LRMoE RealData Preliminary

Spark Tseung

March 3, 2020

Introduction

This document is Part I of a demo series of the `LRMoE` (Logit-weighted Reduced Mixture-of-Experts) package on a real dataset. By analysing a French motor third-party liability insurance dataset in `CASdatasets`, we will demonstrate the fitting procedure, diagnostics, visualization and predictive functions of the `LRMoE` package. In this document, we focus on data preprocessing, exploratory analysis and parameter initialization.

Data Preprocessing

In this section, we first perform some exploratory analysis on the dataset. Description of variables are given in the `CASdatasets` user manual, and will not be repeated here. `freMTPLfreq` contains records of policy ID, policyholder information and number of claims, while `freMTPLsev` contains only policy ID and claim amounts for those who have claims. The `PolicyID` variable serves as a unique identifier that links these two datasets. We note that some policyholders have multiple claims, hence `freMTPLsev$PolicyID` contains duplicated records. We aggregate claim amounts by policy and merge these two datasets.

```
# Load data
data(freMTPLfreq, freMTPLsev)

# Aggregate claim amounts
sev.aggre = aggregate(freMTPLsev$ClaimAmount,
                      by = list(PolicyID = freMTPLsev$PolicyID), FUN = "sum")
colnames(sev.aggre)[2] = "ClaimAmount"

# Match two datasets by PolicyID. NA values correspond to no claims.
df.all = merge(freMTPLfreq, sev.aggre, by = "PolicyID", all = TRUE)
df.all$ClaimAmount[is.na(df.all$ClaimAmount)] = 0
df.all.pos = df.all[which(df.all$ClaimAmount>0),]

df.temp = data.matrix(df.all)
```

The response variables are `ClaimNb` and `ClaimAmount`. For simplicity, we exclude `Exposure` and `Density` from the covariates.

```
# Drop covariates: Exposure and Density
df.all = df.all[,-which(names(df.all) %in% c("Exposure", "Density"))]
df.all.pos = df.all.pos[,-which(names(df.all.pos) %in% c("Exposure", "Density"))]
```

Exploratory Analysis

We first calculate some summary statistics of `ClaimNb` and `ClaimAmount`. Less than 4% of policyholders have at least one claim, and the claim amount is right-skewed with a heavy tail.

```
sample.size = nrow(df.all)

# Proportion of zero claim
sum(df.all$ClaimNb==0)/sample.size
```

```
## [1] 0.9627513
```

```
sum(df.all$ClaimAmount==0)/sample.size
```

```
## [1] 0.9627513
```

```
# Other statistics
summary(df.all$ClaimAmount)
```

```
##      Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
##       0.0       0.0       0.0      83.4       0.0 2036833.0
```

```
sd(df.all$ClaimNb[which(df.all$ClaimNb>0)])
```

```
## [1] 0.2314458
```

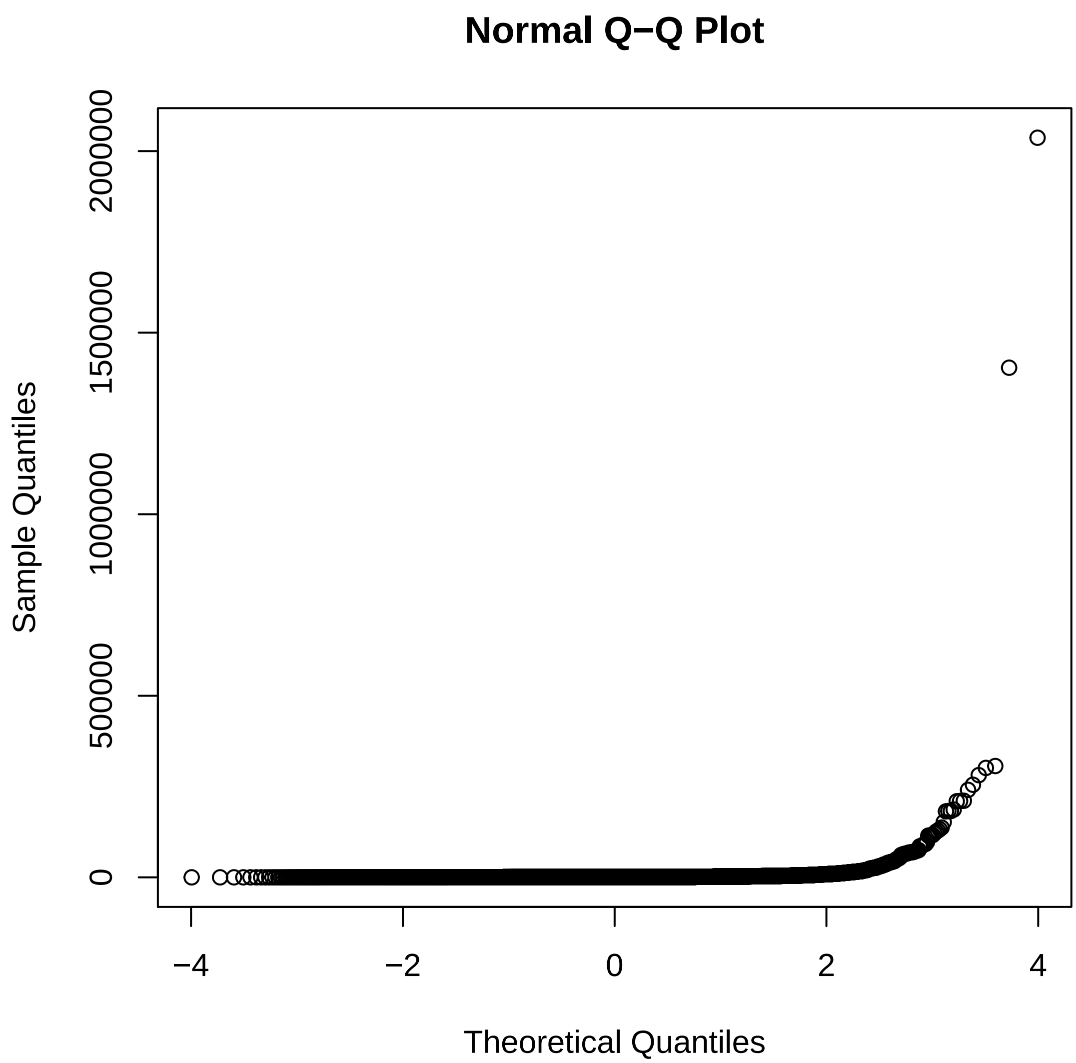
```
sd(df.all$ClaimAmount[which(df.all$ClaimAmount>0)])
```

```
## [1] 21612.28
```

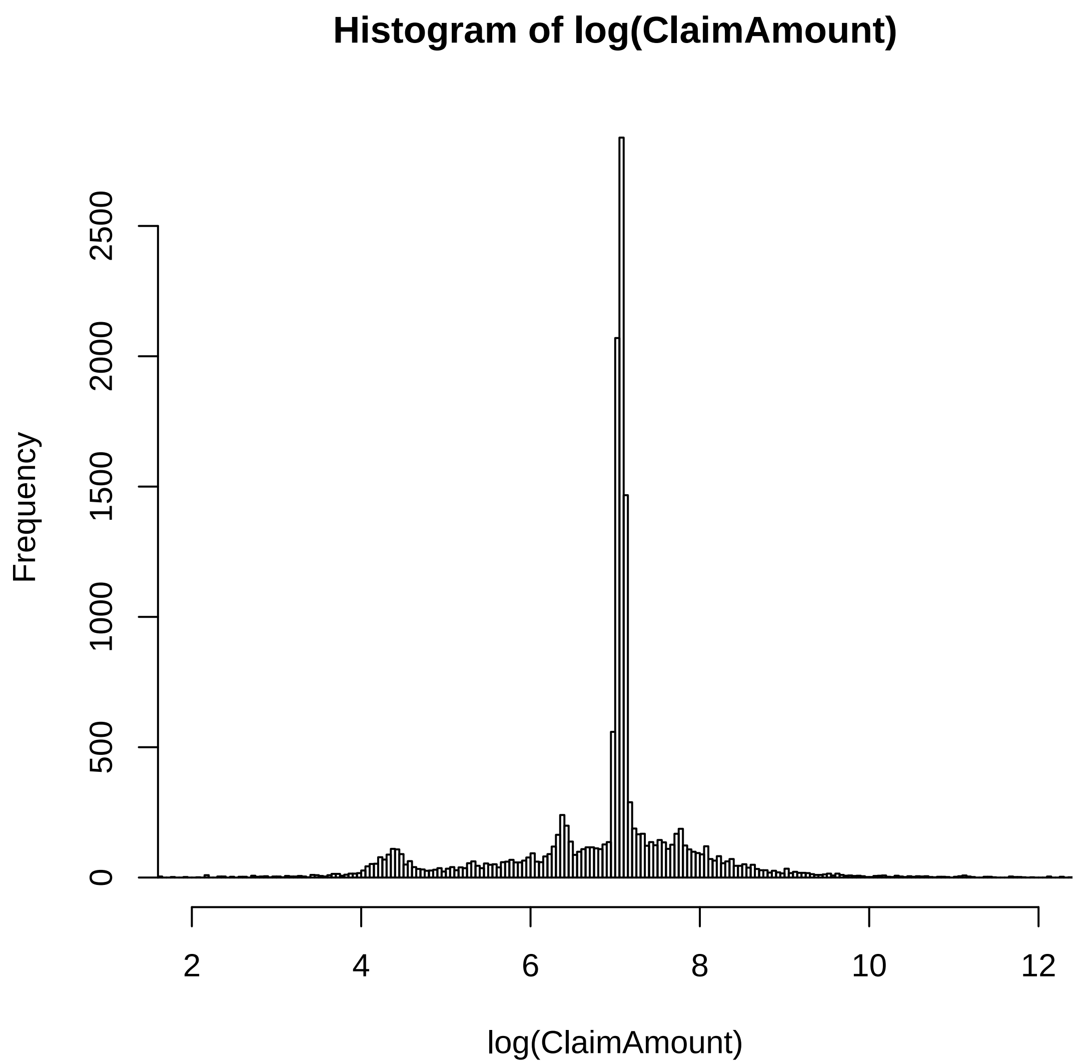
```
kurtosis(df.all$ClaimAmount[which(df.all$ClaimAmount>0)])
```

```
## [1] 6264.353
```

```r
# QQ Plot
qqnorm(df.all$ClaimAmount[which(df.all$ClaimAmount>0 )])
```

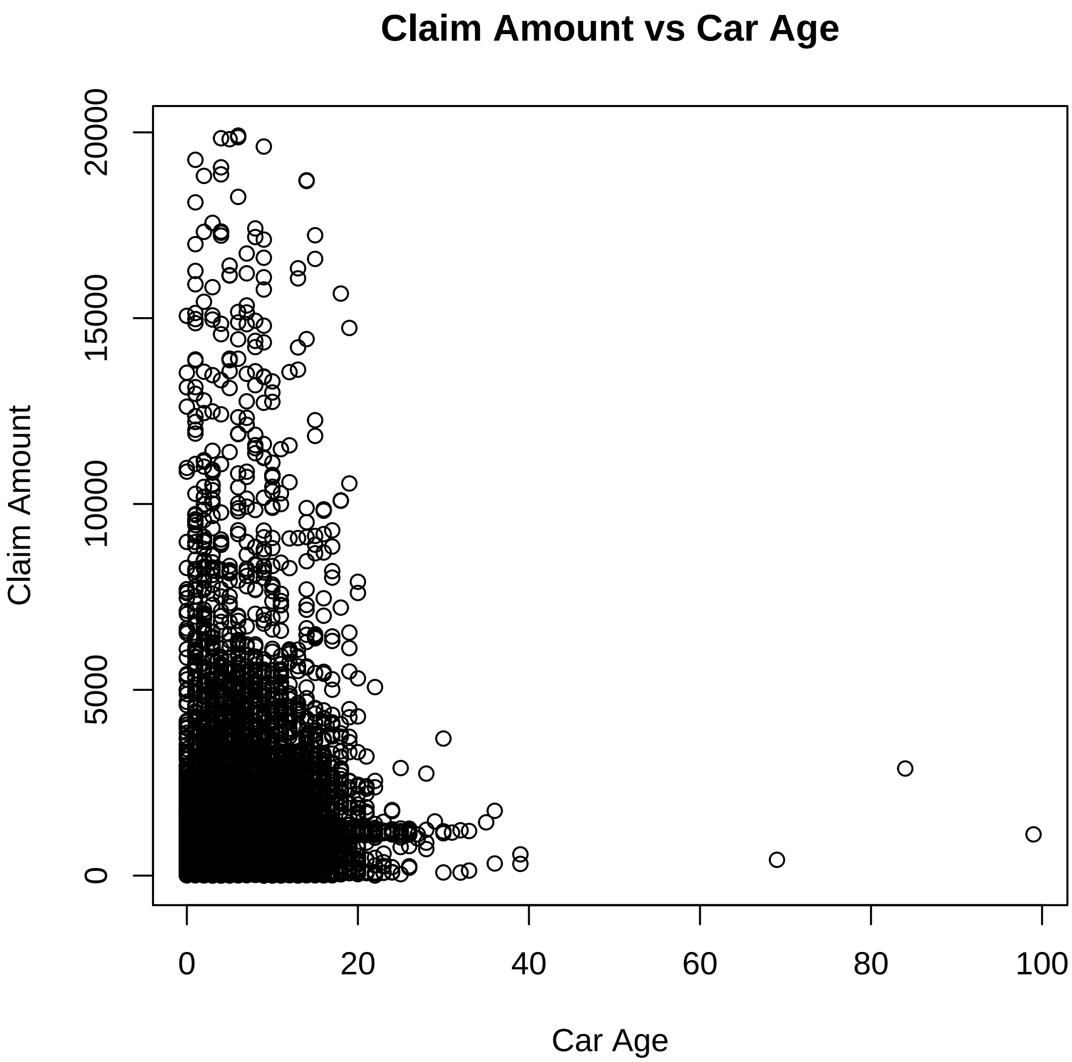


```r
# Histogram
hist(log(df.all$ClaimAmount[which(df.all$ClaimAmount>0 & df.all$ClaimAmount<500000)]),
     breaks = 200, probability = FALSE,
     xlim = c(2, 12),
     main = "Histogram of log(ClaimAmount)", xlab = "log(ClaimAmount)")
```



We can also plot the response variable against each of the covariates. For better presentation, only positive claim amounts less than 20,000 (or 5,000) are plotted.

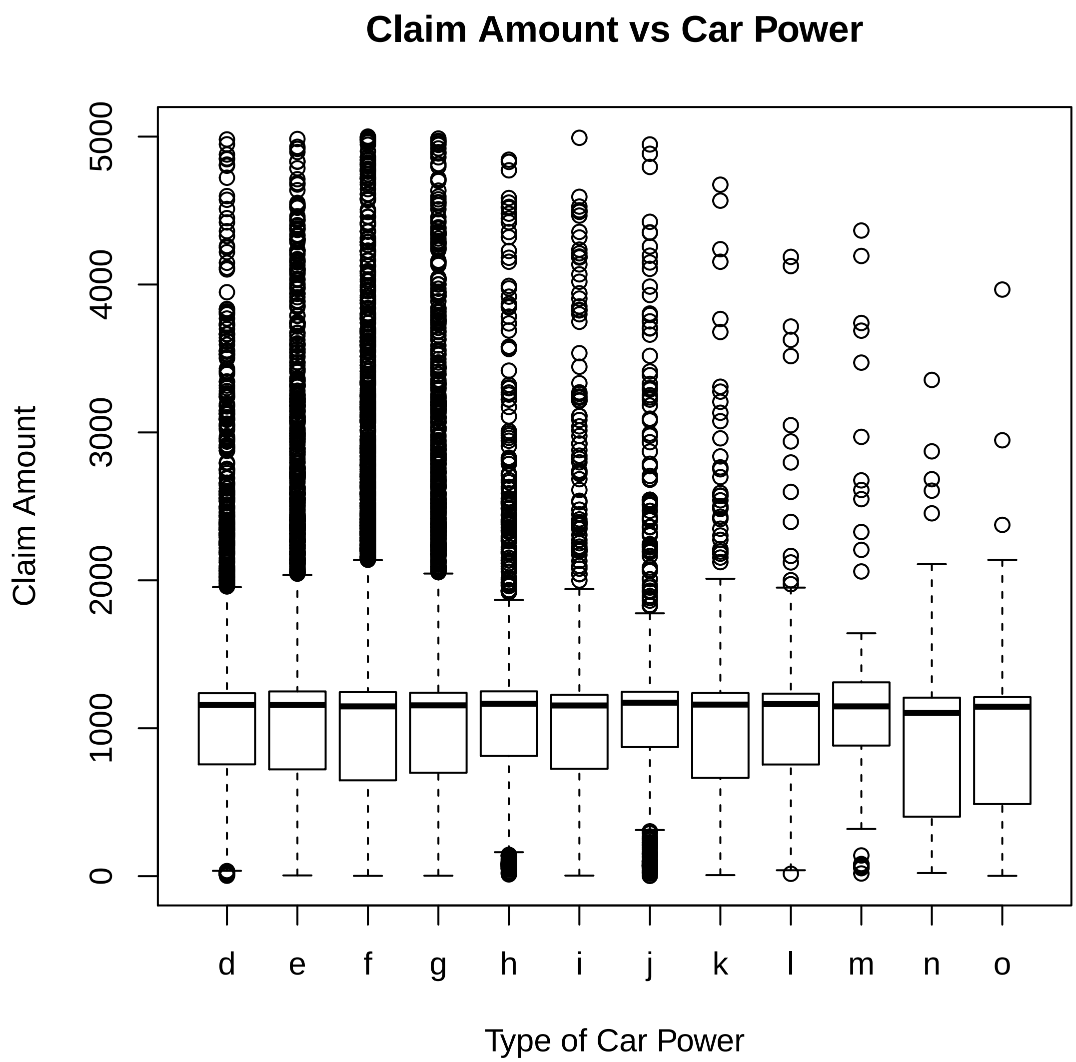
- **Car's Age**:



- **Driver's Age**:


Claim Amount vs Driver's Age
Claim Amount
0
5000
10000
15000
20000
20
40
60
80
100
Driver's Age

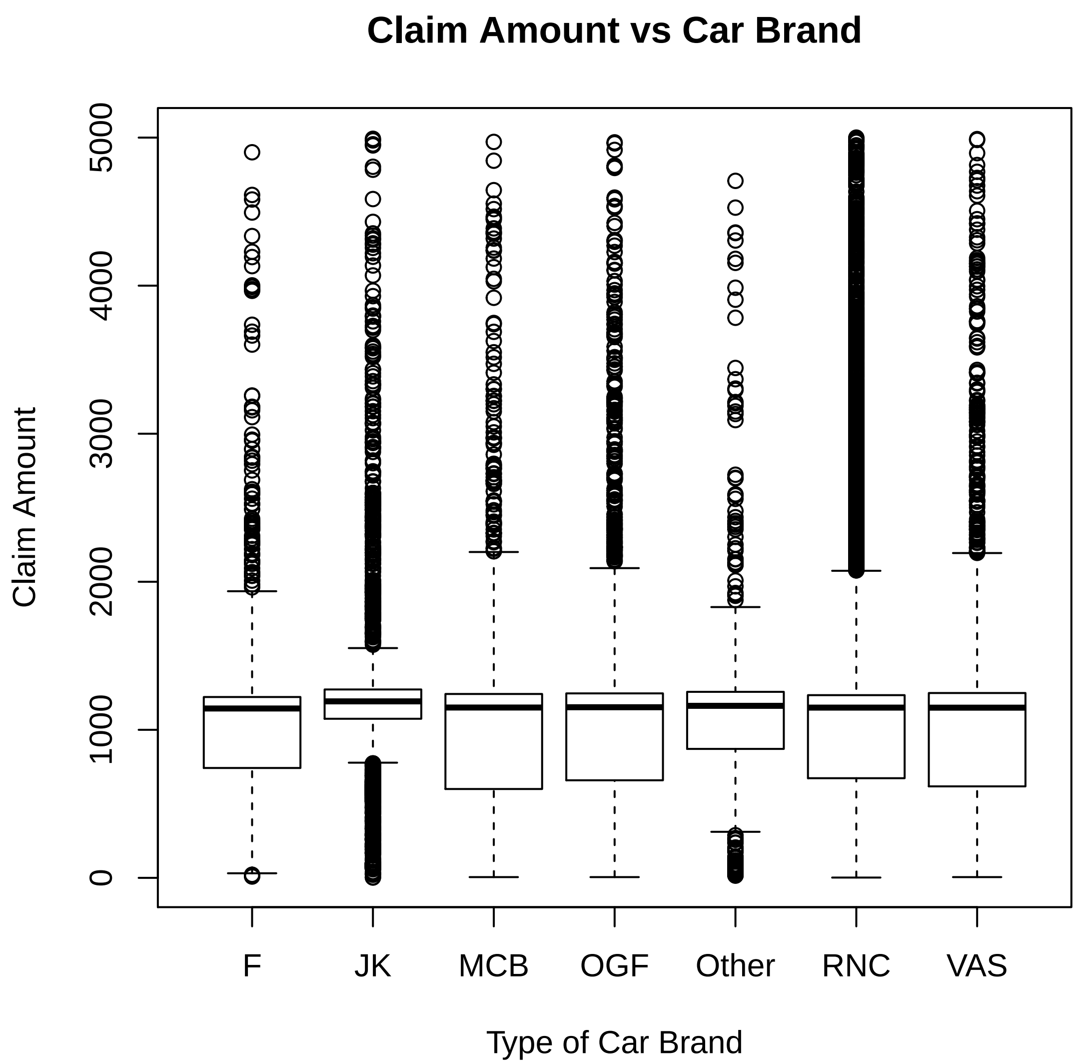
- **Car Power**:



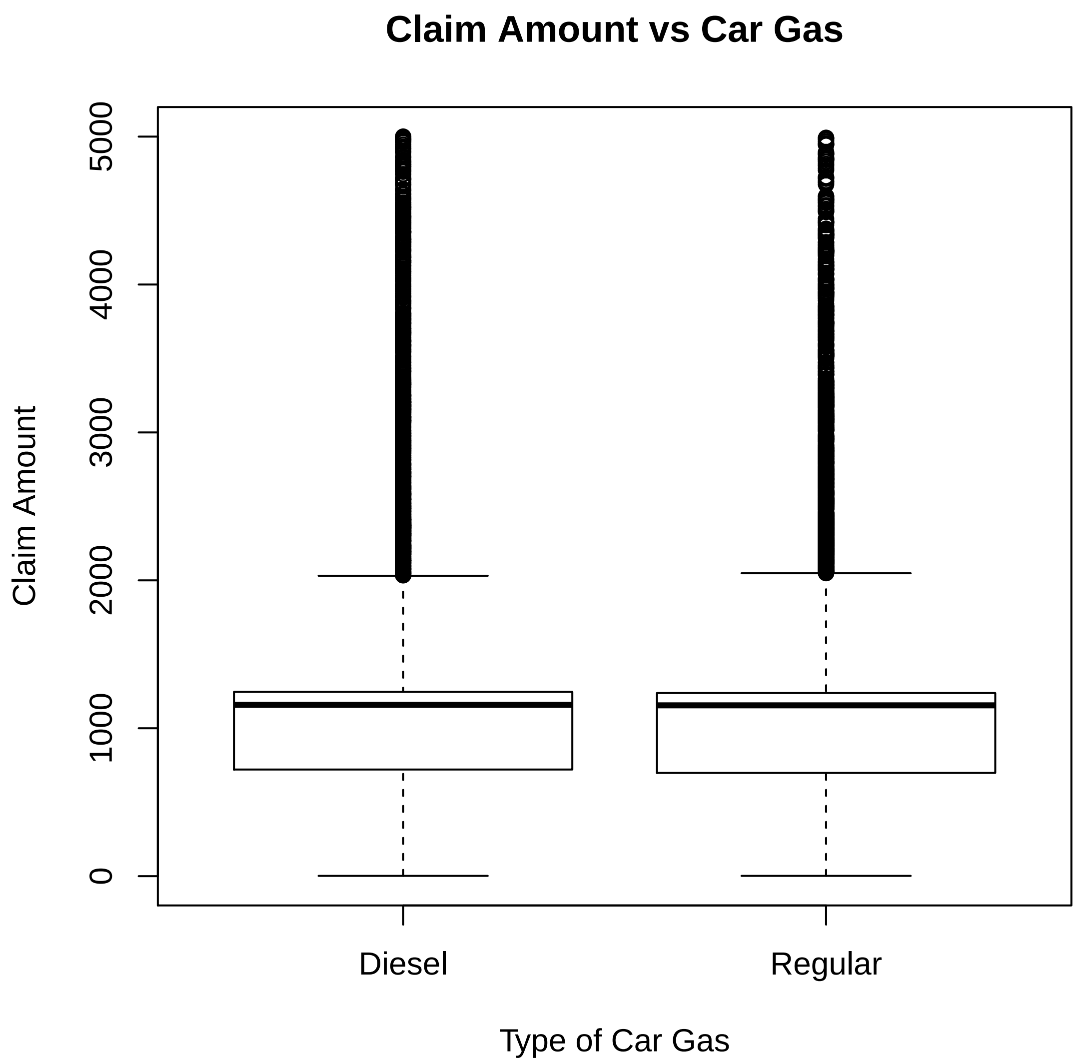
Claim Amount vs Car Power
Claim Amount
0
1000
2000
3000
4000
5000
d
e
f
g
h
i
j
k
l
m
n
o
Type of Car Power

- **Brand**:

```
## [1] "Fiat"                              "Japanese (except Nissan) or Korean"
## [3] "Mercedes, Chrysler or BMW"         "Opel, General Motors or Ford"
## [5] "other"                             "Renault, Nissan or Citroen"
## [7] "Volkswagen, Audi, Skoda or Seat"
```



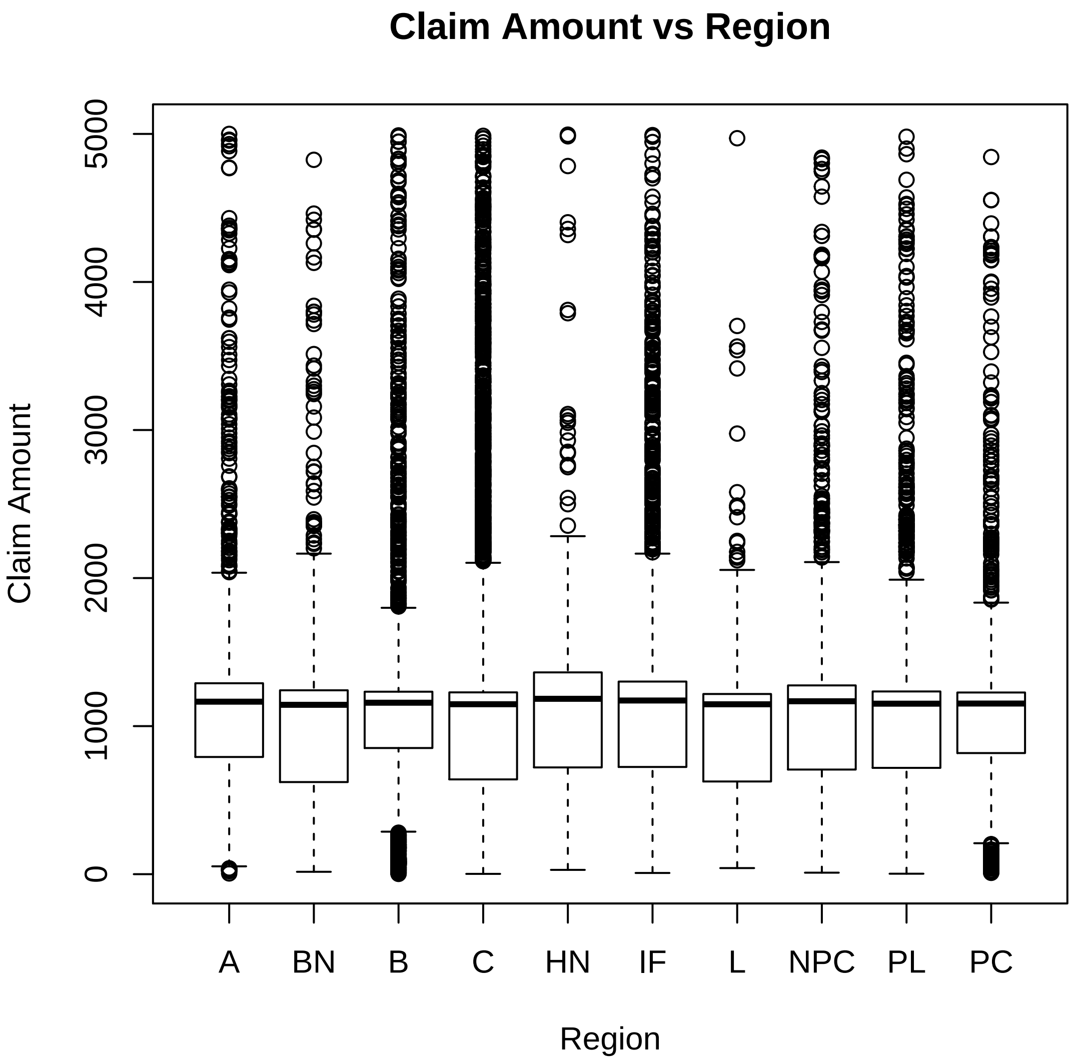
- **Gas**:



Claim Amount vs Car Gas
Claim Amount
0
1000
2000
3000
4000
5000
Diesel
Regular
Type of Car Gas

- **Region**:

```
##  [1] "Aquitaine"          "Basse-Normandie"    "Bretagne"
##  [4] "Centre"             "Haute-Normandie"    "Ile-de-France"
##  [7] "Limousin"           "Nord-Pas-de-Calais" "Pays-de-la-Loire"
## [10] "Poitou-Charentes"
```



Formatting Data

For illustration purposes, we will fit an LRMoE with one respose variable `ClaimAmount` only. The following code shows how to format input matrices `Y` and `X` required by the `LRMoE` package.

```
# Make Y matrix
sample.size = nrow(df.all)
Y = matrix( c(rep(0, sample.size),     # = tl.1
              df.all$ClaimAmount,      # = yl.1
              df.all$ClaimAmount,      # = tu.1
              rep(Inf, sample.size)    # = tu.1
              ),
           ncol = 4, byrow = FALSE)

# Make X matrix
X.continuous = cbind(df.all$CarAge, df.all$DriverAge)
X.power = model.matrix(~df.all$Power, data = df.all)      # Default Power  is 'd'
X.brand = model.matrix(~df.all$Brand, data = df.all)      # Default Brand  is 'Fiat'
X.gas = model.matrix(~df.all$Gas, data = df.all)          # Default Gas    is 'Diesel'
X.region = model.matrix(~df.all$Region, data = df.all)    # Default Region is 'Aquitaine'
X = matrix(cbind(rep(1, sample.size), # Intercept
                 X.continuous, X.power[,-1], X.brand[,-1], X.gas[,-1], X.region[,-1]),
           nrow = sample.size, byrow = FALSE)

colnames(X) = c("Intercept", "CarAge", "DriverAge",
                "Powere", "Powerf", "Powerg", "Powerh", "Poweri", "Powerj",
                "Powerk", "Powerl", "Powerm", "Powern", "Powero",
                "BrandJK", "BrandMCB", "BrandOGF", "BrandOther", "BrandRNC", "BrandVAS",
                "GasRegular",
                "RegionBN", "RegionB", "RegionC", "RegionHN", "RegionIF",
                "RegionL", "RegionNPC", "RegionPL", "RegionPC")
```

We will save the data for the fitting procedures later.

```
save(X, file = "X.Rda")
save(Y, file = "Y.Rda")
```

Parameter Initialization

The `LRMoE` fitting function also requires initialization of `n.comp` (number of latent components), `comp.dist` (component distributions by dimension and by component), `zero.init` (zero inflation) and `params.init` (initialization of component distribution parameters).

Since component distributions included in `LRMoE` are all uni-modal, a good starting point is to observe the numbers of components is to count the number of peaks in the previous histogram of data. We will consider 3~6 latent components, each with 5 combinations of component distributions. For each case, we use k-means clustering and matching of moments to roughly choose initial parameters.

```
# Drop response: ClaimNb
df.all = df.all[,-which(names(df.all) %in% c("ClaimNb"))]
df.all.pos = df.all.pos[,-which(names(df.all.pos) %in% c("ClaimNb"))]
```

```
# Normalize data
df.all.norm = df.all
df.all.norm= df.all.norm[,-which(names(df.all.norm) %in% c("PolicyID", "ClaimNb"))]

df.all.norm$CarAge =
```

```r
  (df.all.norm$CarAge - mean(df.all.norm$CarAge))/sd(df.all.norm$CarAge)
df.all.norm$DriverAge =
  (df.all.norm$DriverAge - mean(df.all.norm$DriverAge))/sd(df.all.norm$DriverAge)
df.all.norm$ClaimAmount =
  (df.all.norm$ClaimAmount - mean(df.all.norm$ClaimAmount))/sd(df.all.norm$ClaimAmount)
```

```r
head(df.all.norm)
```

```
##   Power     CarAge   DriverAge                              Brand     Gas
## 1     g -1.3070259  0.04746775 Japanese (except Nissan) or Korean  Diesel
## 2     g -1.3070259  0.04746775 Japanese (except Nissan) or Korean  Diesel
## 3     f -0.9599851 -0.51087486 Japanese (except Nissan) or Korean Regular
## 4     f -0.9599851 -0.51087486 Japanese (except Nissan) or Korean Regular
## 5     g -1.3070259 -0.30149638 Japanese (except Nissan) or Korean  Diesel
## 6     g -1.3070259 -0.30149638 Japanese (except Nissan) or Korean  Diesel
##              Region ClaimAmount
## 1         Aquitaine -0.01989646
## 2         Aquitaine -0.01989646
## 3 Nord-Pas-de-Calais -0.01989646
## 4 Nord-Pas-de-Calais -0.01989646
## 5   Pays-de-la-Loire -0.01989646
## 6   Pays-de-la-Loire -0.01989646
```

```r
summary(df.all.norm)
```

```
##      Power           CarAge           DriverAge
##  f      :95718   Min.   :-1.30703   Min.   :-1.90673
##  g      :91198   1st Qu.:-0.78646   1st Qu.:-0.79005
##  e      :77022   Median :-0.09238   Median :-0.09212
##  d      :68014   Mean   : 0.00000   Mean   : 0.00000
##  h      :26698   3rd Qu.: 0.77522   3rd Qu.: 0.60581
##  j      :18038   Max.   :16.04501   Max.   : 3.74649
##  (Other):36481
##                                 Brand            Gas
##  Fiat                              : 16723   Diesel :205945
##  Japanese (except Nissan) or Korean: 79060   Regular:207224
##  Mercedes, Chrysler or BMW         : 19280
##  Opel, General Motors or Ford      : 37402
##  other                             :  9866
##  Renault, Nissan or Citroen        :218200
##  Volkswagen, Audi, Skoda or Seat   : 32638
##                Region        ClaimAmount
##  Centre            :160601   Min.   : -0.0199
##  Ile-de-France     : 69791   1st Qu.: -0.0199
##  Bretagne          : 42122   Median : -0.0199
##  Pays-de-la-Loire  : 38751   Mean   :  0.0000
##  Aquitaine         : 31329   3rd Qu.: -0.0199
##  Nord-Pas-de-Calais: 27285   Max.   :485.8049
##  (Other)           : 43290
```

The `LRMoE` package contains a Clustered Method of Moments initialization function which is used in conjunction of `kmeans`. We look at the 3-component case in detail and skip the rest.

3 Latent Components

```
set.seed(7777) # For reproducible results

dim.m = 1
n.comp = 3

norm.km.cluster.3 = kmeans(data.matrix(df.all.norm), n.comp)
norm.init.analysis.3 = cluster.mm.severity(df.all$ClaimAmount, norm.km.cluster.3$cluster)
```

The returned list `norm.init.analysis.3` contains cluster proportion (of the entire population), zero inflation, summary statsitics and parameter initializations for all selection of component distributions. The user can then choose which distributions to use. As a general rule of thumb, initialization with very extreme parameter values should be avoided.

For example, the initialization of the first component is as follows.

```
norm.init.analysis.3[[1]]
```

```
## $cluster.prop
## [1] 0.207397
##
## $zero.prop
## [1] 0.9625394
##
## $mean.pos
## [1] 1736.804
##
## $var.pos
## [1] 14778491
##
## $cv.pos
## [1] 2.213422
##
## $skew.pos
## [1] 10.50369
##
## $kurt.pos
## [1] 143.7694
##
## $gamma.init
##        shape        scale
##    0.2041134 8509.0148004
##
## $lnorm.init
##  meanlog    sdlog
## 6.572390 1.332225
##
## $invgauss.init
##     mean    scale
## 1736.804  354.505
##
## $weibull.init
##    shape    scale
##    3.000 2605.206
```

```
##
## $burr.init
##   shape1   shape2    scale
##     2.00     5.00 12770.62
```

4~6 Latent Components

```
set.seed(7777) # For reproducible results

dim.m = 1
n.comp = 4

norm.km.cluster.4 = kmeans(data.matrix(df.all.norm), n.comp)
norm.init.analysis.4 = cluster.mm.severity(df.all$ClaimAmount, norm.km.cluster.4$cluster)
```

```
set.seed(7777) # For reproducible results

dim.m = 1
n.comp = 5

norm.km.cluster.5 = kmeans(data.matrix(df.all.norm), n.comp)
norm.init.analysis.5 = cluster.mm.severity(df.all$ClaimAmount, norm.km.cluster.5$cluster)
```

```
set.seed(7777) # For reproducible results

dim.m = 1
n.comp = 6

norm.km.cluster.6 = kmeans(data.matrix(df.all.norm), n.comp)
norm.init.analysis.6 = cluster.mm.severity(df.all$ClaimAmount, norm.km.cluster.6$cluster)
```

We will save all initilizations for use in the next step.

```
save(norm.init.analysis.3, file = "RealDataInit3.Rda")
save(norm.init.analysis.4, file = "RealDataInit4.Rda")
save(norm.init.analysis.5, file = "RealDataInit5.Rda")
save(norm.init.analysis.6, file = "RealDataInit6.Rda")
```