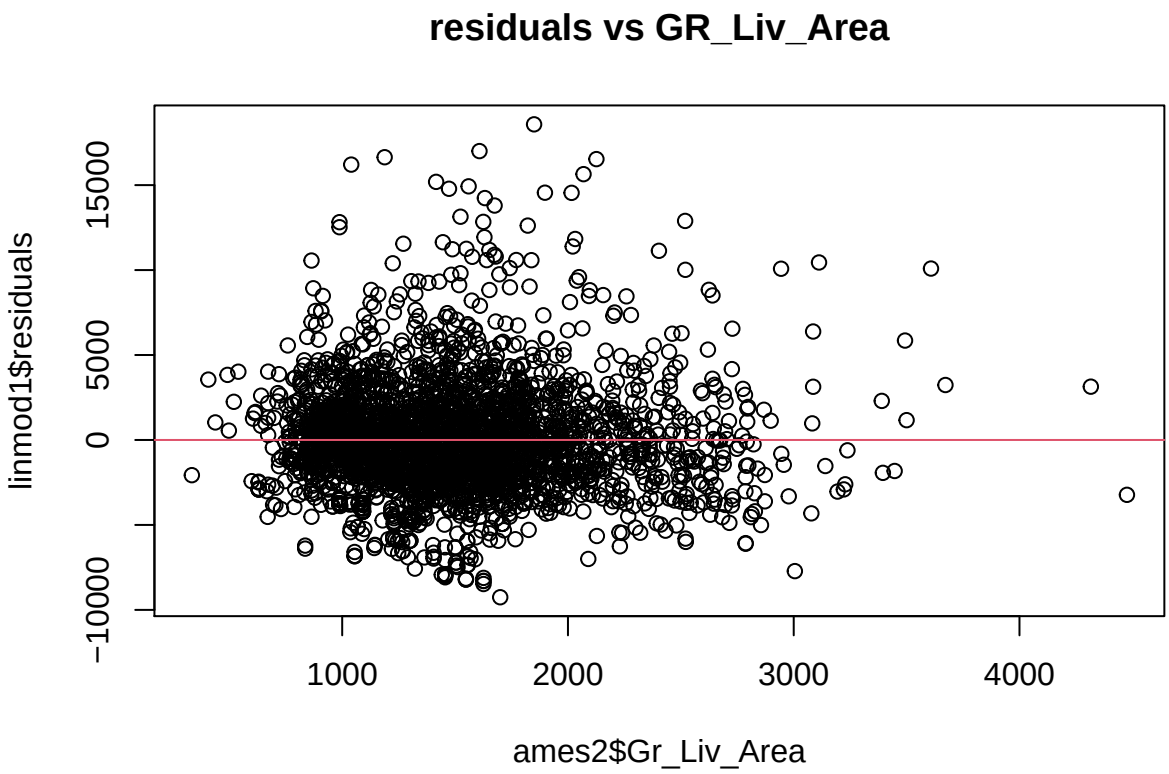
```
# do qq plot
qqnorm(linmod1$residual)

# add line with sd of residuals
abline(0,sd(linmod1$residuals),col=2)
```



The normal q-q plot looks like somewhat of a straight line in the middle, but at either end it does go a bit wonky, especially on the right hand side.

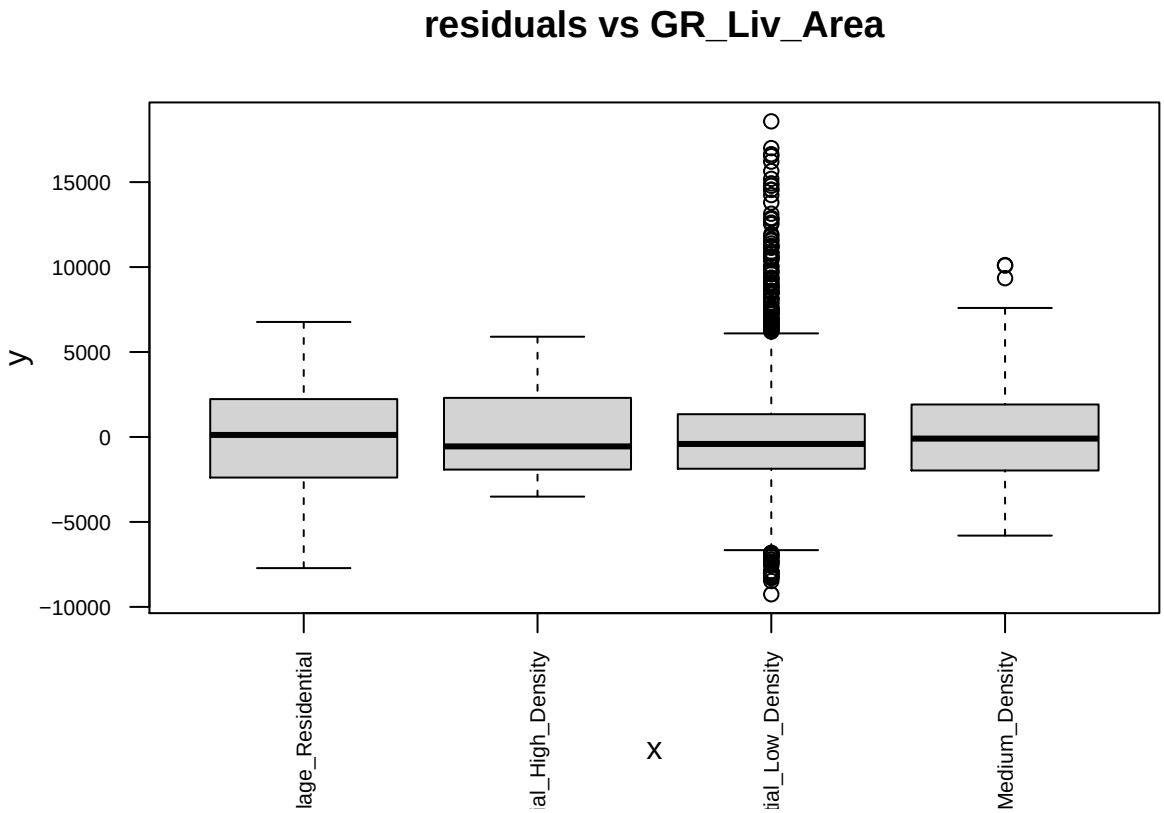
To investigate this further, we should try plotting the residuals against Gr_Liv_Area and against the different levels of MS_Zoning.

```
# plot residuals vs GR_Liv_Area
plot(ames2$Gr_Liv_Area,linmod1$residuals,main="residuals vs GR_Liv_Area")
abline(h=0,col=2)
```



This looks decent again, though the residuals are larger above the zero line than below.

```
plot(ames2$MS_Zoning,linmod1$residuals,main="residuals vs GR_Liv_Area", las=2, cex.axis=0.7)
```



This graph shows that the spread of residuals is pretty even, apart from for Residential_Low_Density. Predictions using this factor level will be less accurate than with other levels.

Overall, the model is pretty good, though there may be some issues when predicting properties with MS_Zoning == Residential_Low_Density.

h. Use the lm command to build a second linear model, linmod2, for Lot_Area as a function of MS_Zoning, Gr_Liv_Area and Garage_Cars. (2 points)

```
# build model
linmod2 <- lm(Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars,
              family=gaussian,
              data=ames2)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

```
# view models
linmod2
```

```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars,
##     data = ames2, family = gaussian)
##
## Coefficients:
```

```
##                            (Intercept)    MS_ZoningResidential_High_Density
##                               1759.224                              1355.811
##    MS_ZoningResidential_Low_Density  MS_ZoningResidential_Medium_Density
##                               4141.499                               725.008
##                           Gr_Liv_Area                          Garage_Cars
##                                 2.631                               296.727
```

```
summary(linmod2)
```

```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars,
##     data = ames2, family = gaussian)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8969.4 -1916.6  -344.4  1509.3 18598.3
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       1759.2236   349.4869   5.034 5.11e-07 ***
## MS_ZoningResidential_High_Density   1355.8111   680.8434   1.991  0.04654 *
## MS_ZoningResidential_Low_Density    4141.4993   282.2487  14.673  < 2e-16 ***
## MS_ZoningResidential_Medium_Density  725.0078   318.0421   2.280  0.02270 *
## Gr_Liv_Area                          2.6305     0.1422  18.497  < 2e-16 ***
## Garage_Cars                         296.7266    94.2029   3.150  0.00165 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3218 on 2862 degrees of freedom
## Multiple R-squared:  0.3115, Adjusted R-squared:  0.3103
## F-statistic: 258.9 on 5 and 2862 DF,  p-value: < 2.2e-16
```

i. Use Anova and Adjusted R-squared to compare these two models, and decide which is a better model. (3 points)

```
# anova for first model
Anova(linmod1)
```

```
## Anova Table (Type II tests)
##
## Response: Lot_Area
##                 Sum Sq   Df F value    Pr(>F)
## MS_Zoning   6.3775e+09    3  204.71 < 2.2e-16 ***
## Gr_Liv_Area 5.3603e+09    1  516.17 < 2.2e-16 ***
## Residuals   2.9731e+10 2863
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# anova for second model
Anova(linmod2)
```

```
## Anova Table (Type II tests)
##
## Response: Lot_Area
##                 Sum Sq   Df  F value  Pr(>F)
## MS_Zoning   6.0796e+09    3 195.7549 < 2e-16 ***
## Gr_Liv_Area 3.5421e+09    1 342.1513 < 2e-16 ***
## Garage_Cars 1.0271e+08    1   9.9217 0.00165 **
## Residuals   2.9629e+10 2862
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the ANOVA we see that, in both models, MS_Zoning and Gr_Liv_Area are strongly significant predictors of Lot_Area. In the second model, Garage_Cars is also a significant predictor of Lot_Area, though not as strong as the other two variables. The residuals have been reduced slightly also, though not by a great deal.

```
# summarise models again to get r^2 values
summary(linmod1)
```

```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area, data = ames2,
##     family = gaussian)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -9256  -1905   -362   1508  18576
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          2051.0533   337.5074   6.077 1.39e-09 ***
## MS_ZoningResidential_High_Density    1123.0227   677.8742   1.657   0.0977 .
## MS_ZoningResidential_Low_Density     4069.8548   281.7687  14.444  < 2e-16 ***
## MS_ZoningResidential_Medium_Density   548.2983   313.5425   1.749   0.0804 .
## Gr_Liv_Area                             2.8442     0.1252  22.719  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3223 on 2863 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.3081
## F-statistic: 320.2 on 4 and 2863 DF,  p-value: < 2.2e-16
```

```
summary(linmod2)
```

```
##
## Call:
## lm(formula = Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars,
##     data = ames2, family = gaussian)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8969.4 -1916.6  -344.4  1509.3 18598.3
```

```
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      1759.2236   349.4869   5.034 5.11e-07 ***
## MS_ZoningResidential_High_Density  1355.8111   680.8434   1.991  0.04654 *
## MS_ZoningResidential_Low_Density   4141.4993   282.2487  14.673  < 2e-16 ***
## MS_ZoningResidential_Medium_Density  725.0078   318.0421   2.280  0.02270 *
## Gr_Liv_Area                          2.6305     0.1422  18.497  < 2e-16 ***
## Garage_Cars                        296.7266    94.2029   3.150  0.00165 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3218 on 2862 degrees of freedom
## Multiple R-squared:  0.3115, Adjusted R-squared:  0.3103
## F-statistic: 258.9 on 5 and 2862 DF,  p-value: < 2.2e-16
```

The first model has an adjusted R-squared of 0.3081, and so explains **30.8%** of variance.

The second model has an adjusted R-squared of 0.3103, and so explains **31.0%** of variance.

The second model is slightly better, though the difference between the models is almost negligible.

j. Construct a confidence interval and a prediction interval for the lot area of a residential property in the High Density zone, with a 2 car garage and a living area of 1000 sq ft. Explain what these two intervals mean. (4 points)

```
confint(linmod2)
```

```
##                                        2.5 %      97.5 %
## (Intercept)                      1073.95212 2444.495025
## MS_ZoningResidential_High_Density    20.81793 2690.804360
## MS_ZoningResidential_Low_Density   3588.06799 4694.930539
## MS_ZoningResidential_Medium_Density  101.39306 1348.622509
## Gr_Liv_Area                          2.35170    2.909398
## Garage_Cars                        112.01422  481.438887
```

```
# get confidence intervals
predict(linmod2,
        data.frame(MS_Zoning="Residential_High_Density",
                           Gr_Liv_Area=1000,
                           Garage_Cars=2), interval="confidence")
```

```
##        fit      lwr      upr
## 1 6339.037 5103.145 7574.928
```

```
# get prediction intervals
predict(linmod2,
        data.frame(MS_Zoning="Residential_High_Density",
                           Gr_Liv_Area=1000,
                           Garage_Cars=2), interval="prediction")
```

```
##        fit    lwr      upr
## 1 6339.037 -89.76 12767.83
```

The 95% confidence interval for this predicted lot area is 5103 to 7575 square feet. This means that there is a 95% chance that the average property in the High Density zone, with a 2 car garage, and a living area of 1000 sq ft will have a lot area between 5103 and 7575 square feet.

The 95% prediction interval for this predicted lot area is -89.76 to 12770 square feet. This means that 95% of properties in the High Density zone, with a 2 car garage, and a living area of 1000 sq ft will have a lot area between -89.76 and 12770 square feet. Obviously, it is impossible to have a negative lot area, so this model has some room for improvement.

k. Now use the lmer function to build a third model, mmod1, for Lot_Area as a function of zoning, living area, garage size and neighborhood. What is the critical number to pull out from this, and what does it tell us? (3 points)

```
# create model
mmod1 <- lmer(Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars +
                (1|Neighborhood), data=ames2)

# view model
mmod1
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Lot_Area ~ MS_Zoning + Gr_Liv_Area + Garage_Cars + (1 | Neighborhood)
##    Data: ames2
## REML criterion at convergence: 53876.11
## Random effects:
##  Groups       Name        Std.Dev.
##  Neighborhood (Intercept) 2632
##  Residual                 2884
## Number of obs: 2868, groups:  Neighborhood, 28
## Fixed Effects:
##                          (Intercept)    MS_ZoningResidential_High_Density
##                              743.339                              2038.254
##    MS_ZoningResidential_Low_Density  MS_ZoningResidential_Medium_Density
##                             3902.977                               334.528
##                          Gr_Liv_Area                          Garage_Cars
##                                2.556                               613.732
```

The most critical number to pull out of this is the standard deviation of the effect of neighborhood, which is 2632. This tells us that there is 2632 square feet of standard deviation which is caused by the area that the property is in.

l. Construct 95% confidence intervals around each parameter estimate for mmod1. What does this tell us about the importance of the random effect? (2 points)

```
# get confidence intervals
confint(mmod1)
```

```
## Computing profile confidence intervals ...
```

```
##                                      2.5 %      97.5 %
## .sig01                           1973.672866 3497.769407
## .sigma                           2808.537490 2958.524273
## (Intercept)                      -694.527611 2177.928266
## MS_ZoningResidential_High_Density  531.528343 3540.134428
```

```
## MS_ZoningResidential_Low_Density    2922.737560 4882.578351
## MS_ZoningResidential_Medium_Density -784.245068 1451.999542
## Gr_Liv_Area                           2.280627    2.833066
## Garage_Cars                         420.776640  805.784529
```

These confidence intervals show that the random effect of neighborhood is an important predictor of lot area. The confidence interval for the random effect is smaller than for other predictors, and there is a 97.5% chance that the coefficient for the random effect is greater than 1970. We can be pretty sure that neighborhood is an important predictor of lot area.

m. Write out the full mathematical expression for the model in linmod2 and for the model in mmod1. You may round to the nearest integer in all coefficients. (4 points)

```
# get standard deviation of linmod2
sd(linmod2$residuals)
```

```
## [1] 3214.709
```

linmod2:

$$\begin{aligned}\text{Lot\_Area} \sim N(&1759+\\ &1356 \times \text{MS\_ZoningResidential\_High\_Density}+\\ &4141 \times \text{MS\_ZoningResidential\_Low\_Density}+\\ &725 \times \text{MS\_ZoningResidential\_Medium\_Density}+\\ &3 \times \text{Gr\_Liv\_Area}+\\ &297 \times \text{Garage\_Cars},\\ &3215)\end{aligned}$$

mmod1:

$$\begin{aligned}\text{Lot\_Area} \sim N(&743+\\ &2038 \times \text{MS\_ZoningResidential\_High\_Density}+\\ &3903 \times \text{MS\_ZoningResidential\_Low\_Density}+\\ &335 \times \text{MS\_ZoningResidential\_Medium\_Density}+\\ &3 \times \text{Gr\_Liv\_Area}+\\ &614 \times \text{Garage\_Cars}+\\ &N(0, 2632),\\ &2884)\end{aligned}$$

3. Logistic Regression

a. Construct a logistic regression model glmod1 for "played" as a function of player age and position. (2 points)

```r
# I used 'playerstats' variable here, as 'football' and 'footballngk' only contain
# players where 'played'==TRUE

# change 'position' to factor
playerstats$position <- as.factor(playerstats$position)

# construct model
poissonmod1 <- glm(played ~ age + position, family=binomial, data=playerstats)

# view model
poissonmod1
```

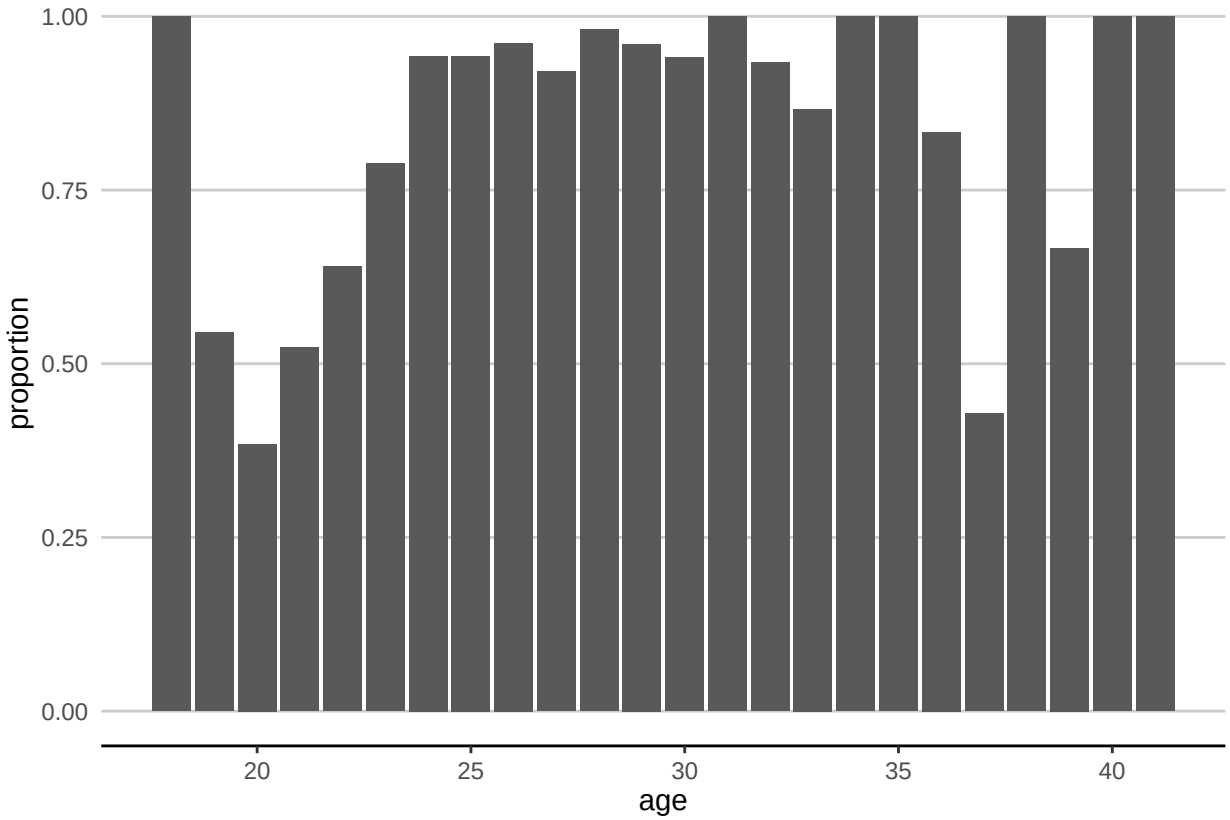
```
##
## Call:  glm(formula = played ~ age + position, family = binomial, data = playerstats)
##
## Coefficients:
##        (Intercept)                 age     positionForward  positionGoalkeeper
##            -5.2648              0.2750              0.8404             -2.2711
## positionMidfielder
##             0.6025
##
## Degrees of Freedom: 569 Total (i.e. Null);  565 Residual
## Null Deviance:       440.1
## Residual Deviance: 349.1     AIC: 359.1
```

It makes sense that goalkeepers here are less likely to have played than other positions. Goalkeepers are rotated a lot less often than other positions, and so it is unsurprising that a lot of goalkeepers didn't play at all. It is also unsurprising players were more likely to play if they were older: there were probably quite a few youth players who didn't play at all.

```r
# bar chart showing proportion of players who played for each age
p5 <- playerstats %>%
  group_by(age) %>%
  summarise(playedtrue = sum(played==TRUE),
            totalage=n(),
            proportion = playedtrue/totalage) %>%
  ggplot(
    mapping = aes(
      x = age,
      y = proportion
    )
  )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# show barchart
p5 + geom_bar(stat="identity")
```



This bar chart shows a rough trend, that players get more likely to play from the age of 20 to 24, and then this plateau's somewhat.

b. Construct confidence bands for the variable played as a function of age for each position (hint: create a new data frame for each position). Colour these with different tranparent colours for each position and plot them together on the same axes. Put the actual data on the plot, coloured to match the bands, and jittered in position to make it possible to see all points. Ensure you have an informative main plot title, axes labels and a legend. (6 points)

```
# make new dataframes for each position
goalkeepers <- playerstats %>%
  filter(position=='Goalkeeper')

defenders <- playerstats %>%
  filter(position=='Defender')

midfielders <- playerstats %>%
  filter(position=='Midfielder')

forwards <- playerstats %>%
  filter(position=='Forward')
```

```
# create new model for each position
pmod_g <- glm(played ~ age, family=binomial, data=goalkeepers)
pmod_g
```

```
##
```

```
## Call:  glm(formula = played ~ age, family = binomial, data = goalkeepers)
##
## Coefficients:
## (Intercept)          age
##    -1.42050      0.06734
##
## Degrees of Freedom: 56 Total (i.e. Null);  55 Residual
## Null Deviance:       73.87
## Residual Deviance: 72.35     AIC: 76.35
```

```
pmod_d <- glm(played ~ age, family=binomial, data=defenders)
pmod_d
```

```
##
## Call:  glm(formula = played ~ age, family = binomial, data = defenders)
##
## Coefficients:
## (Intercept)          age
##     -8.4343       0.4058
##
## Degrees of Freedom: 188 Total (i.e. Null);  187 Residual
## Null Deviance:       147.7
## Residual Deviance: 107.4     AIC: 111.4
```

```
pmod_m <- glm(played ~ age, family=binomial, data=midfielders)
pmod_m
```

```
##
## Call:  glm(formula = played ~ age, family = binomial, data = midfielders)
##
## Coefficients:
## (Intercept)          age
##     -8.4529       0.4406
##
## Degrees of Freedom: 210 Total (i.e. Null);  209 Residual
## Null Deviance:       136.7
## Residual Deviance: 97.91     AIC: 101.9
```

```
pmod_f <- glm(played ~ age, family=binomial, data=forwards)
pmod_f
```

```
##
## Call:  glm(formula = played ~ age, family = binomial, data = forwards)
##
## Coefficients:
## (Intercept)          age
##     -3.5193       0.2376
##
## Degrees of Freedom: 112 Total (i.e. Null);  111 Residual
## Null Deviance:       57.79
## Residual Deviance: 52.46     AIC: 56.46
```

```r
# get confidence intervals for each position in previous lm
confint(pmod_g)
```

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

##                   2.5 %    97.5 %
## (Intercept) -4.83741804 1.8636086
## age         -0.03923918 0.1809276
```

```r
confint(pmod_d)
```

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

##                   2.5 %     97.5 %
## (Intercept) -12.4817086 -4.9107520
## age           0.2618739  0.5765796
```

```r
confint(pmod_m)
```

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

##                   2.5 %     97.5 %
## (Intercept) -12.822878 -4.7336791
## age           0.279435  0.6379871
```

```r
confint(pmod_f)
```

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

##                   2.5 %    97.5 %
## (Intercept) -9.35290068 1.7237735
## age          0.03356854 0.4837371
```

```r
# make variable for prediction, CI, etc for goalkeepers
ilinkg <-family(pmod_g)$linkinv
newplayed_g <- with(goalkeepers,
                  data.frame(age=seq(min(goalkeepers$age),
                                     max(goalkeepers$age),
                                     length=length(
                                       unique(goalkeepers$age)))))
# get predictions for each age
newplayed_g <- cbind(newplayed_g,
                   predict(pmod_g,
                           newplayed_g,
                           type="link",
                           se.fit=TRUE)[1:2])
# get CI's for each age
newplayed_g <- transform(newplayed_g,
                         Fitted=ilinkg(fit),
                         Upper=ilinkg(fit+(1.96*se.fit)),
                         Lower=ilinkg(fit-(1.96*se.fit)))
```

```
# make variable for prediction, CI, etc for defenders
ilinkd <-family(pmod_d)$linkinv
newplayed_d <- with(defenders,
                  data.frame(age=seq(min(defenders$age),
                                     max(defenders$age),
                                     length=length(
                                       unique(defenders$age)))))
# get predictions for each age
newplayed_d <- cbind(newplayed_d,
                   predict(pmod_d,
                           newplayed_d,
                           type="link",
                           se.fit=TRUE)[1:2])
# get CI's for each age
newplayed_d <- transform(newplayed_d,
                         Fitted=ilinkd(fit),
                         Upper=ilinkd(fit+(1.96*se.fit)),
                         Lower=ilinkd(fit-(1.96*se.fit)))
```

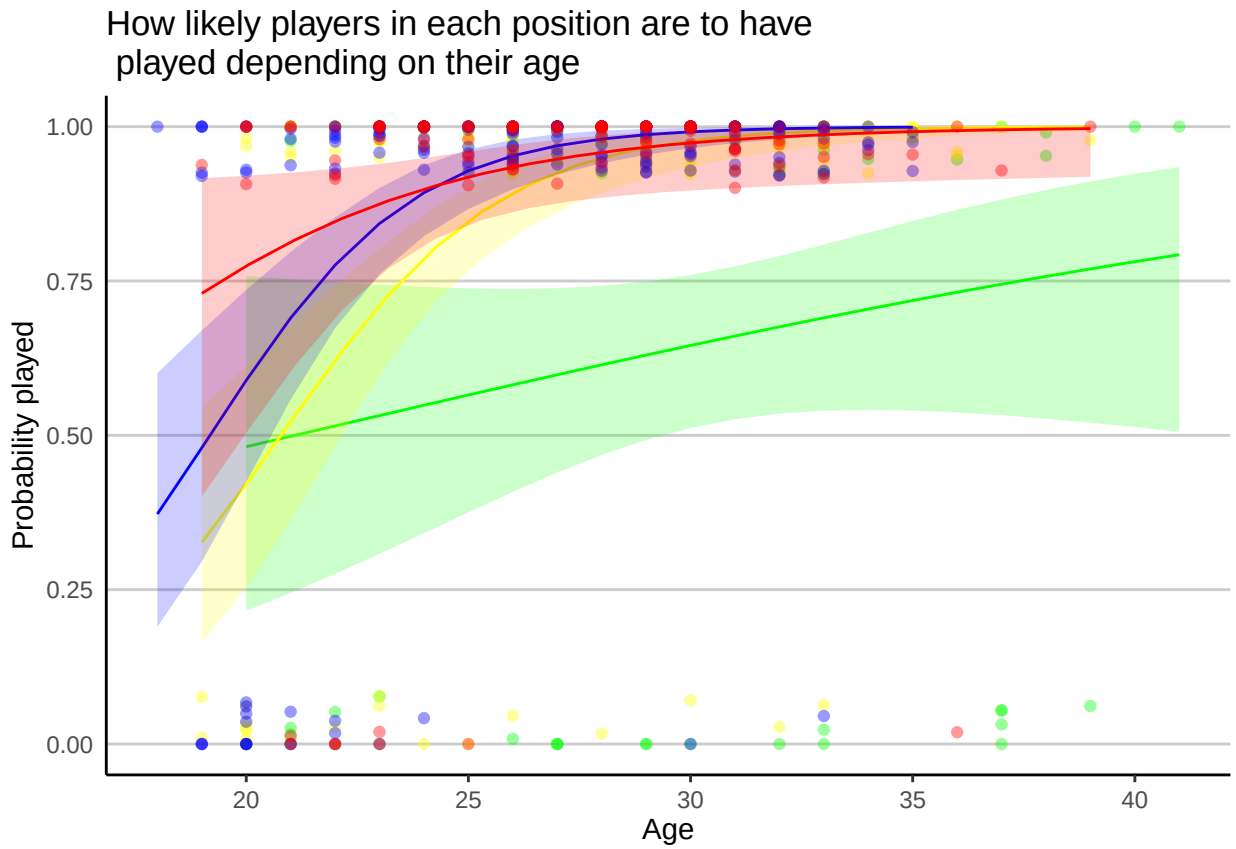
```
# make variable for prediction, CI, etc for midfielders
ilinkm <-family(pmod_m)$linkinv
newplayed_m <- with(midfielders,
                  data.frame(age=seq(min(midfielders$age),
                                     max(midfielders$age),
                                     length=length(
                                       unique(midfielders$age)))))
# get predictions for each age
newplayed_m <- cbind(newplayed_m,
                   predict(pmod_m,
                           newplayed_m,
                           type="link",
                           se.fit=TRUE)[1:2])
# get CI's for each age
newplayed_m <- transform(newplayed_m,
                         Fitted=ilinkm(fit),
                         Upper=ilinkm(fit+(1.96*se.fit)),
                         Lower=ilinkm(fit-(1.96*se.fit)))
```

```
# make variable for prediction, CI, etc for forwards
ilinkf <-family(pmod_f)$linkinv
newplayed_f <- with(forwards,
                  data.frame(age=seq(min(forwards$age),
                                     max(forwards$age),
                                     length=length(
                                       unique(forwards$age)))))
# get predictions for each age
newplayed_f <- cbind(newplayed_f,
                   predict(pmod_f,
                           newplayed_f,
                           type="link",
                           se.fit=TRUE)[1:2])
# get CI's for each age
newplayed_f <- transform(newplayed_f,
```

```
                        Fitted=ilinkf(fit),
                        Upper=ilinkf(fit+(1.96*se.fit)),
                        Lower=ilinkf(fit-(1.96*se.fit)))
```

```
# make plot object
p6 <-ggplot(playerstats,
       aes(x=age,
           y=as.numeric(played))) +
  # add labesl and title
  labs(title = "How likely players in each position are to have \n played depending on their age",
       y = "Probability played",
       x = "Age",
       colour='Legend') +
  # improve aesthetics
  theme(axis.line.y  = element_line(colour = "black")) +
  # scale to make predictions after jitter stay between 0 and 1
  scale_y_continuous(limits = c(0, 1), oob = scales::squish)

p6 +

  # add goalkeeper prediction line
  geom_line(data = newplayed_g,
            aes(y = Fitted,
                x = age),
            colour="green") +
  # add CI's
  geom_ribbon(data = newplayed_g,
              aes(ymin = Lower,
                  ymax = Upper,
                  x = age),
              fill = "green",
              alpha = 0.2,
              inherit.aes = FALSE) +
  # add prediction for each age
  geom_point(data = goalkeepers,
             position = position_jitter(w=0, h=0.08),
             colour="green",
             alpha=0.4) +



  # add defender prediction line
  geom_line(data = newplayed_d,
            aes(y = Fitted,
                x = age),
            colour="yellow") +
  # add CI's
  geom_ribbon(data = newplayed_d,
              aes(ymin = Lower,
                  ymax = Upper,
                  x = age),
              fill = "yellow",
              alpha = 0.2,
```

```
            inherit.aes = FALSE) +
# add prediction for each age
geom_point(data = defenders,
           position = position_jitter(w=0, h=0.08),
           colour="yellow",
           alpha=0.4) +



# add midfielder prediction line
geom_line(data = newplayed_m,
          aes(y = Fitted,
              x = age),
          colour="blue") +
# add CI's
geom_ribbon(data = newplayed_m,
            aes(ymin = Lower,
                ymax = Upper,
                x = age),
            fill = "blue",
            alpha = 0.2,
            inherit.aes = FALSE) +
# add prediction for each age
geom_point(data = midfielders,
           position = position_jitter(w=0, h=0.08),
           colour="blue",
           alpha=0.4) +



# add forward prediction line
geom_line(data = newplayed_f,
          aes(y = Fitted,
              x = age),
          colour="red") +
# add CI's
geom_ribbon(data = newplayed_f,
            aes(ymin = Lower,
                ymax = Upper,
                x = age),
            fill = "red",
            alpha = 0.2,
            inherit.aes = FALSE) +
# add prediction for each age
geom_point(data = forwards,
           position = position_jitter(w=0, h=0.1),
           colour="red",
           alpha=0.4) +

  labs(title = "How likely players in each position are to have \n played depending on their age",
     y = "Probability played",
     x = "Age",
     colour='Legend')
```



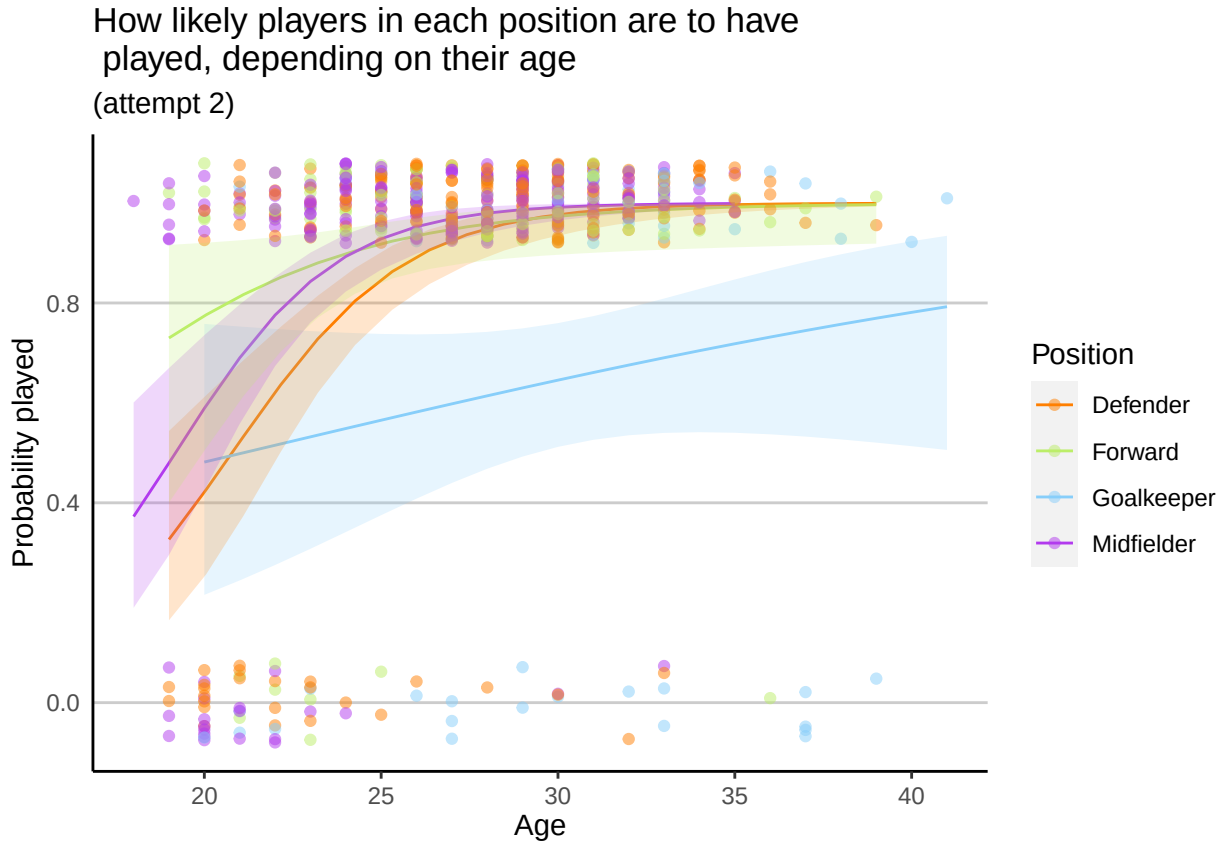
I'm not sure how to add a legend to the graph above. I looked online, and it seems as though I'll have to make a dataframe containing all the players to be able to do this.

```
# so that i can add a legend to the plot, i will make a new dataframe containing
# the predictions and CI's for each position

# first, create a new dataframe for each position
newdf_g <- data.frame(newplayed_g)
newdf_g$position <- "Goalkeeper"

newdf_d <- data.frame(newplayed_d)
newdf_d$position <- "Defender"

newdf_m <- data.frame(newplayed_m)
newdf_m$position <- "Midfielder"

newdf_f <- data.frame(newplayed_f)
newdf_f$position <- "Forward"

# concatenate these dtaframes
newdf <- rbind(newdf_g, newdf_d, newdf_m, newdf_f)
```

```
# now, do the same as before but with newdf

p7 <-ggplot(playerstats,
       aes(x=age,
```

```
          y=as.numeric(played),
          colour=position)) +
  # add labesl and title
  labs(title = "How likely players in each position are to have \n played, depending on their age",
       subtitle = "(attempt 2)",
       y = "Probability played",
       x = "Age",
       colour='Position') +
  # improve aesthetics
  theme(axis.line.y  = element_line(colour = "black"))
  # # scale to make predictions after jitter stay between 0 and 1
  # scale_y_continuous(limits = c(0, 1), oob = scales::squish)

p7 +
  #customize scale colours
  scale_color_manual(values=c("darkorange1",
                              "darkolivegreen2",
                              "lightskyblue",
                              "darkorchid2")) +

  # add prediction lines
  geom_line(data = newdf,
            aes(y = Fitted,
                x = age)) +

  # add prediction for each age
  geom_point(data = playerstats,
             position = position_jitter(w=0, h=0.08),
             alpha=0.5) +


# add CI's for goalkeepers
  geom_ribbon(data = newdf[newdf$position=="Goalkeeper",],
              aes(ymin = Lower,
                  ymax = Upper,
                  x = age),
              fill = "lightskyblue",
              alpha = 0.2,
              inherit.aes = FALSE) +

# add CI's for defenders
  geom_ribbon(data = newdf[newdf$position=="Defender",],
              aes(ymin = Lower,
                  ymax = Upper,
                  x = age),
              fill = "darkorange1",
              alpha = 0.2,
              inherit.aes = FALSE) +


# add CI's for midfielders
  geom_ribbon(data = newdf[newdf$position=="Midfielder",],
              aes(ymin = Lower,
```

```
                    ymax = Upper,
                    x = age),
                fill = "darkorchid2",
                alpha = 0.2,
                inherit.aes = FALSE) +


# add CI's for forwards
  geom_ribbon(data = newdf[newdf$position=="Forward",],
                aes(ymin = Lower,
                    ymax = Upper,
                    x = age),
                fill = "darkolivegreen2",
                alpha = 0.2,
                inherit.aes = FALSE)
```



c. Split the data using set.seed(123) and rebuild the model on 70% of the data. Cross validate on the remaining 30%. Plot the ROCs for both data and comment on your findings. (6 points)

```
# set seed
set.seed(123)

# split data 70:30
training_samples <- playerstats$played %>%
  createDataPartition(p = 0.7, list = FALSE)
train_data <- playerstats[training_samples, ]
```

```
test_data <- playerstats[-training_samples, ]
```

```
# train model
train_model <- glm(played ~ age + position,
                   family = binomial,
                   data = train_data)
```

```
# make predictions with training data
predtrain<-predict(train_model,type="response")

# make predictions with test data
predtest<-predict(train_model,newdata=test_data,type="response")

# show ROC curve for training data
roctrain<-roc(response=train_data$played,
              predictor=predtrain,
              plot=TRUE,
              main="ROC Curve for prediction of whether a player has played",
              auc=TRUE)
```

```
## Setting levels: control = FALSE, case = TRUE
```

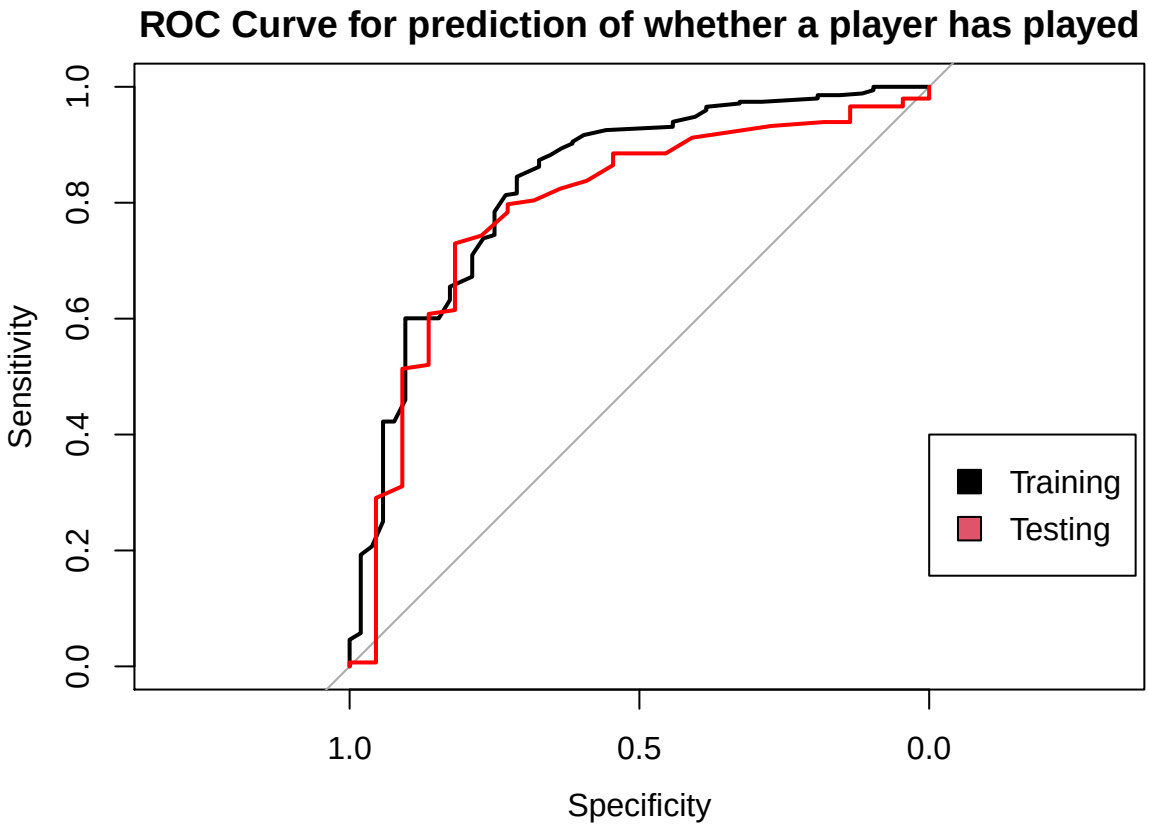
```
## Setting direction: controls < cases
```

```
# adfd ROC curve for testing data
roc(response=test_data$played,
    predictor=predtest,
    plot=TRUE,
    auc=TRUE,
    add=TRUE,
    col="red")
```

```
## Setting levels: control = FALSE, case = TRUE
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = test_data$played, predictor = predtest,     auc = TRUE, plot = TRUE, add = TRU
##
## Data: predtest in 22 controls (test_data$played FALSE) < 148 cases (test_data$played TRUE).
## Area under the curve: 0.7905
```

```
legend(0, 0.4,legend=c("Training","Testing"),fill=1:2)
```



The ROC curves look pretty similar in shape, though there are some small gaps. The model doesn't appear to be massively overfitted to the training data.

4. Multinomial Regression

a. For the dataset footballngk, create a model multregmod to predict position from goals_per_90_overall, assists_per_90_overall, conceded_per_90_overall and cards_per_90_overall. (2 points)

```
# create model
multregmod <- multinom(position ~ goals_per_90_overall +
                         assists_per_90_overall +
                         conceded_per_90_overall +
                         cards_per_90_overall,
                       data = footballngk)
```

```
## # weights:  18 (10 variable)
## initial  value 504.263040
## iter  10 value 429.868747
## iter  20 value 425.352173
## iter  30 value 425.289072
## final  value 425.289026
## converged
```

```r
# view model
multregmod
```

```
## Call:
## multinom(formula = position ~ goals_per_90_overall + assists_per_90_overall +
##     conceded_per_90_overall + cards_per_90_overall, data = footballngk)
##
## Coefficients:
##            (Intercept) goals_per_90_overall assists_per_90_overall
## Forward     -2.4030684             9.609048               3.888129
## Midfielder  -0.4545415             4.816176               2.752608
##            conceded_per_90_overall cards_per_90_overall
## Forward                  0.3324864         -0.030419009
## Midfielder               0.0472488          0.009268531
##
## Residual Deviance: 850.5781
## AIC: 870.5781
```

b. Write out the formulas for this model in terms of P(Forward) and P(Midfielder). You may round coefficients to 2 digits. All other factors equal, what position is a player with more assists more likely to play? (5 points)

$$\begin{aligned}\text{P(Forward)} \sim N(&-2.4+\\ &9.6 \times \text{goals\_per\_90\_overall}+\\ &3.9 \times \text{assists\_per\_90\_overall}+\\ &0.33 \times \text{conceded\_per\_90\_overall}-\\ &0.030 \times \text{cards\_per\_90\_overall}\end{aligned}$$

$$\begin{aligned}\text{P(Midfielder)} \sim N(&-0.45+\\ &4.8 \times \text{goals\_per\_90\_overall}+\\ &2.8 \times \text{assists\_per\_90\_overall}+\\ &0.047 \times \text{conceded\_per\_90\_overall}+\\ &0.0093 \times \text{cards\_per\_90\_overall}\end{aligned}$$

All other factors equal, a player with more assists is more likely to play as a forward

c. Evaluate the performance of this model using a confusion matrix and by calculating the sum of sensitivities for the model. Comment on your findings. (3 points)

```r
# make confusion matrix
multitable <- table(footballngk$position,
                    predict(multregmod,
                            type="class"))
# add column names for clarity
names(dimnames(multitable)) <- list("Actual",
                                    "Predicted")
# view confusion matrix
multitable
```

```
##             Predicted
## Actual       Defender Forward Midfielder
##   Defender        109       2         53
##   Forward          21      51         33
##   Midfielder       79      20         91
```

```
# calculate sum of sensitivities
SSens <- multitable[1,1]/sum(footballngk$position=="Defender") +
  multitable[2,2]/sum(footballngk$position=="Forward") +
  multitable[3,3]/sum(footballngk$position=="Midfielder")
SSens
```

```
## [1] 1.629296
```

```
# get average sum of sensitivities per position
mean_SSens <- SSens/3

mean_SSens
```

```
## [1] 0.5430986
```

The model predicts about 54% of players' positions correctly. The model seems to predict a lot of defenders as midfielders. This could be because some positions in defense are allowed more freedom to go forward, and so may get similar stats to some midfielders. The opposite is also true, and this is likely to be because some midfielders are more defensive and will therefore have similar stats to some defenders.

Similar is true of forwards: some forwards are involved more in the build-up play and so may get more similar stats to midfielders or even defenders, causing them to be classified incorrectly.

Also, the categories of defender, midfielder, and forward are pretty broad, and with more precise position labels such as striker, winger, or full-back we may get more accurate predictions of position.

5. Poisson/quasipoisson Regression

a. For the football dataset, first create a variable indicating the total number of all cards a player received overall. Then create a model countmod to predict the total number of cards a player received based on position and appearances. (3 points)

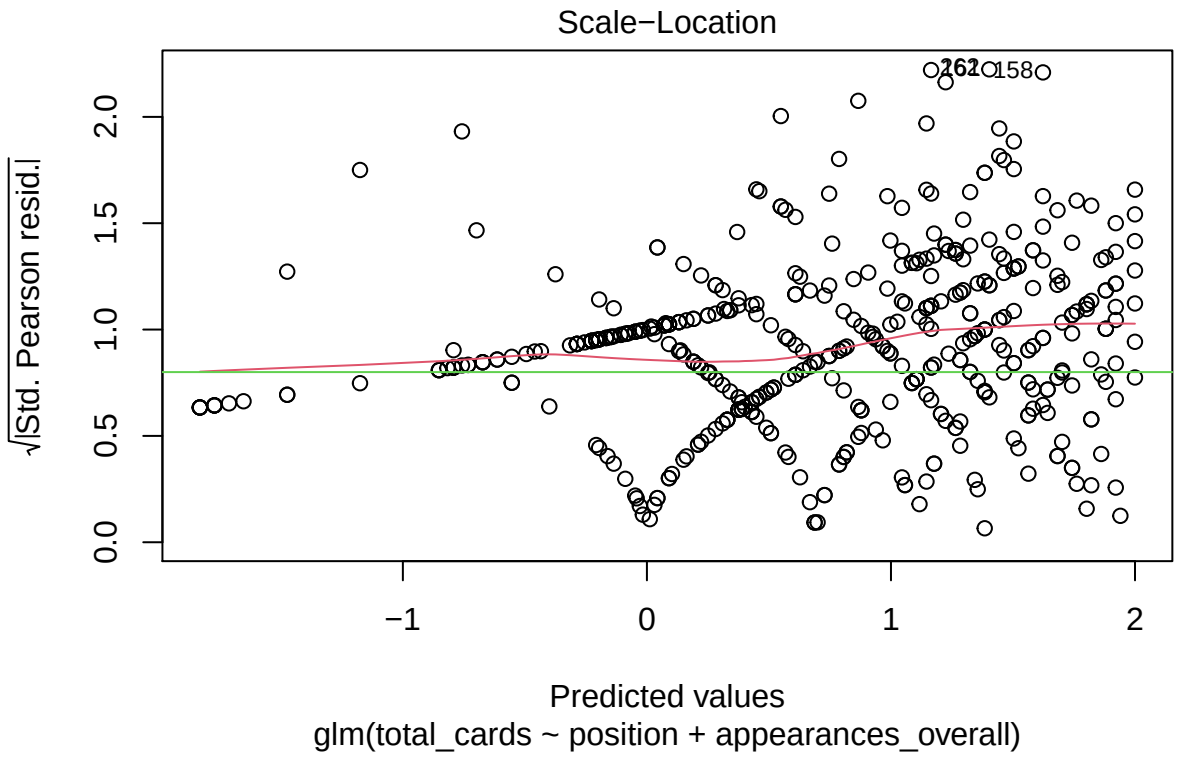
```
# make total_cards variable
football$total_cards <- football$yellow_cards_overall +
  football$red_cards_overall

# change position to factor
football$position <- as.factor(football$position)
```

```
# generate model
countmod <- glm(total_cards ~ position +
                  appearances_overall,
                data = football,
                family = "poisson")

# show model
summary(countmod)
```

```
##
## Call:
## glm(formula = total_cards ~ position + appearances_overall, family = "poisson",
##     data = football)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8438  -1.2518  -0.4009   0.5860   3.8636
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.267619   0.091564  -2.923  0.00347 **
## positionForward     -0.644150   0.086043  -7.486 7.08e-14 ***
## positionGoalkeeper  -1.623990   0.197731  -8.213  < 2e-16 ***
## positionMidfielder  -0.078637   0.059862  -1.314  0.18897
## appearances_overall  0.059669   0.002989  19.965  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1468.53  on 495  degrees of freedom
## Residual deviance:  848.46  on 491  degrees of freedom
## AIC: 1940.2
##
## Number of Fisher Scoring iterations: 5
```

b. Check the assumption of the model using a diagnostic plot and comment on your findings. (2 points)

```r
# to evaluate dispersion assumption, plot abs value of residuals vs predicted means
plot(countmod,which=3)
abline(h=0.8,col=3)
```



The mean dispersion is not completely flat, but is pretty close to 0.8. The plot does show slight overdispersion, which increases as the prediction increases. It would probably be better to use a quasipoisson model which does not assume dispersion to be equal to the mean.

```
# try building a quasipoisson model
quasicountmod <- glm(total_cards ~ position +
                       appearances_overall,
                     data = football,
                     family = "quasipoisson")

# show model
summary(quasicountmod)
```

```
##
## Call:
## glm(formula = total_cards ~ position + appearances_overall, family = "quasipoisson",
##     data = football)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8438  -1.2518  -0.4009   0.5860   3.8636
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.267619   0.118376  -2.261   0.0242 *
## positionForward     -0.644150   0.111237  -5.791 1.25e-08 ***
```

```
## positionGoalkeeper  -1.623990   0.255630  -6.353 4.82e-10 ***
## positionMidfielder  -0.078637   0.077391  -1.016   0.3101
## appearances_overall  0.059669   0.003864  15.443  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.671378)
##
##     Null deviance: 1468.53  on 495  degrees of freedom
## Residual deviance:  848.46  on 491  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

In the quasipoisson model, the dispersion parameter is estimated at 1.67 instead of 1, confirming overdispersion in the previous model.

c. What do the coefficients of the model tell us about which position gets the most cards? For the same minutes of play, how many times more cards do they get than the position that gets the least cards? (3 points)

As defender was the baseline factor coefficient in the model, and all other position coefficients were negative, the coefficients show that **defenders tend to get the most cards**.

For the second part of the question, we should build a new model using minutes played to predict total cards received, as the previous model used appearances rather than minutes played.

```
# build new model with minutes played instead of appearances
countmod2 <- glm(total_cards ~ position +
                   minutes_played_overall,
                 data = football,
                 family = "quasipoisson")

# show model
countmod2
```

```
##
## Call:  glm(formula = total_cards ~ position + minutes_played_overall,
##     family = "quasipoisson", data = football)
##
## Coefficients:
##             (Intercept)         positionForward      positionGoalkeeper
##               -0.086344               -0.336091                -1.713140
##     positionMidfielder  minutes_played_overall
##                0.110223                0.000631
##
## Degrees of Freedom: 495 Total (i.e. Null);  491 Residual
## Null Deviance:       1469
## Residual Deviance: 821.3     AIC: NA
```

```
# make predictions to compare midfielders (most cards per minute) to goalkeepers (least cards per minut

# goalkeeper
gk_cards <- predict(countmod2,
        newdata = data.frame(position="Goalkeeper",
```

```
                                        minutes_played_overall=1000),
        type = "response")

# midfielder
mf_cards <- predict(countmod2,
        newdata = data.frame(position="Midfielder",
                                        minutes_played_overall=1000),
        type = "response")

mf_cards / gk_cards
```

```
##        1
## 6.192646
```

For the same number of minutes played, midfielders get **6.2** times as many cards as goalkeepers do.