Employee Attrition

Jon Gomez (jag2j), Michael Langmayr, Nathan England, and Yihnew Eshetu

About the data

We will analyze the "IBM HR Analytics Employee Attrition & Performance" dataset ((link)[https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset]). The dataset documentation says that it was synthesized by data scientists at IBM. Observations describe hypothetical employees. Rows measure the employee retention status, metrics about the workplace, and details about the employee.

Preparation

Libraries

Loading the data

We load the data using a custom column mapping. A few considerations apply:

- Several columns are originally numeric levels. We recode these from the data description.
- A few columns are irrelevant or constant across the data. We drop these.

General observations

There are 1,470 rows and 31 columns. The features include both factors and quantitative variables. We show a sample observation below.

```
Age                      "41"              MonthlyIncome            "5993"
Attrition                "Yes"             MonthlyRate              "19479"
BusinessTravel           "Travel_Rarely"   NumCompaniesWorked       "8"
DailyRate                "1102"            OverTime                 "Yes"
Department               "Sales"           PercentSalaryHike        "11"
DistanceFromHome         "1"               PerformanceRating        "Excellent"
Education                "College"         RelationshipSatisfaction "Low"
EducationField           "Life Sciences"   StockOptionLevel         "0"
EnvironmentSatisfaction  "Medium"          TotalWorkingYears        "8"
Gender                   "Female"          TrainingTimesLastYear    "0"
HourlyRate               "94"              WorkLifeBalance          "Bad"
JobInvolvement           "High"            YearsAtCompany           "6"
JobLevel                 "2"               YearsInCurrentRole       "4"
JobRole                  "Sales Executive" YearsSinceLastPromotion  "0"
JobSatisfaction          "Very High"       YearsWithCurrManager     "5"
MaritalStatus            "Single"
```

Correlation

We look at columns for which the correlation is greater than 70 in the defactored data:

```
## cor( JobLevel ,  MonthlyIncome ) =  76
## cor( MonthlyIncome ,  TotalWorkingYears ) =  77
## cor( PercentSalaryHike ,  PerformanceRating ) =  77
## cor( YearsAtCompany ,  YearsInCurrentRole ) =  76
## cor( YearsAtCompany ,  YearsWithCurrManager ) =  77
## cor( YearsInCurrentRole ,  YearsWithCurrManager ) =  71
```

Predicting monthly income

Selecting initial models

We use the leaps library to evaluate possible variables to include. Given the fairly small data set, we perform an exhaustive search using `regsubsets`. This provides the following results:

```
## Maximized: r2 with value 0.947986365994346
##    Formula: (response) ~  1 + JobLevel1 + JobLevel3 + JobLevel4 + JobLevel5 + JobRoleResearch Scienti
##
## Maximized: adjr2 with value 0.9476657340039
##    Formula: (response) ~  1 + JobLevel1 + JobLevel3 + JobLevel4 + JobLevel5 + JobRoleResearch Scienti
```

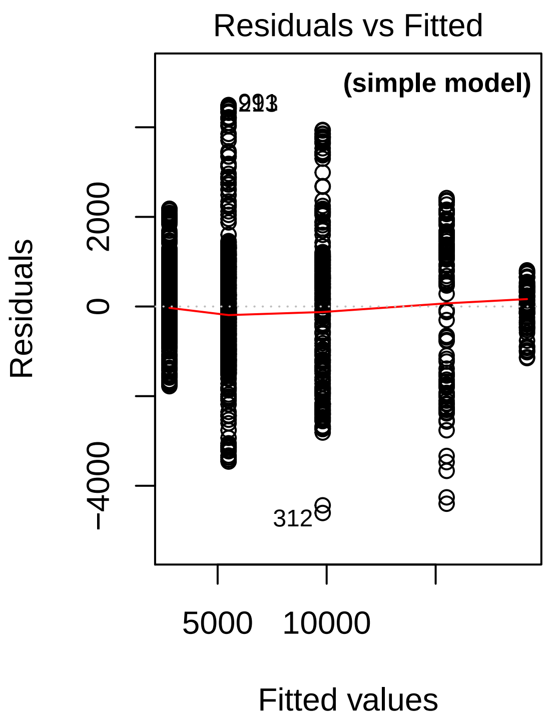
Since the model with the extreme value for every criterion is the same in the exhautive search, we fit the specified model. We also fit the simplified model with only JobLevel, since it has such a high correlation to monthly income (as seen in the "Preparation" section).

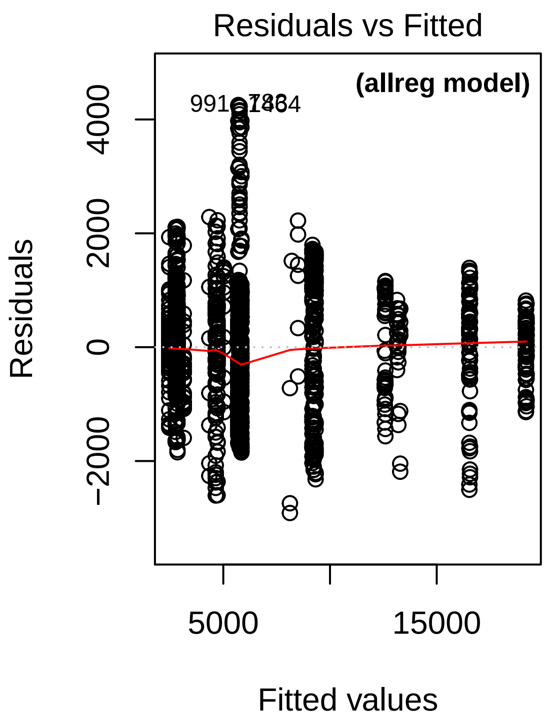
```
##
## Call:
## lm(formula = MonthlyIncome ~ 1 + JobLevel + JobRole, data = ibm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2912.3  -662.0   -87.3   651.8  4253.4
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      5762.72      64.63  89.161  < 2e-16 ***
## JobLevel1                       -1878.46     102.01 -18.415  < 2e-16 ***
## JobLevel3                        3474.85      92.98  37.370  < 2e-16 ***
## JobLevel4                        7439.74     154.88  48.037  < 2e-16 ***
## JobLevel5                       10089.59     202.41  49.848  < 2e-16 ***
## JobRoleResearch Scientist       -1029.30     120.58  -8.536  < 2e-16 ***
## JobRoleLaboratory Technician    -1115.24     120.58  -9.249  < 2e-16 ***
## JobRoleManufacturing Director     -59.07     107.22  -0.551    0.582
## JobRoleHealthcare Representative   87.79     111.27   0.789    0.430
## JobRoleManager                   3328.58     177.16  18.789  < 2e-16 ***
## JobRoleSales Representative     -1416.68     162.62  -8.712  < 2e-16 ***
## JobRoleResearch Director         3357.60     167.92  19.995  < 2e-16 ***
## JobRoleHuman Resources           -735.81     172.87  -4.256 2.21e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1071 on 1457 degrees of freedom
## Multiple R-squared:  0.9487, Adjusted R-squared:  0.9483
## F-statistic:  2246 on 12 and 1457 DF,  p-value: < 2.2e-16
```

```
##
## Call:
## lm(formula = MonthlyIncome ~ JobLevel, data = ibm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4607.3  -713.2   -93.9   694.1  4495.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5502.28      55.80   98.61   <2e-16 ***
## JobLevel1   -2715.36      78.58  -34.56   <2e-16 ***
## JobLevel3    4314.98     103.63   41.64   <2e-16 ***
## JobLevel4   10001.51     137.10   72.95   <2e-16 ***
## JobLevel5   13689.55     164.94   83.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1289 on 1465 degrees of freedom
## Multiple R-squared:  0.9252, Adjusted R-squared:  0.925
## F-statistic:  4530 on 4 and 1465 DF,  p-value: < 2.2e-16
```

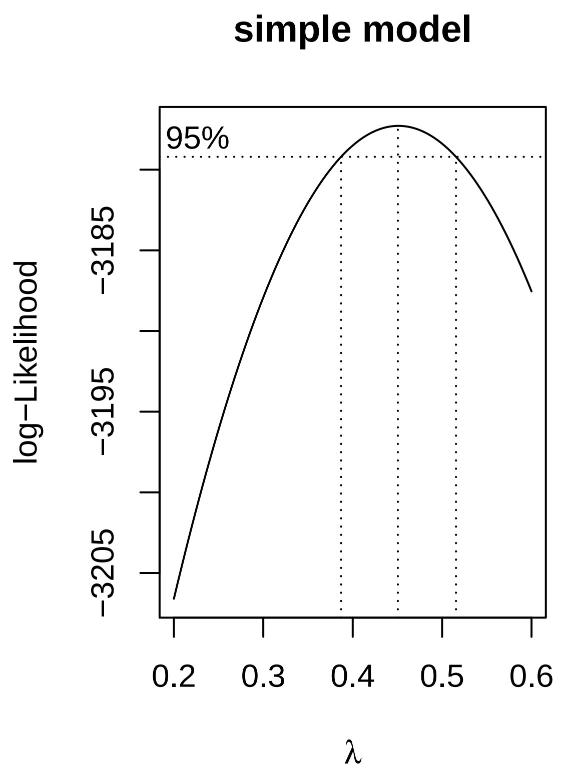
Adequacy and Box-Cox

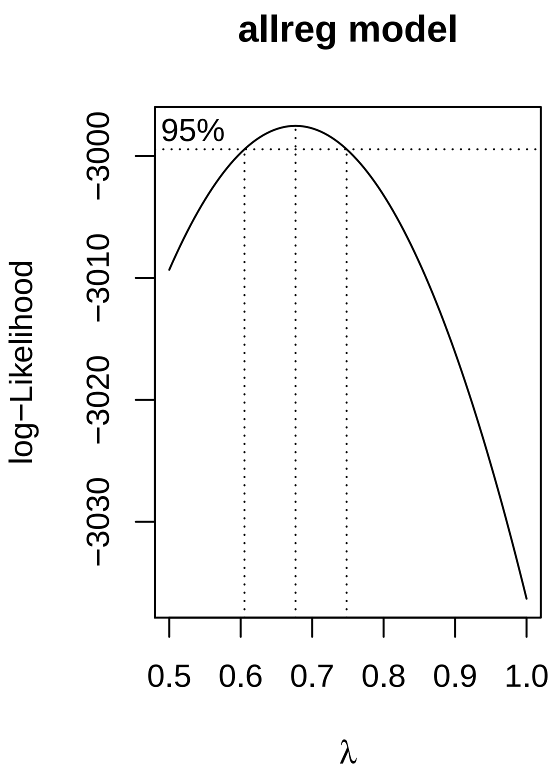
When plotting the residuals, we find that both models have non-constant variance. Further, they exhibit strong clumping due to the use of factorized levels as predictors:





We first try to fix these problems with Box-Cox. We pick transformations using $\lambda = 0.5$ (sqrt) and $\lambda = 0.7$ for the simple and allreg models respectively.

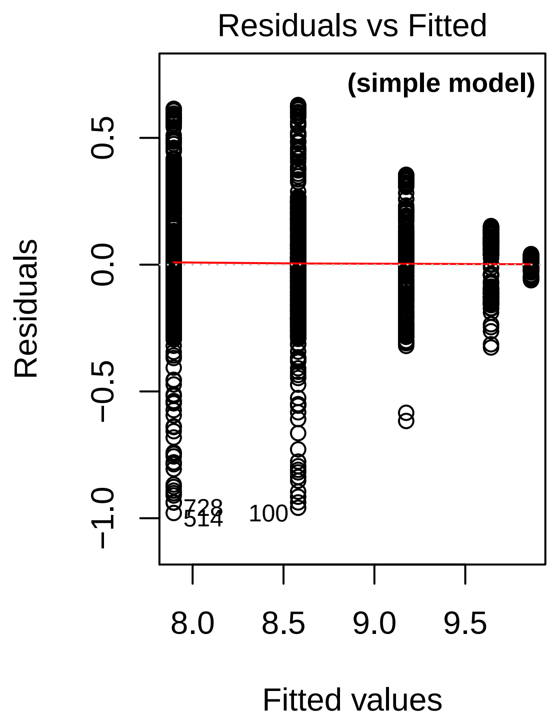


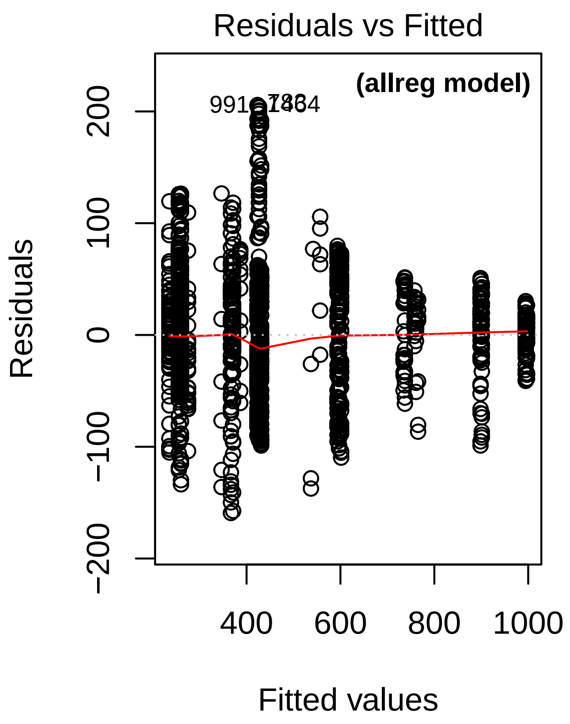


A second round of Box-Cox results in confidence intervals containing 1 for both models. We also see that the p-value for two indicator variables have high p-values in the allreg model, but the other indicator variables remain viable.

```
##
## Simple model ===> predictors with p-value > 0.5

##  None!

## Allreg model ===> predictors with p-value > 0.5

##                                    Estimate Std. Error    t value  Pr(>|t|)
## JobRoleManufacturing Director    -3.300322   5.619101 -0.5873399 0.5570664
## JobRoleHealthcare Representative  4.492074   5.831826  0.7702688 0.4412653
```

More problematically, the constant variance assumption still appears not to be met:



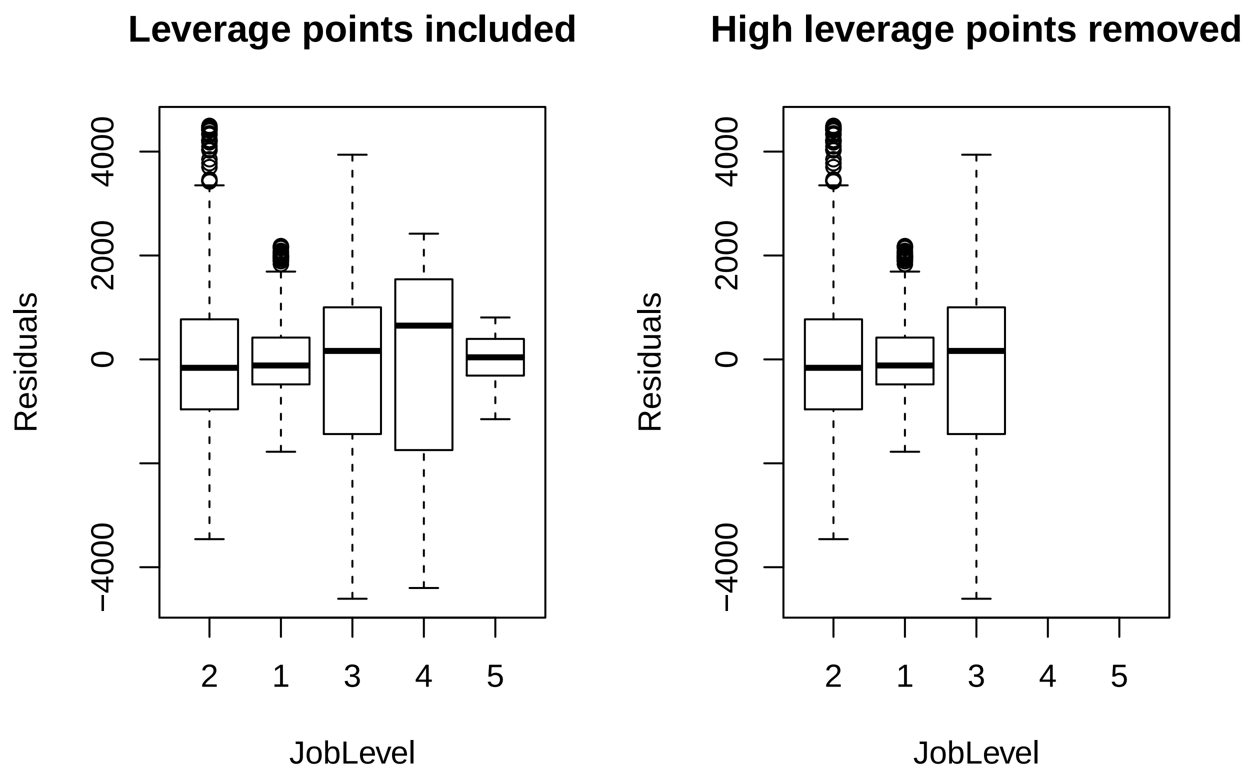


Leverage points

We now turn to leverage points, and we come to an important realization. When we remove them from the simple model, we drop entire job levels:

```
##                  # of leverage points # of Cook's distance relevant points
##                                   175                                    0
```

```
model = model.simple
{par(mfrow=c(1,2))
plot(model$residuals ~ ibm$JobLevel, main = "Leverage points included", xlab="JobLevel", ylab="Residuals
plot(model$residuals[-which(lev>2*p/n)] ~ ibm$JobLevel[-which(lev>2*p/n)], main = "High leverage points
```



At this point, we decided that it made sense to consider individual models for each individual job level since they have such different characteristics. In retrospect, this is another way to interpret the box-and-whisker plots.

Individual models

```r
# Create subsets for each job level
jobLevel1 = subset(ibm, JobLevel == 1)
jobLevel2 = subset(ibm, JobLevel == 2)
jobLevel3 = subset(ibm, JobLevel == 3)
jobLevel4 = subset(ibm, JobLevel == 4)
jobLevel5 = subset(ibm, JobLevel == 5)
```

```r
# using exhaustive search for each job level
# reload existing results
load(file="models by job level - images and object dumps/joblevel_data.R", verbose = TRUE)
```

```
## Loading objects:
##   allreg.jobLevel1
##   allreg.jobLevel2
##   allreg.jobLevel3
##   allreg.jobLevel4
##   allreg.jobLevel5
```

```r
# run again
#allreg.jobLevel1 = regsubsets(MonthlyIncome ~., data=jobLevel1, nbest=1, really.big = T)
#allreg.jobLevel2 = regsubsets(MonthlyIncome ~., data=jobLevel2, nbest=1, really.big = T)
#allreg.jobLevel3 = regsubsets(MonthlyIncome ~., data=jobLevel3, nbest=1, really.big = T)
#allreg.jobLevel4 = regsubsets(MonthlyIncome ~., data=jobLevel4, nbest=1, really.big = T)
#allreg.jobLevel5 = regsubsets(MonthlyIncome ~., data=jobLevel5, nbest=1, really.big = T)
```

```r
# fit regression model for best r2 model for job level 1
cat("=== joblevel1\n")
```

```
## === joblevel1
```

```r
pp_allreg(allreg.jobLevel1)
```

```
## Maximized: r2 with value 0.132988503672628
##    Formula: (response) ~  1 + AttritionYes + JobRoleSales Representative + NumCompaniesWorked + Perc
##
## Maximized: adjr2 with value 0.11834853469149
##    Formula: (response) ~  1 + AttritionYes + JobRoleSales Representative + NumCompaniesWorked + Perc
```

```r
cat("=== joblevel2\n")
```

```
## === joblevel2
```

```r
pp_allreg(allreg.jobLevel2)
```

```
## Maximized: r2 with value 0.150548565355962
##    Formula: (response) ~  1 + BusinessTravelTravel_Rarely + DailyRate + EducationCollege + Education
##
## Maximized: adjr2 with value 0.135958750638793
##    Formula: (response) ~  1 + BusinessTravelTravel_Rarely + DailyRate + EducationCollege + Education
```

```r
cat("=== joblevel3\n")
```

```
## === joblevel3
```

```r
pp_allreg(allreg.jobLevel3)
```

```
## Maximized: r2 with value 0.695564396196394
##    Formula: (response) ~  1 + BusinessTravelTravel_Frequently + EducationMaster + JobRoleLaboratory
##
## Maximized: adjr2 with value 0.682391701801045
##    Formula: (response) ~  1 + BusinessTravelTravel_Frequently + EducationMaster + JobRoleLaboratory
```

```r
cat("=== joblevel4\n")
```

```
## === joblevel4
```

```r
pp_allreg(allreg.jobLevel4)
```

```
## Maximized: r2 with value 0.817682594713165
##    Formula: (response) ~  1 + EducationDoctor + EnvironmentSatisfactionVery High + JobRoleManager + 
##
## Maximized: adjr2 with value 0.800590337967524
##    Formula: (response) ~  1 + EducationDoctor + EnvironmentSatisfactionVery High + JobRoleManager + 
```

```r
cat("=== joblevel5\n")
```

```
## === joblevel5
```

```r
pp_allreg(allreg.jobLevel5)
```

```
## Maximized: r2 with value 0.368029993171166
##    Formula: (response) ~  1 + BusinessTravelTravel_Frequently + DailyRate + EducationCollege + Educa
##
## Maximized: adjr2 with value 0.271627788739649
##    Formula: (response) ~  1 + BusinessTravelTravel_Frequently + DailyRate + EducationCollege + Educa
```

We fit two of the better models:

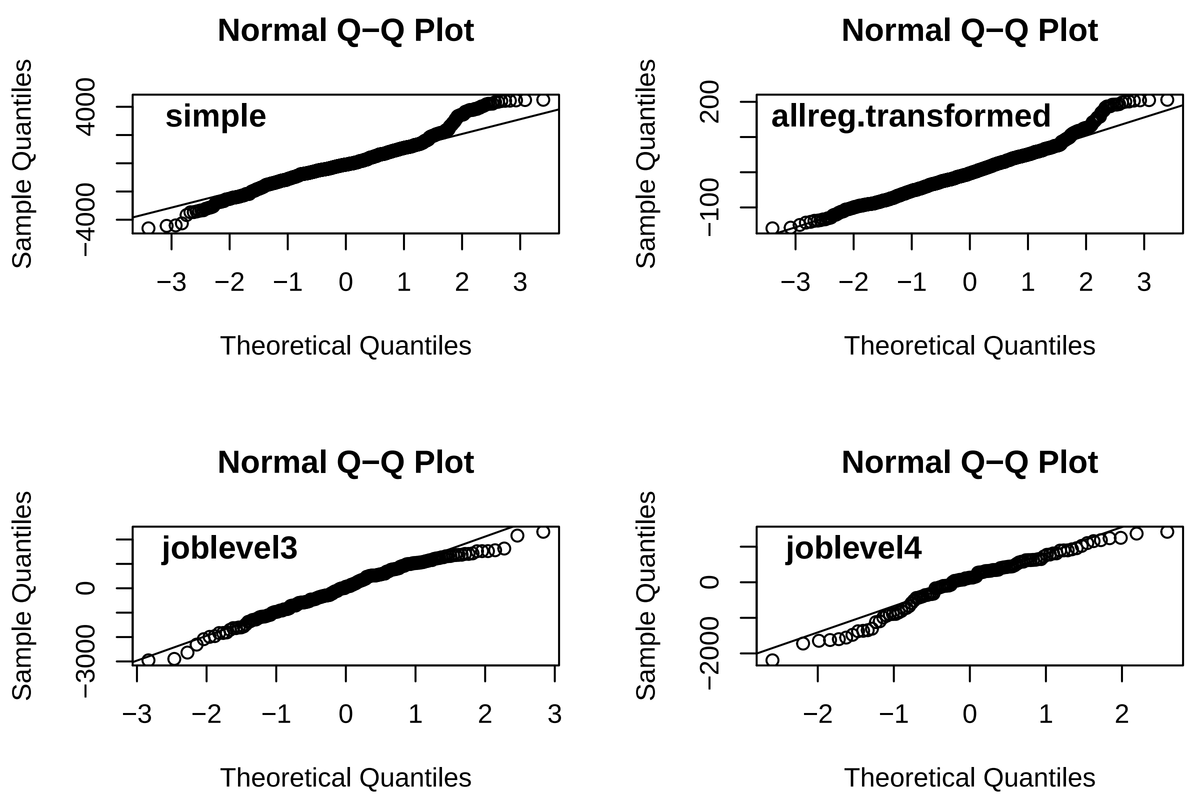
```r
model.joblevel3 = lm(MonthlyIncome ~ 1 + BusinessTravel + Education + JobRole + NumCompaniesWorked + Rel
model.joblevel4 = lm(MonthlyIncome ~ 1 + Education  + EnvironmentSatisfaction + JobRole + MaritalStatus
```

Model adequacy, continued

At this point, we want to assess final adquacy.

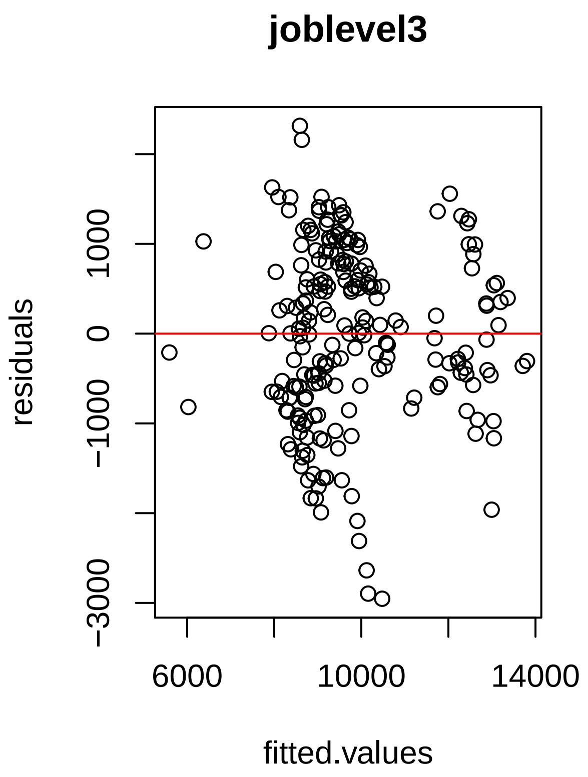
QQ plot.

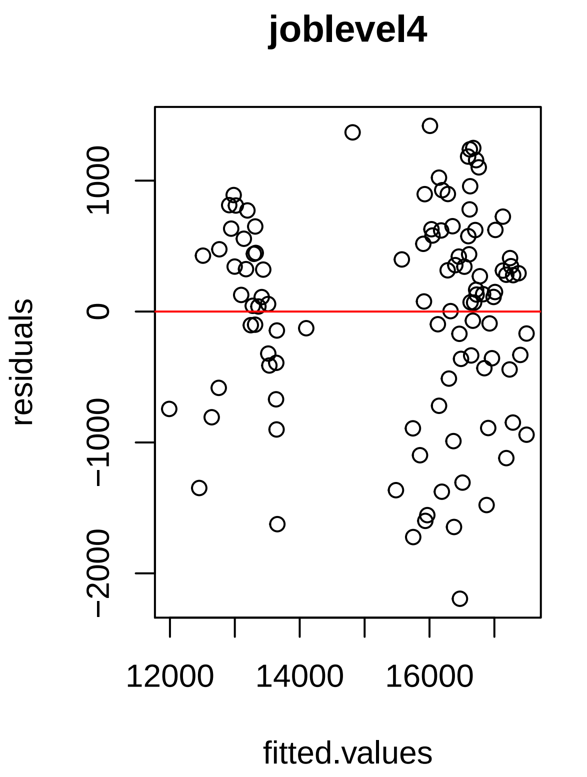
```r
{
  par(mfrow=c(2,2))
  qqnorm(model.simple$residuals); title("simple", adj=0.1, line=-1)
  qqline(model.simple$residuals)

  qqnorm(model.allreg.transform$residuals); title("allreg.transformed", adj=0.1, line=-1)
  qqline(model.allreg.transform$residuals)

  qqnorm(model.joblevel3$residuals); title("joblevel3", adj=0.1, line=-1)
  qqline(model.joblevel3$residuals)

  qqnorm(model.joblevel4$residuals); title("joblevel4", adj=0.1, line=-1)
  qqline(model.joblevel4$residuals)
}
```



Residual plots for the job level models. These look good.

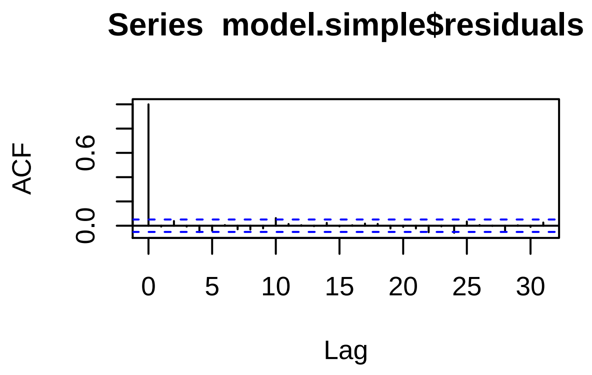
```
{
  par(mfrow=c(1,2))
  with(model.joblevel3, plot(residuals ~ fitted.values, main="joblevel3"))
  abline(h = 0, col="Red")
  with(model.joblevel4, plot(residuals ~ fitted.values, main="joblevel4"))
  abline(h = 0, col="Red")
}
```

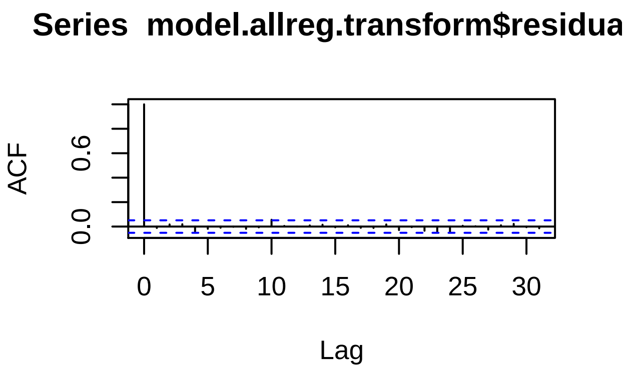


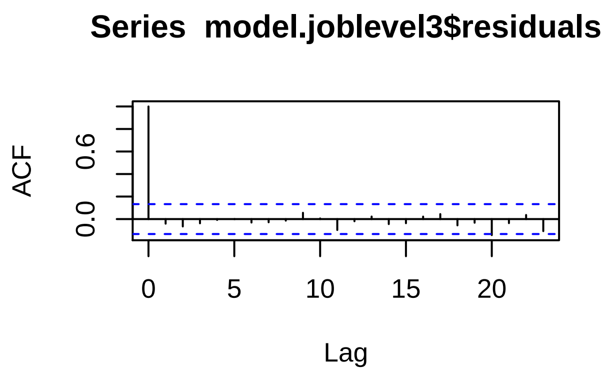


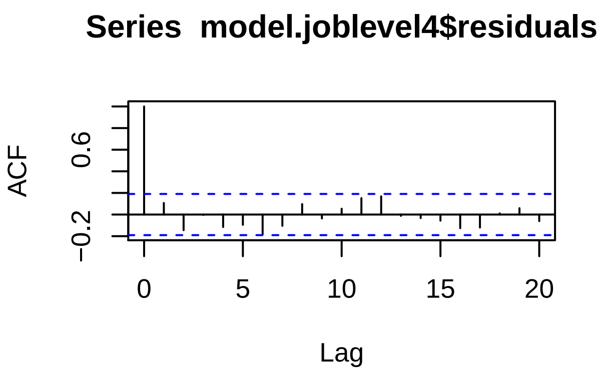
ACFs look fine.

```
{
  par(mfrow=c(2,2))
  acf(model.simple$residuals)
  acf(model.allreg.transform$residuals)
  acf(model.joblevel3$residuals)
  acf(model.joblevel4$residuals)
}
```









Predictive performance

We look at one or two models briefly.

Final recommendations

We consider the following models:

name	formula
model.simple	lm(MonthlyIncome ~ JobLevel, data = ibm)
model.allreg.transformed	lm(MonthlyIncome^0.7 ~ JobLevel + JobRole, data = ibm)
model.joblevel3	lm(MonthlyIncome ~ BusinessTravel + Education + JobRole + NumCompaniesWorked + RelationshipSatisfaction + StockOptionLevel + TotalWorkingYears)
model.joblevel4	lm(MonthlyIncome ~ Education + EnvironmentSatisfaction + JobRole + MaritalStatus + MonthlyRate + PerformanceRating + RelationshipSatisfaction)

For the simplest model, we recommend model.simple (not the transformed one), which fits to each job level an average value. While quick and highly predictive, with an R squared of around 0.93, the model does not meet the constant variance assumption needed for effective use of hypothesis tests. Furthermore, it is also so simplistic that it groups employees into a few preset groups. For slightly the best overall fit for the general employee, we recommend using the model.allreg.transformed variant.

However, the different levels of jobs appear to be rather distinctive based on the box-and-whiskers graph of the residuals against the job levels in these models. For this reason, we investigated looking at models that specialize in particular job levels. Two of these, model.joblevel3 and model.joblevel4 had decent values of R squared, respectively 0.70 and 0.82. These also deal with a broader range of predictors and distinguish a larger variety of employee. The residual plots and qq plots look much better for these models than for the general models.

These models offer some approaches to predicting employee monthly income. The best method will ultimately depend on the specific goals informing the analysis.

Extra: Stepwise selection

In this section, we consider some alternative models generated from stepwise selection models. While stepwise selection may report a model with inferior values for our criteria compared to the exhaustive search, the reported model may still provide a better model in terms of linearity assumptions. For this reason, we do consider a stepwise selection.

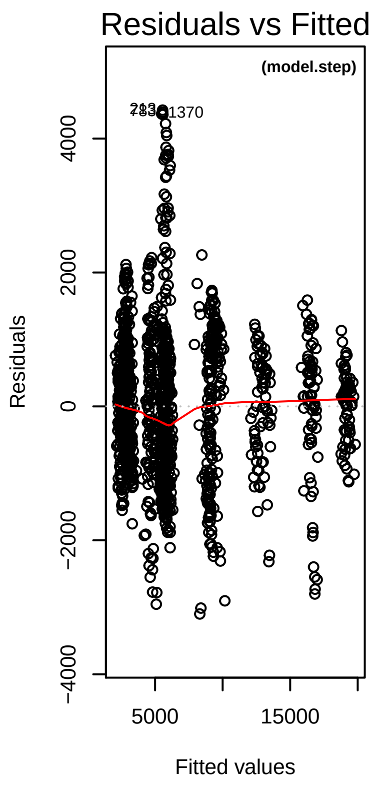
Initial fit and transform

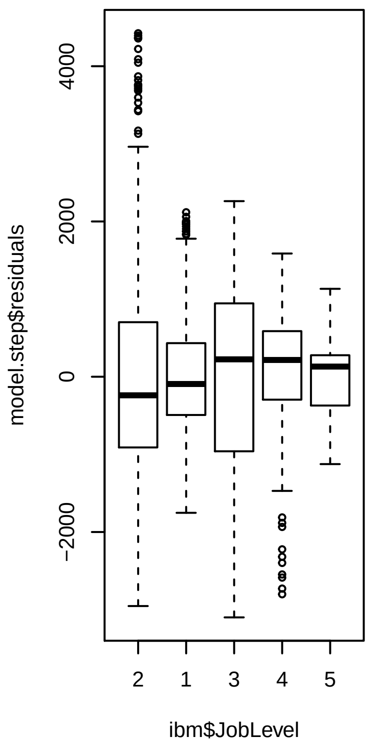
We use the `step` function with `both` directions.

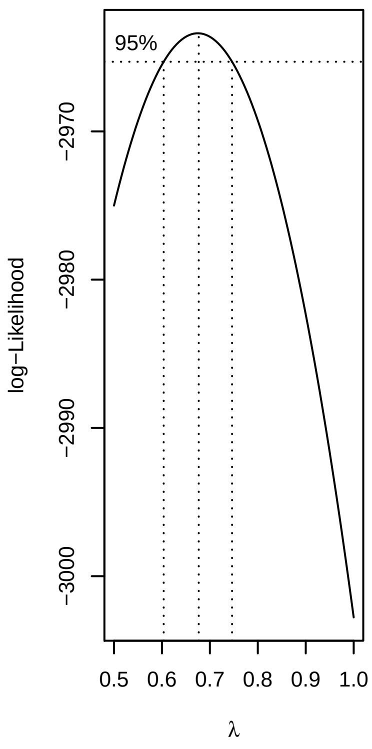
```
## lm(formula = MonthlyIncome ~ Gender + JobInvolvement + JobLevel +
##     JobRole + NumCompaniesWorked + StockOptionLevel + TotalWorkingYears +
##     YearsInCurrentRole, data = ibm)
```

The diagnostic plots are not that much better.

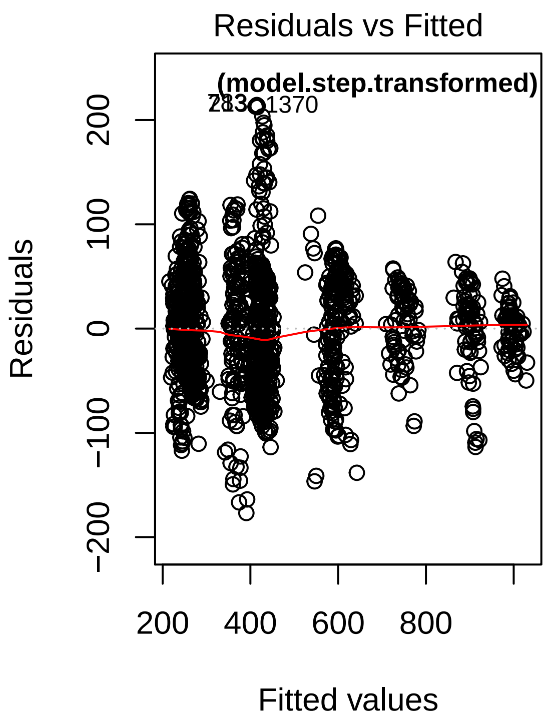
```r
{par(mfrow=c(1,3))
plot(model.step, which = 1); title("(model.step)", line=-1, adj=0.95, cex.main=0.8)
plot(model.step$residuals ~ ibm$JobLevel)
boxcox(model.step, lambda = seq(0.5,1, by=0.01))}
```

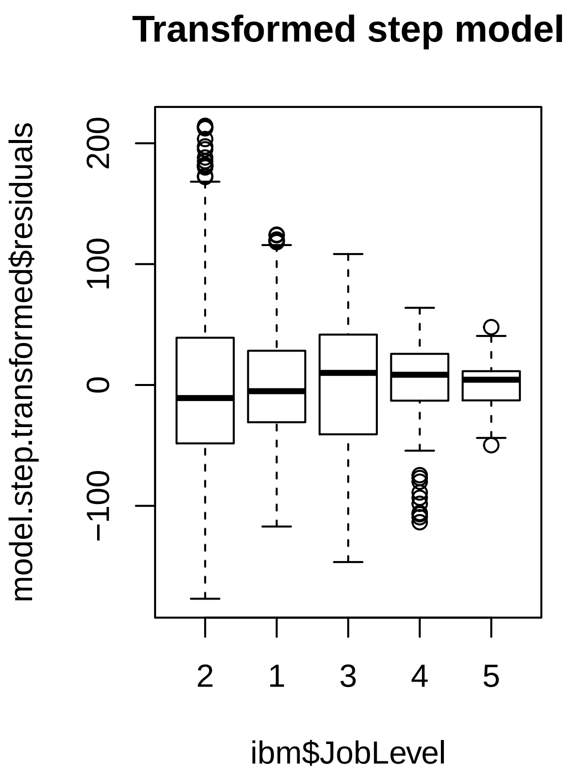






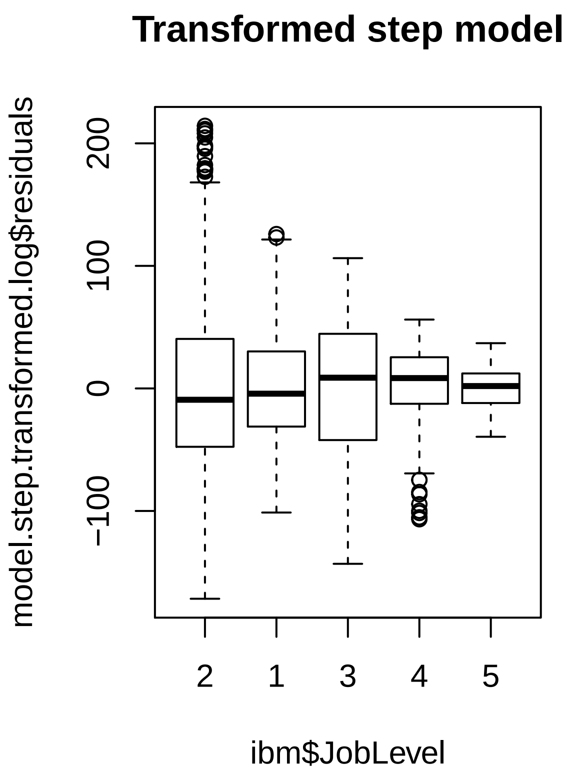
With the Box-Cox transform, we get the following:

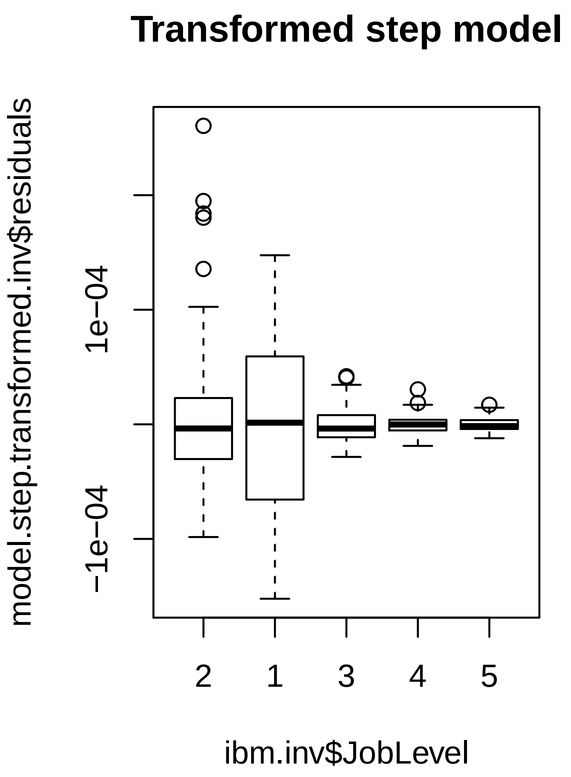




Additional remediation

We continue our efforts and perform some transforms to the (quantitative) predictors. These are motivated by a desire to transform the shape of the residuals. In particular, we use the log and multiplicative inverse transforms on the predictors. Unfortunately, this does not really help.


Transformed step model
model.step.transformed.log$residuals
200
100
0
−100
2
1
3
4
5
ibm$JobLevel


Transformed step model
model.step.transformed.inv$residuals
1e−04
−1e−04
2
1
3
4
5
ibm.inv$JobLevel