Analysis of Vehicular Insurance Losses and Actuarial Symboling

Glib Dolotov, Yunhan Liao, Sarthi Shah, Qianyan Wu

12/02/2021

1. Introduction

The import, sale, and insurance of automobiles has been a core part of the American economy for decades. Of particular interest to insurance companies are the expenses incured by vehicle owners as a result of mechanical breakdowns, theft, and collision damage. In order to properly price insurance policies, car insurers try to predict these losses based on a variety of other factors. To this end, many vehicle statistics are collected. One such compendium of statistics is the Ward's Automotive Yearbook. From the UCI Machine Learning Repository, we selected a selection of the 1985 Ward's Automotive Yearbook for our analyses. http://archive.ics.uci.edu/ml/datasets/Automobile

The data consists of various attributes of vehicles including weight and dimensions, engine properties, etc. Of particular interest are the symboling and normalized losses variables. The symbol represents a vehicle's "riskiness". It is an ordinal categorical variable for the purposes of our analysis. Normalized losses is the "relative average loss payment per insured vehicle year." We examine how these two factors affect each other and how other factors affect them in this project.

2. Analysis

2.2 Handling Systematic Tied Values in the Data

A visual examination of the data reveals many ties within the symboling and normalized losses variables. For example, examining rows 174 to 178 reveals the same symbling and normalized loss were assigned to five vehicles considered distinct within the data set. However examination of their characteristics suggests that they are different variants of the same model, with symbol and losses being assigned on a per-model basis. The five Toyota vehicles in rows 174 to 178 have identical dimensions, wheel bases, and (mostly) the same engines. Lines 154 and 155, however, reveal two near identical vehicles with different normalized losses, suggesting that the drive wheels (fwd and 4wd for the two vehicles) are considered different models with regards to the symboling and normalized losses.

```
# Multiple differences, same symboling and normalized losses
data[174:178,]
```

```
##     symboling normalized.losses   make fuel.type aspiration num.of.doors
## 174        -1                65 toyota       gas        std         four
## 175        -1                65 toyota    diesel      turbo         four
## 176        -1                65 toyota       gas        std         four
## 177        -1                65 toyota       gas        std         four
## 178        -1                65 toyota       gas        std         four
##     body.style drive.wheels engine.location wheel.base length width height
## 174      sedan          fwd           front      102.4  175.6  66.5   54.9
## 175      sedan          fwd           front      102.4  175.6  66.5   54.9
## 176  hatchback          fwd           front      102.4  175.6  66.5   53.9
## 177      sedan          fwd           front      102.4  175.6  66.5   54.9
## 178  hatchback          fwd           front      102.4  175.6  66.5   53.9
```

```
##     curb.weight engine.type num.of.cylinders engine.size fuel.system bore
## 174        2326         ohc             four         122        mpfi 3.31
## 175        2480         ohc             four         110         idi 3.27
## 176        2414         ohc             four         122        mpfi 3.31
## 177        2414         ohc             four         122        mpfi 3.31
## 178        2458         ohc             four         122        mpfi 3.31
##     stroke compression.ratio horsepower peak.rpm city.mpg highway.mpg price
## 174   3.54               8.7         92     4200       29          34  8948
## 175   3.35              22.5         73     4500       30          33 10698
## 176   3.54               8.7         92     4200       27          32  9988
## 177   3.54               8.7         92     4200       27          32 10898
## 178   3.54               8.7         92     4200       27          32 11248
```

```
# Only difference in below two is drive.wheels
data[154:155,]
```

```
##     symboling normalized.losses   make fuel.type aspiration num.of.doors
## 154         0                77 toyota       gas        std         four
## 155         0                81 toyota       gas        std         four
##     body.style drive.wheels engine.location wheel.base length width height
## 154      wagon          fwd           front       95.7  169.7  63.6   59.1
## 155      wagon          4wd           front       95.7  169.7  63.6   59.1
##     curb.weight engine.type num.of.cylinders engine.size fuel.system bore
## 154        2280         ohc             four          92        2bbl 3.05
## 155        2290         ohc             four          92        2bbl 3.05
##     stroke compression.ratio horsepower peak.rpm city.mpg highway.mpg price
## 154   3.03                9         62     4800       31          37  6918
## 155   3.03                9         62     4800       27          32  7898
```

Without examining every possible tie individually, we can determine if the per-model assignment holds across the entire data by fitting a multiple linear regression. Consider a statistical model where each configuration of traits that determine a vehicle model correspond to distinct normalized loss statistics. If this model over-fits, we can conclude that normalized loss and symboling are assigned per vehicle model throughout the entire dataset. As it turns out, the model `normalized.losses ~ make : drive.wheels : wheel.base` fits incredibly well. The output below is abridged. Full output is in **Appendix 5.4** which will be split off into a separate file (it is quite long).

```
# Does identifying the model predict ties in symboling and normalized losses?
# See Appendix 5.4 for full printout
print.sum2(summary(lm(normalized.losses ~ make : drive.wheels : wheel.base, data = data)))
```

```
##
## Call:
## lm(formula = normalized.losses ~ make:drive.wheels:wheel.base,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.024 -12.130   0.000   9.411 128.860
##
## Residual standard error: 19.8 on 139 degrees of freedom
##   (41 observations deleted due to missingness)
## Multiple R-squared:  0.7338, Adjusted R-squared:  0.6878
## F-statistic: 15.96 on 24 and 139 DF,  p-value: < 2.2e-16
```

A consequence of the many ties and the same vehicle model being repeated throughout the data is that when we consider wheel-base a categorical variable, the model fits even better.

```
print.sum2(summary(lm(normalized.losses ~ make : drive.wheels : as.factor(wheel.base), data = data)))
```

```
##
## Call:
## lm(formula = normalized.losses ~ make:drive.wheels:as.factor(wheel.base),
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.0000  -0.1136   0.0000   2.0000  23.0000
##
## Residual standard error: 10.74 on 111 degrees of freedom
##   (41 observations deleted due to missingness)
## Multiple R-squared:  0.9375, Adjusted R-squared:  0.9083
## F-statistic: 32.03 on 52 and 111 DF,  p-value: < 2.2e-16
```

```
# Interestingly, considering wheel.base as categorical improves the model (curious aside, nothing else)
# AIC is lower, F-statistic is higher, R^2 is higher
AIC(lm(normalized.losses ~ make : drive.wheels : wheel.base, data = data))
```

```
## [1] 1469.643
```

```
AIC(lm(normalized.losses ~ make : drive.wheels : as.factor(wheel.base), data = data))
```

```
## [1] 1287.917
```

These repeated instances of a single normalized loss and symboling assignment means that any analysis of these two variables will be vulnerable to very strong bias. For example, a model with eight variants all sharing the same symbol and loss will have more weight in any analysis than a model which only has two variants. For this reason, we will "collapse" or "deflate" the data by removing redundant rows. The `collapse_model` function source code can be found in **Appendix 5.5**. Unless a variable we are analyzing differs between two variants of the same model, the rows representing the extra variants are removed from the pool.

```
# See Appendix 5.5 for collapse_model function code
deflated_data <- collapse_model(data)
head(data)
```

```
##   symboling normalized.losses        make fuel.type aspiration num.of.doors
## 1         3                NA alfa-romero       gas        std          two
## 2         3                NA alfa-romero       gas        std          two
## 3         1                NA alfa-romero       gas        std          two
## 4         2               164        audi       gas        std         four
## 5         2               164        audi       gas        std         four
## 6         2                NA        audi       gas        std          two
##    body.style drive.wheels engine.location wheel.base length width height
## 1 convertible          rwd           front       88.6  168.8  64.1   48.8
## 2 convertible          rwd           front       88.6  168.8  64.1   48.8
## 3   hatchback          rwd           front       94.5  171.2  65.5   52.4
## 4       sedan          fwd           front       99.8  176.6  66.2   54.3
## 5       sedan          4wd           front       99.4  176.6  66.4   54.3
## 6       sedan          fwd           front       99.8  177.3  66.3   53.1
##   curb.weight engine.type num.of.cylinders engine.size fuel.system bore stroke
## 1        2548        dohc             four         130        mpfi 3.47   2.68
## 2        2548        dohc             four         130        mpfi 3.47   2.68
## 3        2823        ohcv              six         152        mpfi 2.68   3.47
## 4        2337         ohc             four         109        mpfi 3.19   3.40
## 5        2824         ohc             five         136        mpfi 3.19   3.40
```

```
## 6        2507         ohc             five         136        mpfi 3.19   3.40
##   compression.ratio horsepower peak.rpm city.mpg highway.mpg price
## 1               9.0        111     5000       21          27 13495
## 2               9.0        111     5000       21          27 16500
## 3               9.0        154     5000       19          26 16500
## 4              10.0        102     5500       24          30 13950
## 5               8.0        115     5500       18          22 17450
## 6               8.5        110     5500       19          25 15250
```

```
head(deflated_data)
```

```
##   symboling normalized.losses        make fuel.type aspiration num.of.doors
## 1         3                NA alfa-romero       gas        std          two
## 3         1                NA alfa-romero       gas        std          two
## 4         2               164        audi       gas        std         four
## 6         2                NA        audi       gas        std          two
## 7         1               158        audi       gas        std         four
## 8         1                NA        audi       gas        std         four
##    body.style drive.wheels engine.location wheel.base length width height
## 1 convertible          rwd           front       88.6  168.8  64.1   48.8
## 3   hatchback          rwd           front       94.5  171.2  65.5   52.4
## 4       sedan          fwd           front       99.8  176.6  66.2   54.3
## 6       sedan          fwd           front       99.8  177.3  66.3   53.1
## 7       sedan          fwd           front      105.8  192.7  71.4   55.7
## 8       wagon          fwd           front      105.8  192.7  71.4   55.7
##   curb.weight engine.type num.of.cylinders engine.size fuel.system bore stroke
## 1        2548        dohc             four         130        mpfi 3.47   2.68
## 3        2823        ohcv              six         152        mpfi 2.68   3.47
## 4        2337         ohc             four         109        mpfi 3.19   3.40
## 6        2507         ohc             five         136        mpfi 3.19   3.40
## 7        2844         ohc             five         136        mpfi 3.19   3.40
## 8        2954         ohc             five         136        mpfi 3.19   3.40
##   compression.ratio horsepower peak.rpm city.mpg highway.mpg price
## 1               9.0        111     5000       21          27 13495
## 3               9.0        154     5000       19          26 16500
## 4              10.0        102     5500       24          30 13950
## 6               8.5        110     5500       19          25 15250
## 7               8.5        110     5500       19          25 17710
## 8               8.5        110     5500       19          25 18920
```

The above output demonstrates the difference between the original `data` and the collapsed `deflated_data` as rows 2 and 5 were removed. The summary tables are available in **Appendix 5.6**.

2.3 Comparison of Population Means

2.3.0 Variable Selection

To perform a comparison of two population means, several assumptions must be evaluated. The sampling distribution must be approximately normal (this is achievable by a sufficiently large sample size, or by normality testing). Furthermore, since we do not know the variances of our variables a priori, we must evaluate whether or not to pool the variances or consider them unequal. For our first attempt, we tried to test whether or not the population means for normalized losses differed between gas and diesel vehicles. However, as shown in the below output, we did not have enough data normally distributed to meet the necessary testing assumptions.

```r
# Part 1 Analysis -- Comparing population means
diesel <- data[data$fuel.type == "diesel",]
gas <- data[data$fuel.type == "gas",]
c(nrow(gas), nrow(diesel))
```

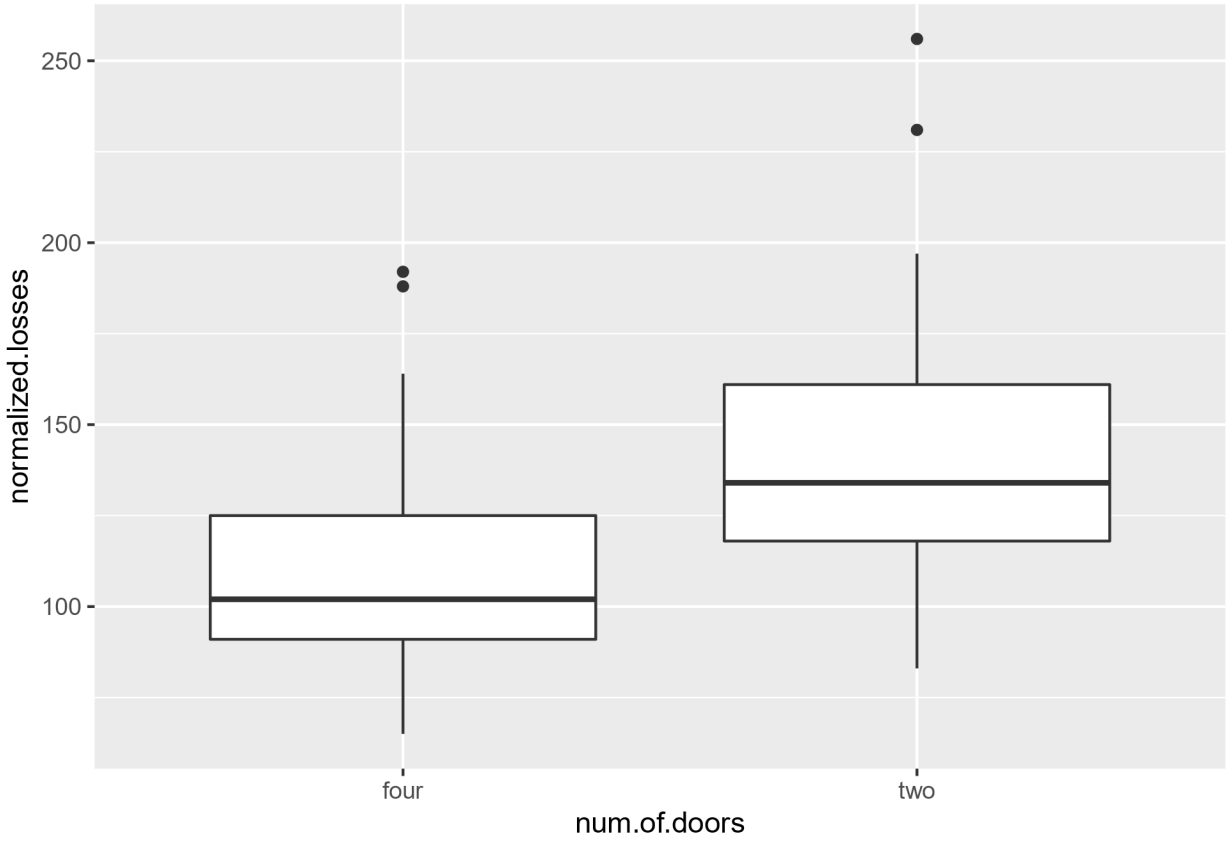
```
## [1] 185  20
```

```r
# Testing normality for diesel (gas has over 100 observations)
shapiro.test(diesel$normalized.losses)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  diesel$normalized.losses
## W = 0.78716, p-value = 0.002528
```

```r
# Reject normality assumption
```

Next, we considered comparing population means for two-door and four-door vehicles. A visualization via box-plots suggests these populations may have differing means and may have closer-to-normal distributions.

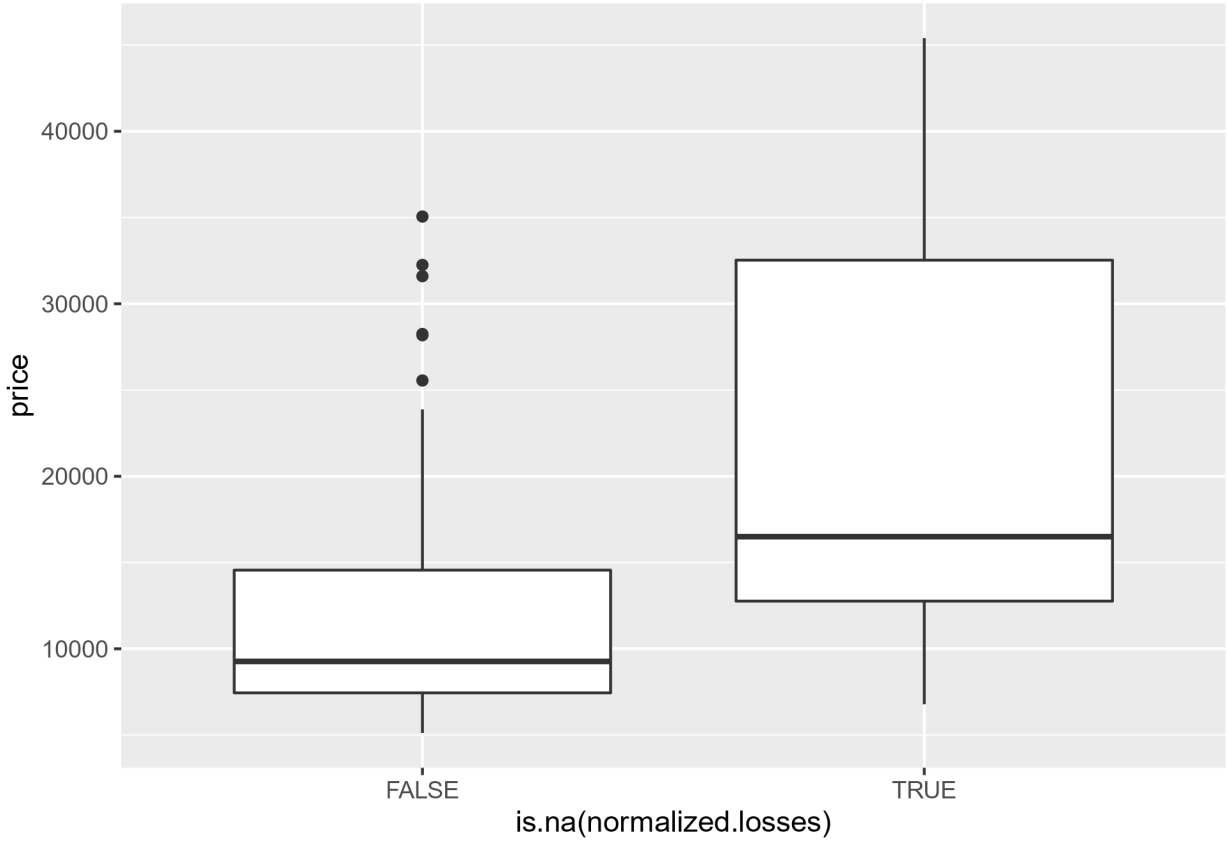


We deflate the data keeping in mind the `num.of.doors` variable to create the `deflated_data_doors` variable to use in our analysis. This will prevent biased weighing of data towards models with high numbers of variants. Note that this is different from `deflated_data`, which did not consider `num.of.doors` in determining which rows to keep.

```r
deflated_data_doors <- collapse_model(data, "num.of.doors")
```

2.3.1 Determining Randomness of Missing Values

As part of our analysis, we will be evaluating the viability of mean imputation on MCAR and MNAR data. If our data contains one form of missingness, the other will need to be simulated. The variable with missing values that we will be working most with is `normalized.losses`. The boxplot below shows that, within the scope of the entire dataset, `normalized.losses` does not seem to be random. `price` is one variable for which `normalized.losses` would be considered MNAR. This makes sense given the fact that some vehicle models may be entering the American import market for the first time, with no available domestic loss payment data. So while, systematically, it may not be random, it may still be random relative to certain variables.



Furthermore, a proportion test across the `make` variable indicates that, for that variable, `normalized.losses` should also be considered MNAR: the probability that data is missing is not equal across the different makes. The source for the custom `prop_test_across` function can be found in **Appendix 5.7**.

```
prop_test_across(deflated_data, "make")
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## na       2    3    2    0    0    0    3    1    1     3     1     1     0
## notna    0    2    3    3    4    7    0    1    6     3     0     5     9
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
## na        1     1     2     2     0     0     1     2     0
## notna     1     3     1     0     2     4    10     3     3
```

```
## Warning in prop.test(as.matrix(d)): Chi-squared approximation may be incorrect
```

```
##
##  22-sample test for equality of proportions without continuity
##  correction
##
```

```
## data:  as.matrix(d)
## X-squared = 44.774, df = 21, p-value = 0.001854
## alternative hypothesis: two.sided
## sample estimates:
##     prop 1     prop 2     prop 3     prop 4     prop 5     prop 6     prop 7
## 1.00000000 0.60000000 0.40000000 0.00000000 0.00000000 0.00000000 1.00000000
##     prop 8     prop 9    prop 10    prop 11    prop 12    prop 13    prop 14
## 0.50000000 0.14285714 0.50000000 1.00000000 0.16666667 0.00000000 0.50000000
##    prop 15    prop 16    prop 17    prop 18    prop 19    prop 20    prop 21
## 0.25000000 0.66666667 1.00000000 0.00000000 0.00000000 0.09090909 0.40000000
##    prop 22
## 0.00000000
```

```
# prop_test_across(data, "make")
# Commented out, no significant difference
```

However, for the current analysis, `num.of.doors`, a proportion test shows that we cannot reject the null hypothesis of completely random distribution of missingness across the two populations. Therefore, our data contains MCAR data for this analysis. Later, we will simulate MNAR missingness of `normalized.losses` data across the `num.of.doors` populations.

```
prop_test_across(deflated_data_doors, "num.of.doors")
```

```
##       [,1] [,2]
## na      15   18
## notna   39   32
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  as.matrix(d)
## X-squared = 0.47511, df = 1, p-value = 0.4906
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.2802914  0.1158469
## sample estimates:
##    prop 1    prop 2
## 0.2777778 0.3600000
```

```
# prop_test_across(data, "num.of.doors")
# Commented out, no significant difference
```

2.3.2 Performing Two Sample T-Test

Using the `full_t_test_doors` function (source for it and it's helper functions can be found in **Appendix 5.8**), we will simultaneously test the normality assumption, equal variance assumption, and if the former is met, conduct a two sample t-test. To demonstrate how deflating the data affects our tests and the consequent statistical inferences, we first perform the t-test on the uncollapsed `data` with all redundant rows present.

```
t_test_result_comparisons <- full_t_test_doors(data, FALSE, "mcar", "none")
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 68
## Data satisfies normality assumption: n >= 30; CLT.
## n = 95
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 0.873198470951505 , 2.13268298268207 )
```

```
##
##  Two Sample t-test
##
## data:  two.doors and four.doors
## t = 5.6559, df = 161, p-value = 6.912e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  19.01438 39.41565
## sample estimates:
## mean of x mean of y
##  138.8676  109.6526
##
## Standard error of difference of means: 5.165379433045
```

For comparison, here is the output for the two sample t-test on the `deflated_data_doors` variable. Although the conclusion remains the same, the results are strikingly different. The p-values differ by four four orders of magnitude and the standard error (and therefore range of the confidence interval) is significantly larger for the deflated data.

```
t_test_result_comparisons <-
  rbind(t_test_result_comparisons,
        full_t_test_doors(deflated_data_doors, TRUE, "mcar", "none"))
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 32
## Data satisfies normality assumption: n >= 30; CLT.
## n = 39
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 0.852990461217518 , 3.33838464155348 )
##
##  Two Sample t-test
##
## data:  two.doors and four.doors
## t = 3.7903, df = 69, p-value = 0.0003185
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  15.80855 50.93985
## sample estimates:
## mean of x mean of y
##  144.6562  111.2821
##
## Standard error of difference of means: 8.80507646888928
```

2.3.3 Performing Two Sample T-Test After Performing Mean Imputation on MCAR Values

The previous t-tests dealt with the missing data with a *complete-case analysis* approach: discarding any rows that had missing data in the relevant columns. Another strategy, *mean imputation*, substitutes missing values with the mean for that variable. Below are outputs of t-tests on `data` and `deflated_data_doors` after mean imputation was performed on their `normalized.losses` variables. `mean_impute_normalized_losses` source code in **Appendix 5.9**

```
data_mean_imputed <- mean_impute_normalized_losses(data)
deflated_data_doors_mean_imputed <- mean_impute_normalized_losses(deflated_data_doors)

t_test_result_comparisons <-
  rbind(t_test_result_comparisons,
```

```
        full_t_test_doors(data_mean_imputed, FALSE, "mcar", "mean"))
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 89
## Data satisfies normality assumption: n >= 30; CLT.
## n = 114
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 0.859122745047808 , 1.89822665128023 )
##
##  Two Sample t-test
##
## data:  two.doors and four.doors
## t = 5.5182, df = 201, p-value = 1.048e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  14.89512 31.45911
## sample estimates:
## mean of x mean of y
##  134.8876  111.7105
##
## Standard error of difference of means: 4.20014443492214
```

```
t_test_result_comparisons <-
  rbind(t_test_result_comparisons,
        full_t_test_doors(deflated_data_doors_mean_imputed, TRUE, "mcar", "mean"))
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 48
## Data satisfies normality assumption: n >= 30; CLT.
## n = 52
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 0.845096648332959 , 2.62686742082177 )
##
##  Two Sample t-test
##
## data:  two.doors and four.doors
## t = 3.6836, df = 98, p-value = 0.0003768
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  10.85265 36.20344
## sample estimates:
## mean of x mean of y
##  138.6458  115.1178
##
## Standard error of difference of means: 6.38730445709209
```

2.3.4 Performing Two Sample T-Test After Simulating MNAR Values

MCAR and MNAR affect analyses differently and interact differently with imputation methods. We simulated an MNAR scenario by applying the following algorithm: all vehicles in the data with four doors have their `normalized.losses` value replaced with `NA` with a probability of 0.5. This skews the data significantly. A proportion test can be used to confirm that, now, the `normalized.losses` values are MNAR with respect to the `num.of.doors` variable. In this case, the conclusion of the t-test upon this skewed data remains unchanged. This is, however, thanks to the nature of our censoring. Imagine a scenario that censored `normalized.losses` of two door cars with a probability proportional to the `normalized.losses` value. This

would simulate the possibility that very damaged cars, due to their expensive nature, have damages that go unreported to insurers. In such a scenario, the mean `normalized.losses` would drop for two door cars, changing significantly the observed mean difference of the two populations.

```
mnar_deflated_data_doors <- deflated_data_doors
set.seed(2838940)
rs <- runif(nrow(mnar_deflated_data_doors))
for(i in 1:nrow(mnar_deflated_data_doors)){
  if(mnar_deflated_data_doors[i,]$num.of.doors %in% "four"){
    if(rs[i] <= 0.5){
      mnar_deflated_data_doors[i,]$normalized.losses <- NA
    }
  }
}

prop_test_across(mnar_deflated_data_doors, "num.of.doors")
```

```
##       [,1] [,2]
## na      36   18
## notna   18   32
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  as.matrix(d)
## X-squared = 8.5907, df = 1, p-value = 0.003379
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.1043503 0.5089831
## sample estimates:
##    prop 1    prop 2
## 0.6666667 0.3600000
```

```
t_test_result_comparisons <-
  rbind(t_test_result_comparisons,
        full_t_test_doors(mnar_deflated_data_doors, TRUE, "mnar", "none"))
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 32
##
##  Shapiro-Wilk normality test
##
## data:  v
## W = 0.94905, p-value = 0.4104
##
## Data satisfies normality assumption: Shapiro-Wilk test.
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 0.900410061118974 , 5.02861845793183 )
##
##  Two Sample t-test
##
## data:  two.doors and four.doors
## t = 3.4001, df = 48, p-value = 0.001365
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  15.34290 59.74738
```

```
## sample estimates:
## mean of x mean of y
##  144.6562  107.1111
##
## Standard error of difference of means: 11.0424033860436
```

Another noticeable difference is in the standard error of the two test statistics. This is partly explained by the increased censoring and, therefore, smaller sample sizes. It is unlikely that our simulation affected the standard error, but MNAR scenarios could do this. Censoring being more likely closer a value is to the mean, for example, would increase the sample variance. These scenarios are why, when MNAR data is observed, modeling the missingness with respect to the data is crucial to accurate analysis.

2.3.5 Performing Two Sample T-Test After Performing Mean Imputation on MNAR Values

```
mnar_deflated_data_doors_mean_imputed <-
  mean_impute_normalized_losses(mnar_deflated_data_doors)

t_test_result_comparisons <-
  rbind(t_test_result_comparisons,
        full_t_test_doors(mnar_deflated_data_doors_mean_imputed, TRUE, "mnar", "mean"))
```

```
## Data satisfies normality assumption: n >= 30; CLT.
## n = 48
## Data satisfies normality assumption: n >= 30; CLT.
## n = 52
## Testing equal variance assumption...
## Variance ratio confidence interval: ( 1.71206650300618 , 5.32172471385223 )
##
##  Welch Two Sample t-test
##
## data:  two.doors and four.doors
## t = 3.0226, df = 73.858, p-value = 0.003444
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   5.868803 28.576333
## sample estimates:
## mean of x mean of y
##  140.2610  123.0385
##
## Standard error of difference of means: 5.69794502300515
```

2.3.6 Summarizing Effect of MCAR, MNAR, and Mean Imputation on Two Sample T-Test Outcomes

With the exception of reducing sample size, MCAR values have little to no effect on a two sample t-test (or other tests for population mean difference). MNAR, however, may or may not skew the results of mean testing. This implies that the nature of MNAR need be evaluate prior to performing testing on data, even if complete-case analysis is the chosen strategy.

Mean imputation has a very striking effect on the outcomes of testing on population means. This is a logical consequence of the strategy. Although the mean of the missing values might equal the mean of their substitutes, the resulting variance shrinks dramatically. For example, if the missing values are actually 1, 2, 3, 4, and 5 but we substitute in 3, 3, 3, 3, and 3, the mean remains unchanged but our sample variance goes dramatically to 0. The consequence this has on two-sample t-tests, whether data is missing MCAR or MNAR, is that the test is far more likely to reject a hypothesis of equal population means. Effectively, we reduce the power of the test. This is evident from the below summary. Notice how significantly the standard

error drops between tests where the only difference in the data was that mean imputation was performed.

```
t_test_result_comparisons
```

```
##   deflated randomness imputation.method variance.equality  ci.lower ci.upper
## 1    FALSE       mcar              none              TRUE 19.014382 39.41565
## 2     TRUE       mcar              none              TRUE 15.808552 50.93985
## 3    FALSE       mcar              mean              TRUE 14.895116 31.45911
## 4     TRUE       mcar              mean              TRUE 10.852647 36.20344
## 5     TRUE       mnar              none              TRUE 15.342899 59.74738
## 6     TRUE       mnar              mean             FALSE  5.868803 28.57633
##     std.err
## 1  5.165379
## 2  8.805076
## 3  4.200144
## 4  6.387304
## 5 11.042403
## 6  5.697945
```

2.3.7 Conclusion

Regarding the data, we can conclude that the mean `normalized.losses` differ between the populations of two-door and four-door vehicles, with the mean per-vehicle year losses being higher for two-door vehicles. This result is a reasonable expectation. Two door configurations are more common for convertibles, coupes, and luxury sports vehicles making them more expensive. This increased expense can correlate to increased expense in repair due to a lower supply in replacement parts, more frequent theft or vehicle damage, etc. Buyer beware: purchasing a two-door car means you will likely pay more on insurance than a four-door car.

2.4 Testing Impact of Normalized Losses on Symboling via GLM Methods

If we wish to model the symboling assignment to vehicles, we must run a logistic regression to generate a GLM since it is a categorical variable. Conveniently, symboling is ordinal, allowing for additional inference on our data and model.

2.4.1 Determining Randomness of Missing Values

The two factors we initially consider for modeling `symboling` are a vehicle's `normalized.losses` and `price`. As stated previously, many vehicles have missing normalized losses values. Prior to analysis, we must determine whether normalized losses are MCAR or MNAR relative to symboling. Below are the outcomes of two proportion tests. One on the original `data`, and one on `deflated_data`.

```
# Proportion test to check MCAR or MNAR between the two considered variables
prop_test_across(deflated_data, "symboling")
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
## na       0    2    9    7    3    5
## notna    1    8   21   20   11    9
```

```
## Warning in prop.test(as.matrix(d)): Chi-squared approximation may be incorrect
```

```
##
##  6-sample test for equality of proportions without continuity
##  correction
##
## data:  as.matrix(d)
## X-squared = 1.5278, df = 5, p-value = 0.9098
## alternative hypothesis: two.sided
```

```
## sample estimates:
##    prop 1    prop 2    prop 3    prop 4    prop 5    prop 6
## 0.0000000 0.2000000 0.3000000 0.2592593 0.2142857 0.3571429
```

```
# For comparison to inflated data
prop_test_across(data, "symboling")
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
## na       0    2   19    7    3   10
## notna    3   20   48   47   29   17
```

```
## Warning in prop.test(as.matrix(d)): Chi-squared approximation may be incorrect
```

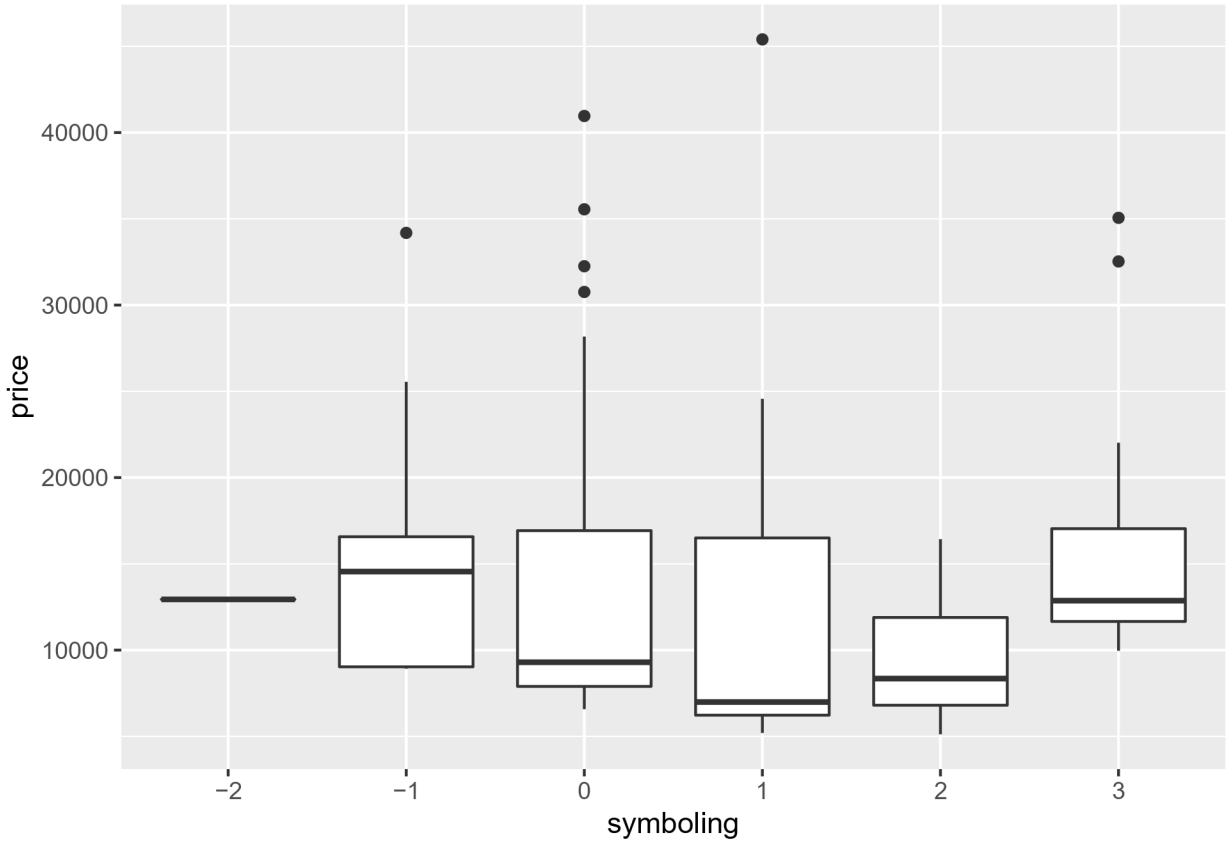
```
##
##  6-sample test for equality of proportions without continuity
##  correction
##
## data:  as.matrix(d)
## X-squared = 14.139, df = 5, p-value = 0.01475
## alternative hypothesis: two.sided
## sample estimates:
##     prop 1     prop 2     prop 3     prop 4     prop 5     prop 6
## 0.00000000 0.09090909 0.28358209 0.12962963 0.09375000 0.37037037
```

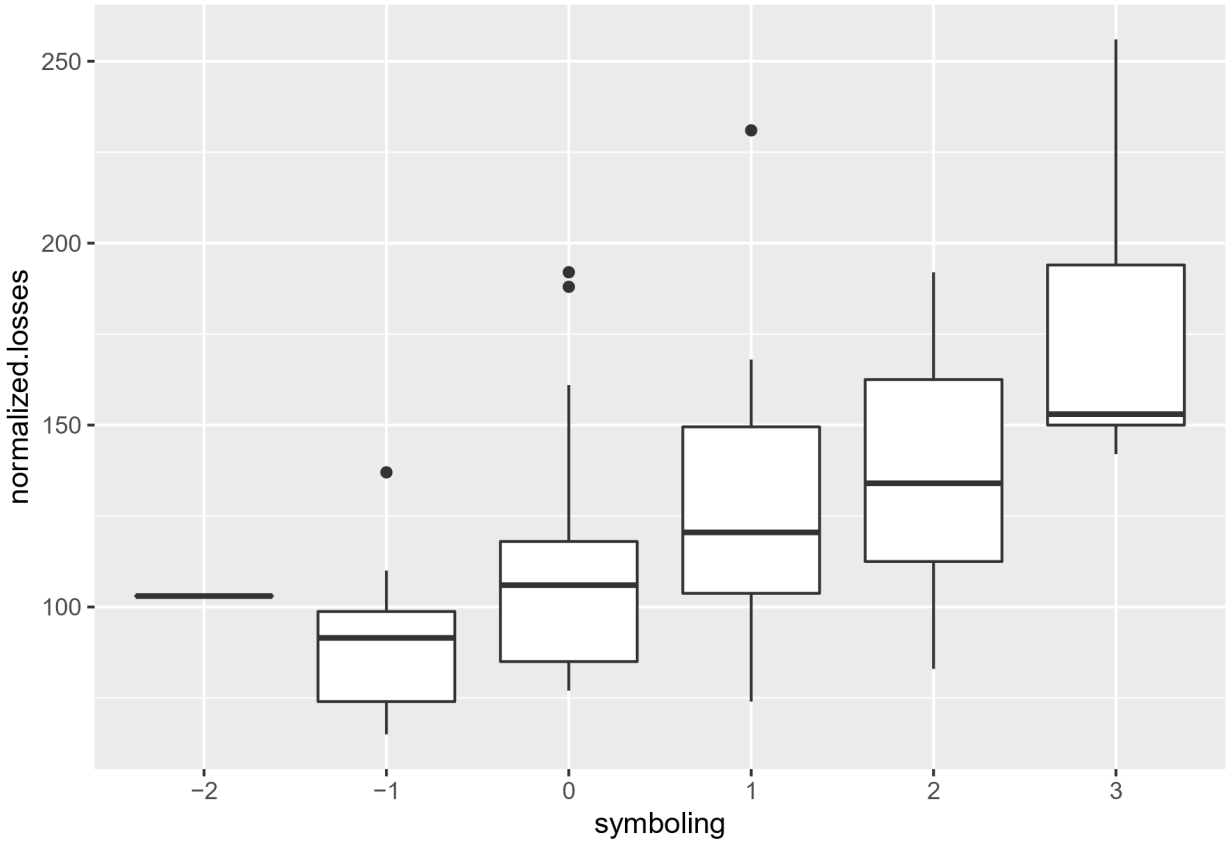
```
# prop_test_across(deflated_data, "make")
# prop_test_across(data, "make")
```

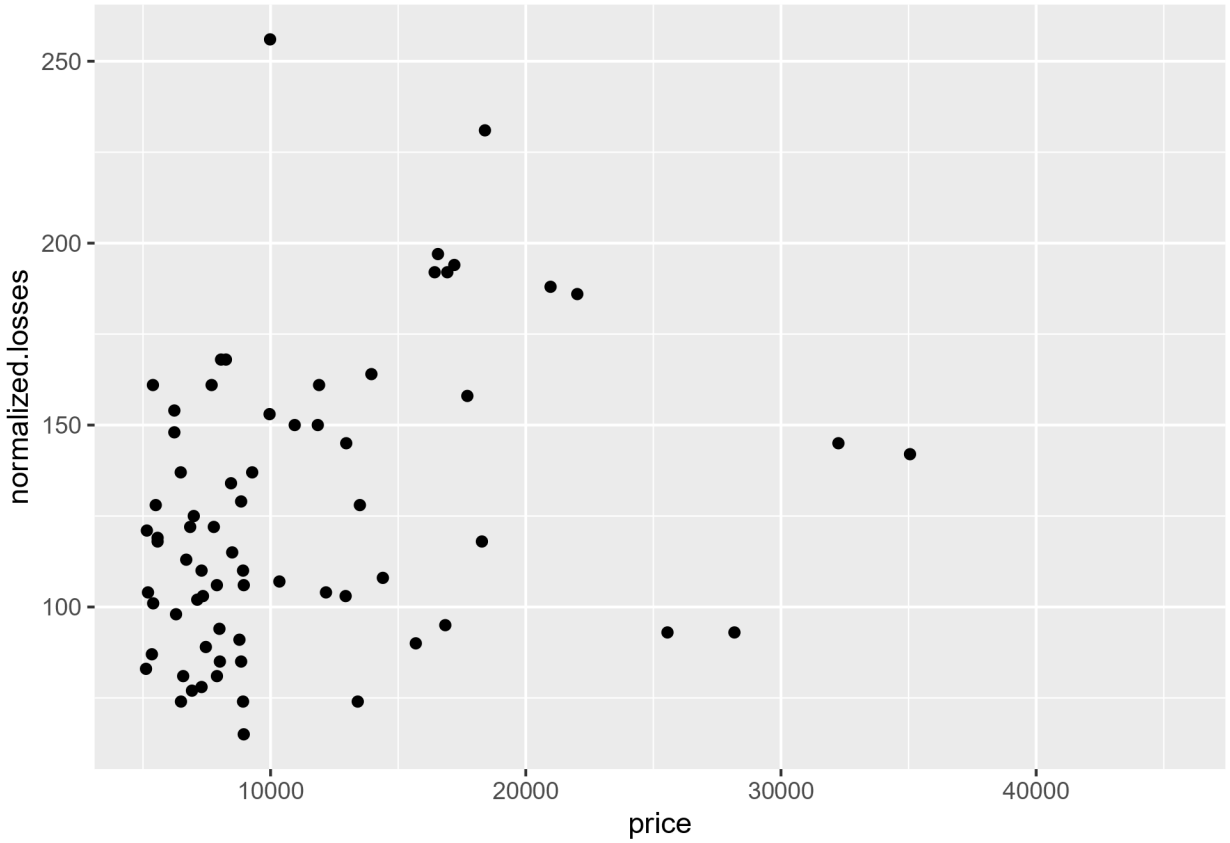
Here is where, for the first time, we see how significantly the row ties skew data. While it did not have an effect on the outcome of the mean comparison test form the previous section, it flips the conclusion of the proportion test that tests for MCAR. For this analysis, we will *not* be using un-deflated data.

2.4.2 Model Parameter Selection Via Likelihood Ratio Testing

After visually inspecting box-plots for `price` and `normalized.losses` against across the `symboling` groups, we have decided to consider both variables as possible parameters within our GLM. Price appears to move symboling towards the extremes as it increases and normalized losses and symboling appear to be directly proportional to one another. Additionally, although price and normalized losses appear relatively uncorrelated, interaction terms between the two may affect the symboling assignment as well.


40000
30000
20000
10000
price
−2
−1
0
1
2
3
symboling


250
200
150
100
normalized.losses
−2
−1
0
1
2
3
symboling



Below, we generate several GLMs with various parameters, including interraction terms. The ones that are removed from consideration on account of coefficient p-values can be found in **Appendix 5.10**. Likelihood ratio testing and comparison of AIC values suggests that the best model is `symboling ~ normalized.losses + price`.

```
m2.1 <- clm(symboling ~ normalized.losses, data = deflated_data)
summary(m2.1)
```

```
## formula: symboling ~ normalized.losses
## data:    deflated_data
##
##  link  threshold nobs logLik AIC    niter max.grad cond.H
##  logit flexible  70   -98.50 209.00 6(0)  5.36e-09 1.3e+06
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## normalized.losses  0.03123    0.00700   4.462 8.12e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -0.8948     1.2309  -0.727
## -1|0    1.5379     0.8208   1.874
## 0|1     3.4659     0.8645   4.009
## 1|2     4.9818     0.9645   5.165
## 2|3     6.2477     1.0734   5.821
```

```
## (26 observations deleted due to missingness)
```

```
m2.3 <- clm(symboling ~ normalized.losses + price, data = deflated_data)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.3)
```

```
## formula: symboling ~ normalized.losses + price
## data:    deflated_data
##
##  link  threshold nobs logLik AIC    niter max.grad cond.H
##  logit flexible  70   -95.70 205.41 7(0)  6.37e-11 1.2e+10
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## normalized.losses  3.712e-02  7.674e-03   4.837 1.32e-06 ***
## price             -9.573e-05  4.145e-05  -2.310   0.0209 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -1.4644     1.2600  -1.162
## -1|0    1.0650     0.8532   1.248
## 0|1     3.2059     0.8832   3.630
## 1|2     4.8303     0.9841   4.909
## 2|3     6.0680     1.0823   5.607
## (26 observations deleted due to missingness)
```

```
lrtest(m2.3, m2.1)
```

```
## Likelihood ratio test
##
## Model 1: symboling ~ normalized.losses + price
## Model 2: symboling ~ normalized.losses
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   7 -95.704
## 2   6 -98.499 -1 5.5911    0.01805 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Select m2.3 as primary model
```

2.4.3 Generating and Evaluating GLM with MCAR Data

As part of variable selection, a model was already generated. Below, we tally whether or not the model accurately predicts which symboling should be assigned to each vehicle. This model, which is built upon a complete-case analysis of MCAR data, accurately predicts 29 out of 96 symbols from 96 - 26 = 70 guesses. An accuracy of 29 / 70 = 41%. The code used to evaluate predicted values of the GLMs can be found in **Appendix 5.11**.

```
sum(is.na(predicted_prob[,1]))
```

```
## [1] 26
```

```
as.character(compare_symboling_fitting[,1]) == compare_symboling_fitting[,2]
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE
## [37] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
## [61] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [85] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
sum(as.character(compare_symboling_fitting[,1]) == compare_symboling_fitting[,2])
```

```
## [1] 29
```

```
## Accuracy =  0.414285714285714
```

2.4.4 Generating and Evaluating GLM with Mean Imputed MCAR Data

Next, we apply mean imputation to fill in the missing `normalized.losses` and re-regress our model. Same as with the previous model, we can evaluate it's performance by counting how many symbols it accurately predicts. This model was able to predict 33 out of 96 symbols. While it is tempting to say the model is "more accurate", the increase in successful guesses by 4 is mostly explained by this: the previous model made NO guess as to a vehicle's symboling when it's normalized loss was missing. The previous model made no guess for 26 values, effectively only making 70 guesses. This model makes no guess for only 3 rows, for a total of 93 guesses. The accuracy for this model is 33 / 93 = 35%. A slight decrease in accuracy. There are six possible guesses on symboling that the model can make. It made an additional 23 guesses. If the model guessed completely randomly for these additional 23 guesses, we would expect and additional 4 successful guesses, the exact difference between the number of successful guesses of the previous model and this one.

Another point of note is that the coefficients of this new model can no longer be assumed statistically significant. The p-value of the `price` coefficient slips to $0.06 > 0.05$. This is another risk of utilizing mean imputation. A parameter selection process which was sound on un-imputed data may no longer be sound once mean imputation is conducted.

```
deflated_data_mean_imputed <- mean_impute_normalized_losses(deflated_data)

m2.3_mean_imputed <- clm(symboling ~ price + normalized.losses, data = deflated_data_mean_imputed)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.3_mean_imputed)
```

```
## formula: symboling ~ price + normalized.losses
## data:    deflated_data_mean_imputed
##
##  link  threshold nobs logLik  AIC    niter max.grad cond.H
##  logit flexible  93   -133.27 280.55 7(0)  5.64e-11 2.3e+10
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## price             -4.331e-05  2.329e-05  -1.860   0.0629 .
## normalized.losses  3.183e-02  6.878e-03   4.628 3.69e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -1.5852     1.2381  -1.280
## -1|0    1.0434     0.8136   1.282
## 0|1     3.0601     0.8350   3.665
## 1|2     4.4314     0.8954   4.949
## 2|3     5.4807     0.9577   5.723
## (3 observations deleted due to missingness)
```

```
sum(is.na(predicted_prob[,1]))
```

```
## [1] 3
```

```
compare_symboling_fitting[,1] == compare_symboling_fitting[,2]
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [37] FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
## [61] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [85] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
```

```
sum(compare_symboling_fitting[,1] == compare_symboling_fitting[,2])
```

```
## [1] 33
```

```
## Accuracy =  0.354838709677419
```

2.4.5 Generating and Evaluating GLM with Simulated MNAR Data

We applied a similar method to that applied in our population mean analysis to create MNAR data: `normalized.losses` data is probabilistically censored at different rates for each level of `symboling`. A proportion test confirms that we have successfully created an MNAR scenario.

```
mnar_deflated_data_symboling <- deflated_data
set.seed(81234)
rs <- runif(nrow(mnar_deflated_data_symboling))
for(i in 1:nrow(mnar_deflated_data_symboling)){
  if(mnar_deflated_data_symboling[i,]$symboling == 3 & rs[i] <= 0.8){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
  }
  if(mnar_deflated_data_symboling[i,]$symboling == 2 & rs[i] <= 0.4){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
  }
  if(mnar_deflated_data_symboling[i,]$symboling == 1 & rs[i] <= 0.2){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
  }
  if(mnar_deflated_data_symboling[i,]$symboling == 0 & rs[i] <= 0.01){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
  }
  if(mnar_deflated_data_symboling[i,]$symboling == -1 & rs[i] <= 0.05){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
  }
  if(mnar_deflated_data_symboling[i,]$symboling == -2 & rs[i] <= 0.025){
    mnar_deflated_data_symboling[i,]$normalized.losses <- NA
```

```
  }
}

prop_test_across(mnar_deflated_data_symboling, "symboling")
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
## na       0    2    9   13    8   11
## notna    1    8   21   14    6    3
```

```
## Warning in prop.test(as.matrix(d)): Chi-squared approximation may be incorrect
```

```
##
##  6-sample test for equality of proportions without continuity
##  correction
##
## data:  as.matrix(d)
## X-squared = 13.398, df = 5, p-value = 0.01992
## alternative hypothesis: two.sided
## sample estimates:
##    prop 1    prop 2    prop 3    prop 4    prop 5    prop 6
## 0.0000000 0.2000000 0.3000000 0.4814815 0.5714286 0.7857143
```

The GLM generated on this data is, interestingly, far MORE accurate (26 correct out of 53 guesses on 96 rows).

```
m2.3_mnar <- clm(symboling ~ price + normalized.losses, data = mnar_deflated_data_symboling)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.3_mnar)
```

```
## formula: symboling ~ price + normalized.losses
## data:    mnar_deflated_data_symboling
##
##  link  threshold nobs logLik AIC    niter max.grad cond.H
##  logit flexible  53   -66.83 147.66 7(0)  4.37e-11 1.3e+10
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## price             -1.543e-04  4.965e-05  -3.107  0.00189 **
## normalized.losses  4.120e-02  1.012e-02   4.073 4.65e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -1.7689     1.4026  -1.261
## -1|0    0.9119     1.0514   0.867
## 0|1     3.5157     1.1300   3.111
## 1|2     5.2929     1.2605   4.199
## 2|3     6.6699     1.3898   4.799
## (43 observations deleted due to missingness)
```

```
sum(is.na(predicted_prob[,1]))
```

```
## [1] 43
```

```
as.character(compare_symboling_fitting[,1]) == compare_symboling_fitting[,2]
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [37] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
## [49] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [61] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
sum(as.character(compare_symboling_fitting[,1]) == compare_symboling_fitting[,2])
```

```
## [1] 26
```

```
## Accuracy =  0.490566037735849
```

2.4.6 Generating and Evaluating GLM with Mean Imputed Simulated MNAR Data

However, once we use mean imputation, we see once again a significant drop in accuracy. Additionally, the `price` coefficient can no longer be assumed statistically significant, possibly necessitating a redo of parameter selection for this model.

```
mnar_deflated_data_symboling_mean_imputed <- mean_impute_normalized_losses(mnar_deflated_data_symboling
m2.3_mnar_mean_imputed <- clm(symboling ~ price + normalized.losses, data = mnar_deflated_data_symbolin
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.3_mnar_mean_imputed)
```

```
## formula: symboling ~ price + normalized.losses
## data:    mnar_deflated_data_symboling_mean_imputed
##
##  link  threshold nobs logLik  AIC    niter max.grad cond.H
##  logit flexible  93   -140.40 294.80 7(0)  9.28e-11 3.2e+10
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## price             -2.817e-05  2.224e-05  -1.267  0.20515
## normalized.losses  2.767e-02  8.436e-03   3.279  0.00104 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -1.9180     1.3386  -1.433
## -1|0    0.6713     0.9631   0.697
## 0|1     2.5700     0.9847   2.610
## 1|2     3.8108     1.0254   3.717
## 2|3     4.7447     1.0608   4.473
## (3 observations deleted due to missingness)
```

```
sum(is.na(predicted_prob[,1]))
```

```
## [1] 3
```

```
compare_symboling_fitting[,1] == compare_symboling_fitting[,2]
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [37]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [49] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## [61] FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
## [73] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
sum(compare_symboling_fitting[,1] == compare_symboling_fitting[,2])
```

```
## [1] 32
```

```
## Accuracy =  0.344086021505376
```

2.4.7 Summarizing Effect of MCAR, MNAR, and Mean Imputation on Ordinal GLM Regression

Mean imputation reduces the accuracy of regressed GLMs. It naively assigns the mean whether or not it is appropriate according to the symboling. *Regression based deterministic imputation* would be a better strategy. Additionally, mean imputation hampers our ability to well select variables for our model.

2.4.8 Conclusion

We conclude that a vehicle's price and normalized losses likely contribute to which symboling it is assigned.

3. Comments

Our analyses are limited in several regards. Firstly, when evaluating for our GLM, we considered the missingness of one independent variable with respect to our dependent variable, *ignoring the possibility of the other independent variable influencing it's missingness*. When evaluating missingness, it needs to be done across all the variables that are being considered. An additional limitation was is our evaluation of GLMs. Since we are working with an ordinal categorical variable, a deviance can be measured, a "residual" of sorts. This would give a good estimate of each model's accuracy and precision. Additional limitations stem from a lack of information. With limited information, it is difficult to evaluate goodness-of-fit on data that wasn't used to regress the model, despite this being desirable when conducting data analysis.

Another limitation of this analysis is in it's practical application. Our second analysis aimed to predict what symboling was assigned by actuaries to vehicles. What factors do they consider when assigning a symbol to a vehicle model? However, a more practical analysis would evaluate how well the symboling predicts the normalized losses.

4. Acknowledgements

Special thanks to Professor Pei-Fen Kuan and Ziji Zhang for their guiding advice and teaching.

5. Appendix

5.1: Required Packages

```
require(foreign)
require(ggplot2)
require(Hmisc)
require(reshape2)
require(MASS)
require(ordinal)
require(rms)
require(lmtest)
require(tidyverse)
require(mice)
```

5.2: Short Summary Function

```
# Print summary function for long strings of coefficients
# Courtesy of LyzandeR
# https://stackoverflow.com/questions/32989379/print-the-summary-of-an-lm-or-fastlm-model-without-print
print.sum2 <-
  function (x, digits = max(3L, getOption("digits") - 3L), symbolic.cor = x$symbolic.cor,
            signif.stars = getOption("show.signif.stars"), ...){
    cat("\nCall:\n", paste(deparse(x$call), sep = "\n", collapse = "\n"),
        "\n\n", sep = "")
    resid <- x$residuals
    df <- x$df
    rdf <- df[2L]
    cat(if (!is.null(x$weights) && diff(range(x$weights)))
      "Weighted ", "Residuals:\n", sep = "")
    if (rdf > 5L) {
      nam <- c("Min", "1Q", "Median", "3Q", "Max")
      rq <- if (length(dim(resid)) == 2L)
        structure(apply(t(resid), 1L, quantile), dimnames = list(nam,
                                                                 dimnames(resid)[[2L]]))
      else {
        zz <- zapsmall(quantile(resid), digits + 1L)
        structure(zz, names = nam)
      }
      print(rq, digits = digits, ...)
    }
    else if (rdf > 0L) {
      print(resid, digits = digits, ...)
    }
    else {
      cat("ALL", df[1L], "residuals are 0: no residual degrees of freedom!")
      cat("\n")
    }
    if (length(x$aliased) == 0L) {
      #cat("\nNo Coefficients\n")
    }
    else {
      if (nsingular <- df[3L] - df[1L]) {
```

```
        #cat("\nCoefficients: (", nsingular, " not defined because of singularities)\n", sep = "")
      }
      else {
        #  cat("\nCoefficients:\n")
      }
      coefs <- x$coefficients
      if (!is.null(aliased <- x$aliased) && any(aliased)) {
        cn <- names(aliased)
        coefs <- matrix(NA, length(aliased), 4, dimnames = list(cn,
                                                                colnames(coefs)))
        coefs[!aliased, ] <- x$coefficients
      }
      #printCoefmat(coefs, digits = digits, signif.stars = signif.stars, na.print = "NA", ...)
    }
    cat("\nResidual standard error:", format(signif(x$sigma,
                                                  digits)), "on", rdf, "degrees of freedom")
    cat("\n")
    if (nzchar(mess <- naprint(x$na.action)))
      cat("  (", mess, ")\n", sep = "")
    if (!is.null(x$fstatistic)) {
      cat("Multiple R-squared: ", formatC(x$r.squared, digits = digits))
      cat(",\tAdjusted R-squared: ", formatC(x$adj.r.squared,
                                             digits = digits), "\nF-statistic:", formatC(x$fstatistic[1
                                                                                                      digits = digit
          x$fstatistic[3L], "DF,  p-value:", format.pval(pf(x$fstatistic[1L],
                                                            x$fstatistic[2L], x$fstatistic[3L], lower.t
                                                         digits = digits))
      cat("\n")
    }
    correl <- x$correlation
    if (!is.null(correl)) {
      p <- NCOL(correl)
      if (p > 1L) {
        cat("\nCorrelation of Coefficients:\n")
        if (is.logical(symbolic.cor) && symbolic.cor) {
          print(symnum(correl, abbr.colnames = NULL))
        }
        else {
          correl <- format(round(correl, 2), nsmall = 2,
                           digits = digits)
          correl[!lower.tri(correl)] <- ""
          print(correl[-1, -p, drop = FALSE], quote = FALSE)
        }
      }
    }
    cat("\n")
    invisible(x)
  }
```

5.3: Data Formatting Code

```
# Part 0: Importing and Formatting Data
data <- read.csv("imports-85.data.csv")
```

```r
data$normalized.losses <- as.integer(replace(data$normalized.losses, data$normalized.losses == "?", NA))
data$bore <- as.numeric(replace(data$bore, data$bore == "?", NA))
data$stroke <- as.numeric(replace(data$stroke, data$stroke == "?", NA))
data$horsepower <- as.integer(replace(data$horsepower, data$horsepower == "?", NA))
data$peak.rpm <- as.integer(replace(data$peak.rpm, data$peak.rpm == "?", NA))
data$price <- as.integer(replace(data$price, data$price == "?", NA))

for(i in names(data)){
  data[i] <- replace(data[i], data[i] == "?", NA)
}

to_be_factors <- c("symboling", "make", "fuel.type", "aspiration",
                   "num.of.doors", "body.style", "drive.wheels",
                   "engine.location", "engine.type", "num.of.cylinders",
                   "fuel.system")

for(i in to_be_factors){
  data[,i] <- as.factor(data[,i])
}
```

5.5: Deflation / Collapse Function Code

```
# Collapsing the ties
# Returns a model stripped of ties (repeat observations)
collapse_model <- function(df, keeper_variable = NULL){
  # Initializing data.frame
  out.df <- df[0,]
  for(i in 1:nrow(data)){
    # Interator df
    temp.df <- df[i,]
    # If current row is not the same model as any existing row in output df

    # Same model logical vector
    same_model <- out.df$make == temp.df$make &
      out.df$symboling %in% temp.df$symboling &
      out.df$normalized.losses %in% temp.df$normalized.losses

    if(sum(same_model) == 0){
      # Add row to output df
      out.df <- rbind(out.df, temp.df)
    }
    else if(!is.null(keeper_variable) & sum(! out.df[same_model,][,keeper_variable] %in% temp.df[,keepe
      out.df <- rbind(out.df, temp.df)
    }
  }
  return(out.df)
}
```

5.6

```
summary(data)
```

```
##  symboling normalized.losses         make      fuel.type   aspiration
##  -2: 3     Min.   : 65       toyota    : 32   diesel: 20   std  :168
##  -1:22     1st Qu.: 94       nissan    : 18   gas   :185   turbo: 37
##  0 :67     Median :115       mazda     : 17
##  1 :54     Mean   :122       honda     : 13
##  2 :32     3rd Qu.:150       mitsubishi: 13
##  3 :27     Max.   :256       subaru    : 12
##            NA's   :41        (Other)   :100
##  num.of.doors       body.style drive.wheels engine.location   wheel.base
##  four:114     convertible: 6   4wd:  9      front:202       Min.   : 86.60
##  two : 89     hardtop    : 8   fwd:120      rear :  3       1st Qu.: 94.50
##  NA's:  2     hatchback  :70   rwd: 76                      Median : 97.00
##               sedan      :96                                Mean   : 98.76
##               wagon      :25                                3rd Qu.:102.40
##                                                             Max.   :120.90
##
##      length          width           height       curb.weight   engine.type
##  Min.   :141.1   Min.   :60.30   Min.   :47.80   Min.   :1488   dohc : 12
##  1st Qu.:166.3   1st Qu.:64.10   1st Qu.:52.00   1st Qu.:2145   dohcv:  1
##  Median :173.2   Median :65.50   Median :54.10   Median :2414   l    : 12
##  Mean   :174.0   Mean   :65.91   Mean   :53.72   Mean   :2556   ohc  :148
##  3rd Qu.:183.1   3rd Qu.:66.90   3rd Qu.:55.50   3rd Qu.:2935   ohcf : 15
```

```
##  Max.   :208.1   Max.   :72.30   Max.   :59.80   Max.   :4066   ohcv : 13
##                                                                 rotor:  4
##  num.of.cylinders  engine.size     fuel.system      bore          stroke
##  eight :  5       Min.   : 61.0   mpfi   :94   Min.   :2.54   Min.   :2.070
##  five  : 11       1st Qu.: 97.0   2bbl   :66   1st Qu.:3.15   1st Qu.:3.110
##  four  :159       Median :120.0   idi    :20   Median :3.31   Median :3.290
##  six   : 24       Mean   :126.9   1bbl   :11   Mean   :3.33   Mean   :3.255
##  three :  1       3rd Qu.:141.0   spdi   : 9   3rd Qu.:3.59   3rd Qu.:3.410
##  twelve:  1       Max.   :326.0   4bbl   : 3   Max.   :3.94   Max.   :4.170
##  two   :  4                       (Other): 2   NA's   :4      NA's   :4
##  compression.ratio   horsepower       peak.rpm       city.mpg
##  Min.   : 7.00     Min.   : 48.0   Min.   :4150   Min.   :13.00
##  1st Qu.: 8.60     1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00
##  Median : 9.00     Median : 95.0   Median :5200   Median :24.00
##  Mean   :10.14     Mean   :104.3   Mean   :5125   Mean   :25.22
##  3rd Qu.: 9.40     3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00
##  Max.   :23.00     Max.   :288.0   Max.   :6600   Max.   :49.00
##                    NA's   :2       NA's   :2
##   highway.mpg        price
##  Min.   :16.00   Min.   : 5118
##  1st Qu.:25.00   1st Qu.: 7775
##  Median :30.00   Median :10295
##  Mean   :30.75   Mean   :13207
##  3rd Qu.:34.00   3rd Qu.:16500
##  Max.   :54.00   Max.   :45400
##                  NA's   :4
```

```
summary(deflated_data)
```

```
##  symboling normalized.losses            make     fuel.type  aspiration
##  -2: 1     Min.   : 65.00    toyota       :11   diesel: 5   std  :87
##  -1:10     1st Qu.: 94.25    nissan       : 9   gas   :91   turbo: 9
##  0 :30     Median :118.00    honda        : 7
##  1 :27     Mean   :125.44    mazda        : 7
##  2 :14     3rd Qu.:150.00    mercedes-benz: 6
##  3 :14     Max.   :256.00    mitsubishi   : 6
##            NA's   :26        (Other)      :50
##  num.of.doors       body.style drive.wheels engine.location   wheel.base
##  four:49      convertible: 3   4wd: 4       front:95        Min.   : 86.60
##  two :46      hardtop    : 5   fwd:56       rear : 1        1st Qu.: 94.50
##  NA's: 1      hatchback  :34   rwd:36                       Median : 96.55
##               sedan      :39                                Mean   : 98.77
##               wagon      :15                                3rd Qu.:101.50
##                                                             Max.   :120.90
##
##      length          width           height       curb.weight   engine.type
##  Min.   :141.1   Min.   :60.30   Min.   :48.80   Min.   :1488   dohc : 6
##  1st Qu.:167.3   1st Qu.:64.00   1st Qu.:51.60   1st Qu.:2163   dohcv: 1
##  Median :173.3   Median :65.50   Median :54.10   Median :2432   l    : 3
##  Mean   :174.0   Mean   :65.98   Mean   :53.75   Mean   :2572   ohc  :71
##  3rd Qu.:180.6   3rd Qu.:66.90   3rd Qu.:55.50   3rd Qu.:2952   ohcf : 5
##  Max.   :208.1   Max.   :72.30   Max.   :59.80   Max.   :4066   ohcv : 9
##                                                                 rotor: 1
##  num.of.cylinders  engine.size     fuel.system      bore          stroke
##  eight : 5        Min.   : 61.0   mpfi   :44   Min.   :2.680   Min.   :2.360
```

```
##  five  : 7        1st Qu.: 97.0   2bbl   :33   1st Qu.:3.090   1st Qu.:3.110
##  four  :68        Median :120.5   1bbl   : 6   Median :3.310   Median :3.290
##  six   :14        Mean   :131.7   idi    : 5   Mean   :3.323   Mean   :3.287
##  three : 1        3rd Qu.:151.2   spdi   : 5   3rd Qu.:3.540   3rd Qu.:3.405
##  twelve: 0        Max.   :308.0   4bbl   : 1   Max.   :3.940   Max.   :4.170
##  two   : 1                        (Other): 2   NA's   :1       NA's   :1
##  compression.ratio   horsepower       peak.rpm       city.mpg
##  Min.   : 7.000    Min.   : 48.0   Min.   :4200   Min.   :14.00
##  1st Qu.: 8.500    1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00
##  Median : 9.000    Median : 98.5   Median :5200   Median :23.50
##  Mean   : 9.548    Mean   :106.6   Mean   :5148   Mean   :25.04
##  3rd Qu.: 9.400    3rd Qu.:123.0   3rd Qu.:5500   3rd Qu.:30.25
##  Max.   :23.000    Max.   :288.0   Max.   :6000   Max.   :49.00
##                    NA's   :2       NA's   :2
##   highway.mpg        price
##  Min.   :16.00   Min.   : 5118
##  1st Qu.:25.00   1st Qu.: 7349
##  Median :29.50   Median : 9980
##  Mean   :30.56   Mean   :13224
##  3rd Qu.:34.50   3rd Qu.:16430
##  Max.   :54.00   Max.   :45400
##                  NA's   :3
```

5.7: Custom Proportion Test Code

```
# Setting up a proportion test to prove that random values in normalized losses are MNAR
prop_test_across <- function(df, f, g = "normalized.losses"){
  d <- data.frame(na = integer(), notna = integer())

  for(l in levels(df[,f])){
    tt <- table(is.na(df[df[,f] == l,][,g]))
    d <- rbind(d, data.frame(na = tt['TRUE'], notna = tt['FALSE'], row.names = NULL))
  }
  d[is.na(d)] <- 0
  print(t(d))
  prop.test(as.matrix(d))
}
```

5.8: Custom T-Test Code

```
t_test_assumption_check <- function(v){
  if(length(v) - sum(is.na(v)) < 30){
    s <- shapiro.test(v)
    print(s)
    if(s$p.value < 0.05){
      cat("Assumptions necessary for t.test fail to hold:
            two.door data insufficient and non-normal.\n")
      return(FALSE)
    }
    else{
      cat("Data satisfies normality assumption: Shapiro-Wilk test.\n")
      return(TRUE)
    }
```

```
  }
  else{
    cat("Data satisfies normality assumption: n >= 30; CLT.\n")
    cat(paste(c("n = ", length(v) - sum(is.na(v)),"\n"), collapse = ""))
  }
  return(TRUE)
}

t_test_doors <- function(df){
  # First test length of available data
  two.doors <- filter(df, num.of.doors %in% "two")$normalized.losses
  four.doors <- filter(df, num.of.doors %in% "four")$normalized.losses

  # For each length of data that is shorter than 30, conduct a normality test
  if(!t_test_assumption_check(two.doors)){
    return(0)
  }

  if(!t_test_assumption_check(four.doors)){
    return(0)
  }

  # If either normality test fails, return an error
  v.test <- var.test(two.doors, four.doors)
  cat("Testing equal variance assumption...\n")
  cat(c("Variance ratio confidence interval: (",v.test$conf.int[1],",",v.test$conf.int[2],")\n"), colla

  eq.var.assumption <- (v.test$p.value > 0.05)

  t <- t.test(two.doors, four.doors, var.equal = eq.var.assumption)
  print(t)
  cat(c("Standard error of difference of means:", t$stderr),"\n")
  invisible(t)
  return(
    data.frame(variance.equality = eq.var.assumption,
               ci.lower = t$conf.int[1],
               ci.upper = t$conf.int[2],
               std.err = t$stderr)
  )
}

full_t_test_doors <- function(df,
                              deflated = c(TRUE, FALSE),
                              randomness = c("mcar", "mnar"),
                              imputation.method = c("none", "mean", "random")){
  return(cbind(
    data.frame(deflated = deflated,
               randomness = randomness,
               imputation.method = imputation.method),
    t_test_doors(df)
  ))
}
```

5.9: Mean Imputation Function Code

```
# Mean imputation
mean_impute_normalized_losses <- function(df){
  output.df <- df
  output.df$normalized.losses <-
    replace(output.df$normalized.losses,
            is.na(output.df$normalized.losses),
            mean(output.df$normalized.losses, na.rm = TRUE))
  return(output.df)
}
```

5.10: Other GLMs Removed From Consideration

```
m2 <- clm(symboling ~ price, data = deflated_data)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2)
```

```
## formula: symboling ~ price
## data:    deflated_data
##
##  link  threshold nobs logLik  AIC    niter max.grad cond.H
##  logit flexible  93   -146.22 304.43 7(0)  6.55e-11 8.1e+09
##
## Coefficients:
##         Estimate Std. Error z value Pr(>|z|)
## price -1.575e-05  2.136e-05  -0.738    0.461
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -4.7400     1.0497  -4.516
## -1|0   -2.2230     0.4363  -5.096
## 0|1    -0.4806     0.3424  -1.404
## 1|2     0.6531     0.3409   1.916
## 2|3     1.5394     0.3875   3.973
## (3 observations deleted due to missingness)
```

```
m2.2 <- clm(symboling ~ normalized.losses:price, data = deflated_data)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.2)
```

```
## formula: symboling ~ normalized.losses:price
## data:    deflated_data
##
##  link  threshold nobs logLik  AIC    niter max.grad cond.H
##  logit flexible  70   -108.91 229.82 7(0)  2.79e-09 7.2e+13
##
## Coefficients:
```

```
##                           Estimate Std. Error z value Pr(>|z|)
## normalized.losses:price 4.007e-07  2.123e-07   1.887   0.0592 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -3.7223     1.0404  -3.578
## -1|0   -1.3976     0.4435  -3.151
## 0|1     0.2314     0.3636   0.636
## 1|2     1.4558     0.3948   3.688
## 2|3     2.5189     0.4938   5.101
## (26 observations deleted due to missingness)
```

```
m2.4 <- clm(symboling ~ normalized.losses + normalized.losses:price, data = deflated_data)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.4)
```

```
## formula: symboling ~ normalized.losses + normalized.losses:price
## data:    deflated_data
##
##  link  threshold nobs logLik AIC    niter max.grad cond.H
##  logit flexible  70   -96.38 206.75 7(0)  7.68e-09 2.6e+14
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## normalized.losses         4.380e-02  9.633e-03   4.547 5.45e-06 ***
## normalized.losses:price -6.337e-07  3.120e-07  -2.031   0.0422 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -0.4033     1.2566  -0.321
## -1|0    2.0855     0.8728   2.389
## 0|1     4.1668     0.9558   4.360
## 1|2     5.7829     1.0758   5.375
## 2|3     7.0181     1.1620   6.040
## (26 observations deleted due to missingness)
```

```
m2.5 <- clm(symboling ~ normalized.losses*price, data = deflated_data)
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary(m2.5)
```

```
## formula: symboling ~ normalized.losses * price
## data:    deflated_data
##
##  link  threshold nobs logLik AIC    niter max.grad cond.H
##  logit flexible  70   -95.35 206.70 7(0)  2.33e-10 1.1e+15
```

```
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## normalized.losses         2.519e-02  1.596e-02   1.578    0.115
## price                    -2.237e-04  1.599e-04  -1.399    0.162
## normalized.losses:price   1.005e-06  1.204e-06   0.835    0.404
##
## Threshold coefficients:
##       Estimate Std. Error z value
## -2|-1  -2.9986     2.2329  -1.343
## -1|0   -0.4153     1.9556  -0.212
## 0|1     1.7628     1.9200   0.918
## 1|2     3.3720     1.9790   1.704
## 2|3     4.6223     2.0163   2.292
## (26 observations deleted due to missingness)
```

5.11: Fitting the GLMs

```
predicted_prob <- predict(m2.3, deflated_data[,c("normalized.losses", "price")])$fit
most_likely_symbol <- apply(predicted_prob, 1, function(x) which(x == max(x)))
names(most_likely_symbol) <- NULL
is.na(most_likely_symbol) <- lengths(most_likely_symbol) == 0
compare_symboling_fitting <- data.frame(deflated_data$symboling, names(unlist(most_likely_symbol)))
```

```
predicted_prob <- predict(m2.3_mean_imputed, deflated_data_mean_imputed[,c("normalized.losses", "price"]
most_likely_symbol <- apply(predicted_prob, 1, function(x) which(x == max(x)))
names(most_likely_symbol) <- NULL
# most_likely_symbol <- most_likely_symbol - 4
is.na(most_likely_symbol) <- lengths(most_likely_symbol) == 0
compare_symboling_fitting <- data.frame(deflated_data_mean_imputed$symboling, names(unlist(most_likely_
```

```
predicted_prob <- predict(m2.3_mnar, mnar_deflated_data_symboling[,c("normalized.losses", "price")])$fi
most_likely_symbol <- apply(predicted_prob, 1, function(x) which(x == max(x)))
names(most_likely_symbol) <- NULL
is.na(most_likely_symbol) <- lengths(most_likely_symbol) == 0
compare_symboling_fitting <- data.frame(mnar_deflated_data_symboling$symboling, names(unlist(most_likel
```

```
predicted_prob <- predict(m2.3_mnar_mean_imputed, mnar_deflated_data_symboling_mean_imputed[,c("normali
most_likely_symbol <- apply(predicted_prob, 1, function(x) which(x == max(x)))
names(most_likely_symbol) <- NULL
is.na(most_likely_symbol) <- lengths(most_likely_symbol) == 0
compare_symboling_fitting <- data.frame(mnar_deflated_data_symboling_mean_imputed$symboling, names(unli
```