final project

jian & kelly

2024-05-05

Jian Jo: part 1 & part 2

Kelly Lee: part 3 & part 4

Part 1: Introduction

This project focuses on the analysis of factors contributing to the cost of treatment of patients and creating predictive models for the charges by the health insurance provider.

This insurance charge data is obtained from the website Kaggle. This data set contains 1338 observations of 7 variables. The variables include:

1. Age: the age of the primary beneficiary under consideration
2. Sex: gender of the insurance contractor — categorized as female or male
3. BMI: Body Mass Index is a measure of body weight relative to height, indicating whether weight is comparatively high or low. It's the ratio of weight (in kilograms) to height (in meters) squared. (ideally, BMI values range between 18.5 and 24.9)
4. Children: the number of children covered by health insurance or the count of dependents
5. Smoker: whether the individual smokes or not
6. Region: the residential area of the beneficiary within the United States, with options such as northeast, southeast, southwest, and northwest
7. Charges: the individual medical costs billed by the health insurance provider

For modeling purposes, the dataset will be split into training and testing sets. All analytical procedures will be conducted using R. This report will comprise total of 4 sections, beginning with this introduction section. The part 2 will focus on Exploratory Data Analysis, while the part 3 will outline the methodology with predictive models. The final section, the part 4, will talk about the final summary and conclusions.

Part 2: Exploratory Data Analysis

```r
# import all necessary packages
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(readr)
library(ggplot2)
library(faraway)
library(lattice)
```

```
##
## Attaching package: 'lattice'

## The following object is masked from 'package:faraway':
##
##     melanoma
```

```r
library(caret)
library(knitr)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following objects are masked from 'package:faraway':
##
##     logit, vif
```

```r
df <- read.csv("insurance.csv")
head(df)
```

```
##   age    sex    bmi children smoker    region   charges
## 1  19 female 27.900        0    yes southwest 16884.924
## 2  18   male 33.770        1     no southeast  1725.552
## 3  28   male 33.000        3     no southeast  4449.462
## 4  33   male 22.705        0     no northwest 21984.471
## 5  32   male 28.880        0     no northwest  3866.855
## 6  31 female 25.740        0     no southeast  3756.622
```

```r
names(df)
```

```
## [1] "age"      "sex"      "bmi"      "children" "smoker"   "region"   "charges"
```

```r
str(df)
```

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : chr  "female" "male" "male" "male" ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : chr  "yes" "no" "no" "no" ...
##  $ region  : chr  "southwest" "southeast" "southeast" "northwest" ...
##  $ charges : num  16885 1726 4449 21984 3867 ...
```

From this table above, we got the basic idea of each variable. There are 1338 rows (observations) along with 7 columns (variables). There are 3 categorical variables—sex, smoker, and region—and 4 numerical variables such as age, bmi, children, and charges.Our response variable will be **charges**. Also, each data type looks valid—fitting to what they are supposed to be.

```r
df$sex <- factor(df$sex)
df$smoker <- factor(df$smoker)
df$region <- factor(df$region)
```

Turn categorical variables as factor to fit a model later on.

```r
# check for null values
colSums(is.na(df))
```

```
##      age      sex      bmi children   smoker   region  charges
##        0        0        0        0        0        0        0
```

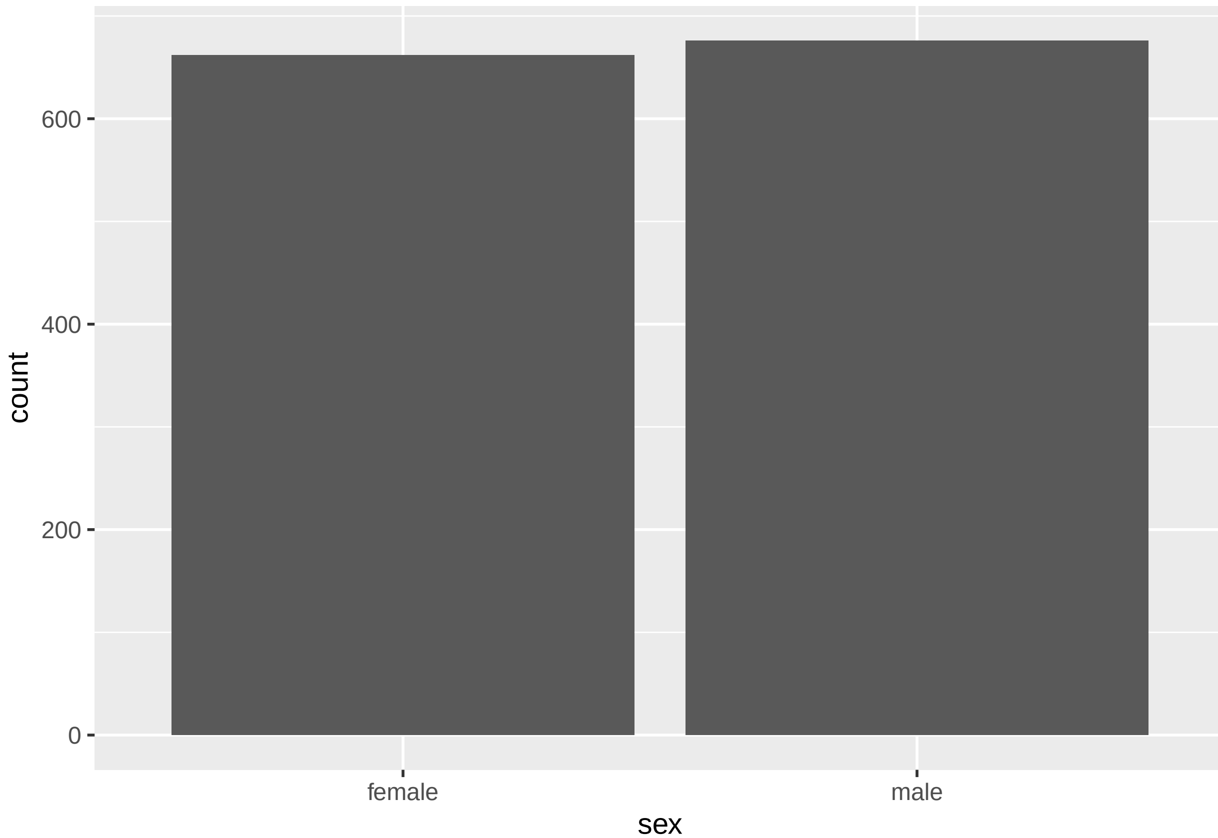
There is no null values in this dataset.

Plotting - individual variable

```r
# visaulize distributions
# categorical
print(table(df['sex']))
```

```
## sex
## female   male
##    662    676
```

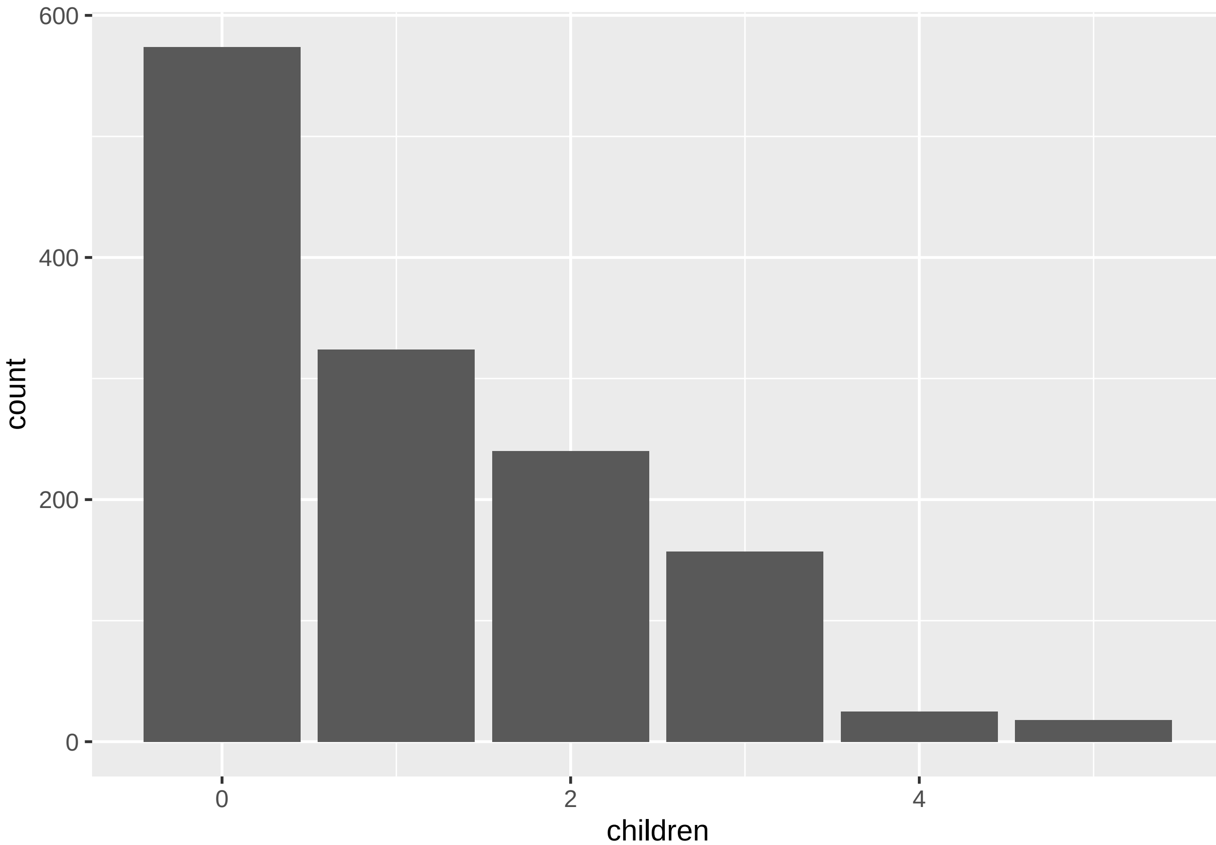
```r
ggplot(data = df) + theme(plot.title = element_text(hjust = 0.57)) + geom_bar(mapping = aes(x = sex))
```




```r
# children
print(table(df['children']))
```

```
## children
##   0   1   2   3   4   5
## 574 324 240 157  25  18
```

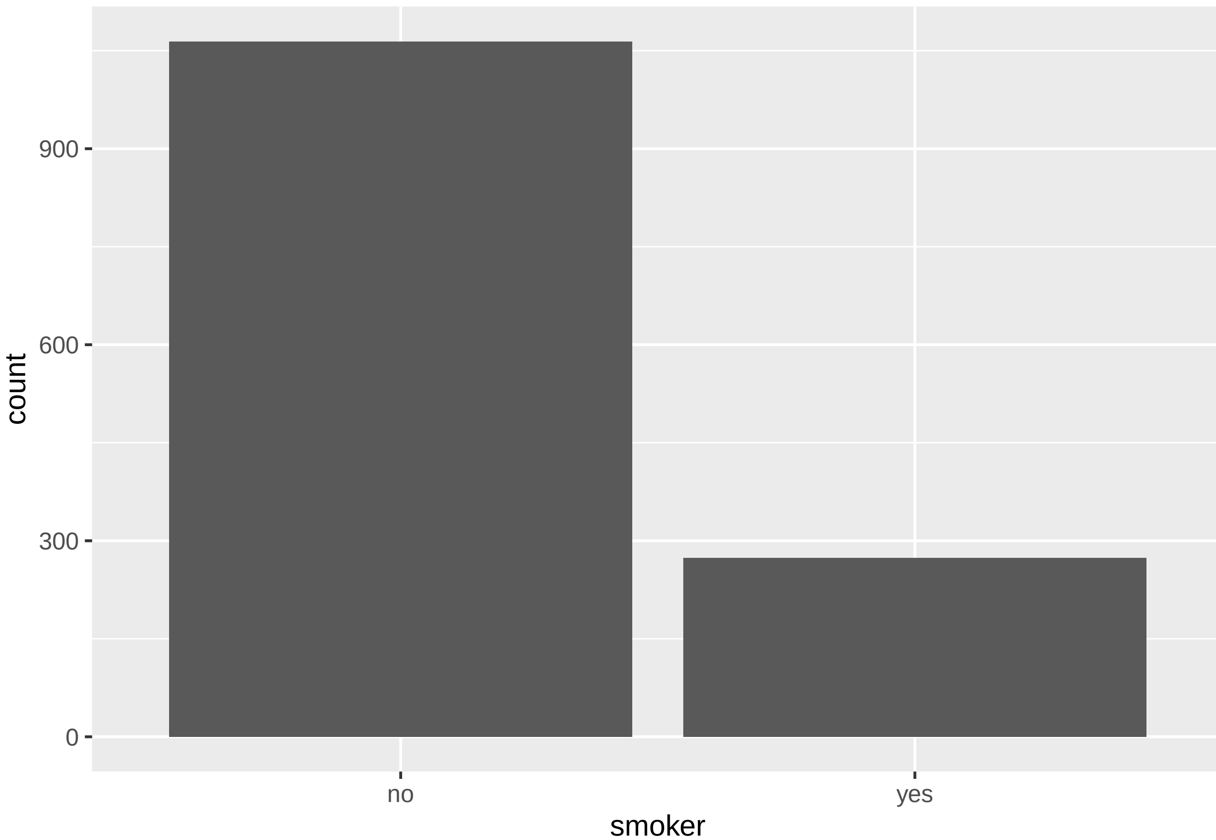
```r
ggplot(data = df) +
  geom_bar(mapping = aes(x = children))
```




```r
# smoker
print(table(df['smoker']))
```

```
## smoker
##   no  yes
## 1064  274
```

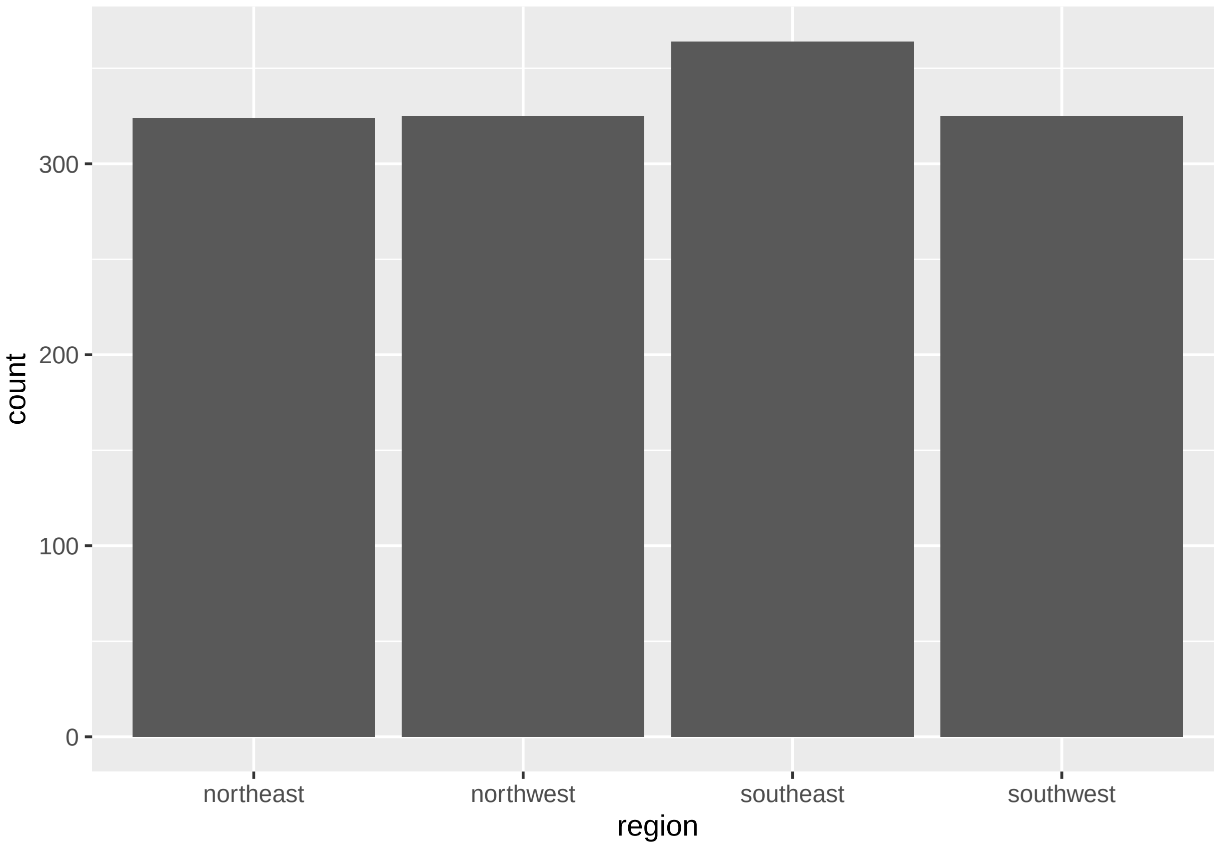
```r
ggplot(data = df) +
  geom_bar(mapping = aes(x = smoker))
```




```r
# region
print(table(df['region']))
```

```
## region
## northeast northwest southeast southwest
##       324       325       364       325
```

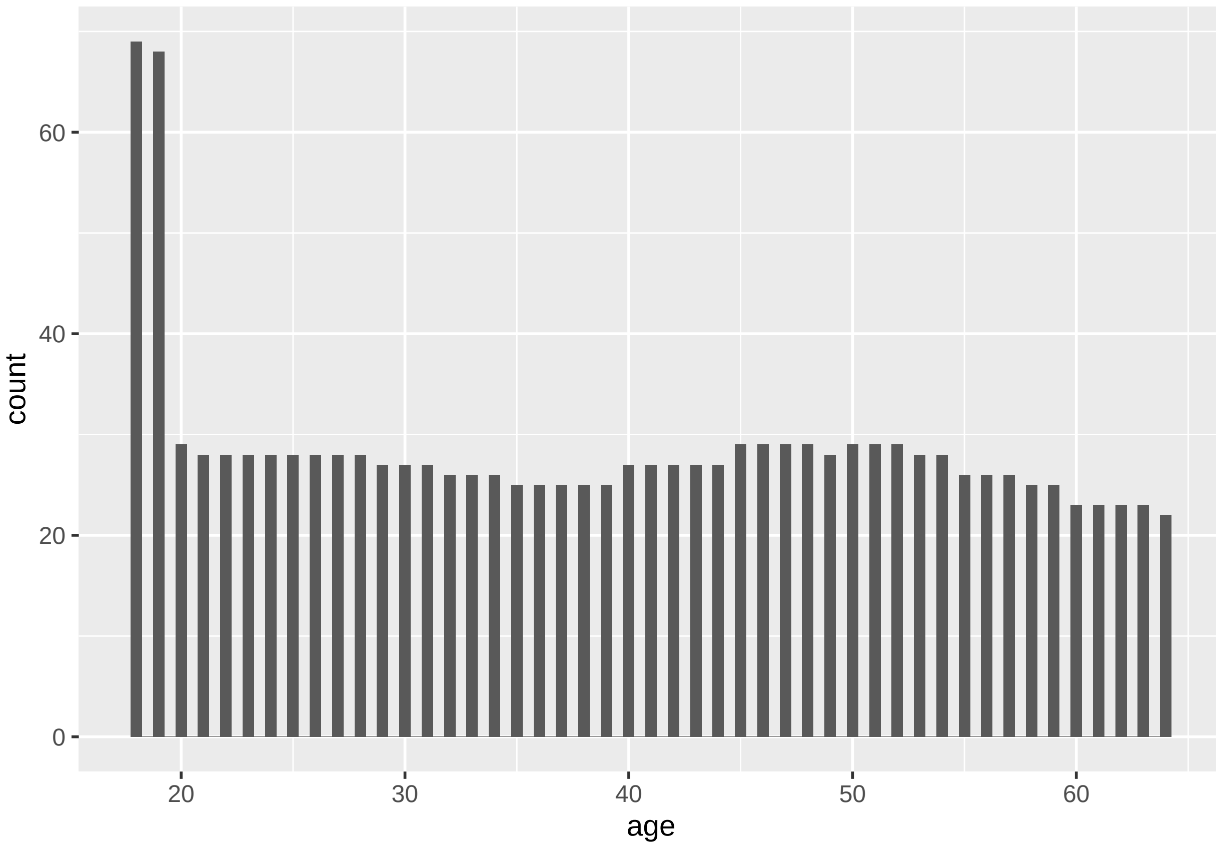
```r
ggplot(data = df) +
  geom_bar(mapping = aes(x = region))
```



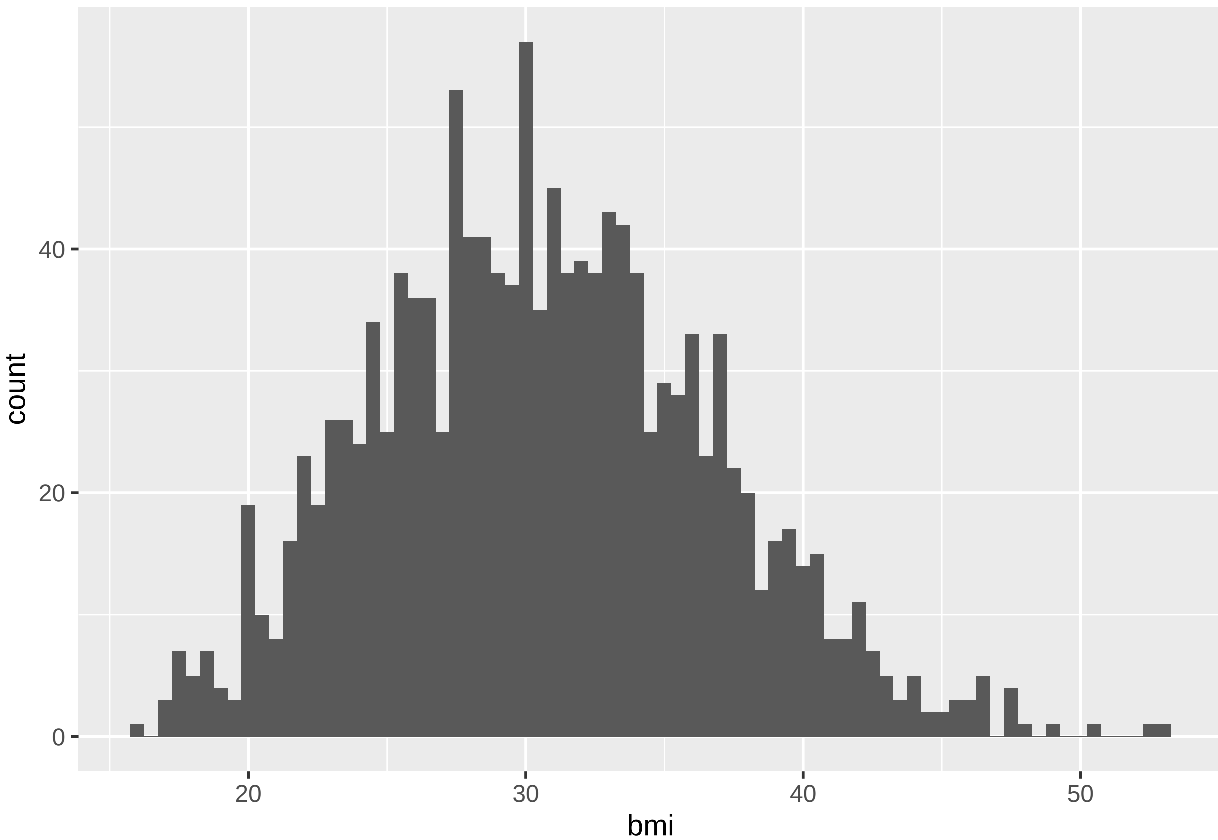

```r
# visualize distributions for continuous variables
print(table(df['age']))
```

```
## age
## 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
## 69 68 29 28 28 28 28 28 28 28 28 27 27 27 26 26 26 25 25 25 25 25 27 27 27 27
## 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
## 27 29 29 29 29 28 29 29 29 28 28 26 26 26 25 25 23 23 23 23 22
```

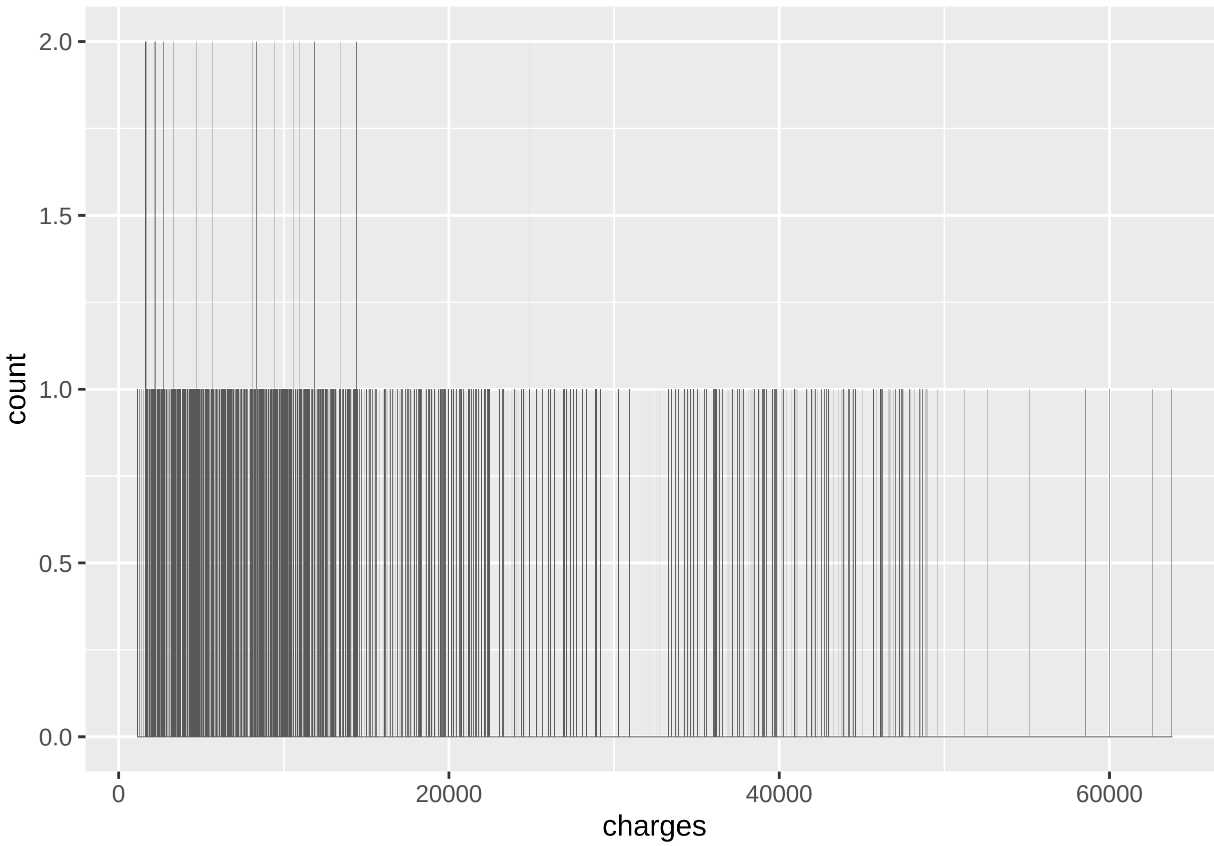
```r
ggplot(data = df) +
  geom_histogram(mapping = aes(x = age), binwidth = 0.5)
```




```r
# bmi
ggplot(data = df) +
  geom_histogram(mapping = aes(x = bmi), binwidth = 0.5)
```




```r
# charges
ggplot(data = df) +
  geom_histogram(mapping = aes(x = charges), binwidth = 0.5)
```



The histogram of the “bmi” is the only one looking like it has a bell-curve but left-skewed.

```r
lapply(df, unique)
```

```
## $age
##  [1] 19 18 28 33 32 31 46 37 60 25 62 23 56 27 52 30 34 59 63 55 22 26 35 24 41
## [26] 38 36 21 48 40 58 53 43 64 20 61 44 57 29 45 54 49 47 51 42 50 39
##
## $sex
## [1] female male
## Levels: female male
##
## $bmi
##   [1] 27.900 33.770 33.000 22.705 28.880 25.740 33.440 27.740 29.830 25.840
##  [11] 26.220 26.290 34.400 39.820 42.130 24.600 30.780 23.845 40.300 35.300
##  [21] 36.005 32.400 34.100 31.920 28.025 27.720 23.085 32.775 17.385 36.300
##  [31] 35.600 26.315 28.600 28.310 36.400 20.425 32.965 20.800 36.670 39.900
##  [41] 26.600 36.630 21.780 30.800 37.050 37.300 38.665 34.770 24.530 35.200
##  [51] 35.625 33.630 28.000 34.430 28.690 36.955 31.825 31.680 22.880 37.335
##  [61] 27.360 33.660 24.700 25.935 22.420 28.900 39.100 36.190 23.980 24.750
##  [71] 28.500 28.100 32.010 27.400 34.010 29.590 35.530 39.805 26.885 38.285
##  [81] 37.620 41.230 34.800 22.895 31.160 27.200 26.980 39.490 24.795 31.300
##  [91] 38.280 19.950 19.300 31.600 25.460 30.115 29.920 27.500 28.400 30.875
## [101] 27.940 35.090 29.700 35.720 32.205 28.595 49.060 27.170 23.370 37.100
## [111] 23.750 28.975 31.350 33.915 28.785 28.300 37.400 17.765 34.700 26.505
## [121] 22.040 35.900 25.555 28.050 25.175 31.900 36.000 32.490 25.300 29.735
## [131] 38.830 30.495 37.730 37.430 24.130 37.145 39.520 24.420 27.830 36.850
## [141] 39.600 29.800 29.640 28.215 37.000 33.155 18.905 41.470 30.300 15.960
```

```
## [151] 33.345 37.700 27.835 29.200 26.410 30.690 41.895 30.900 32.200 32.110
## [161] 31.570 26.200 30.590 32.800 18.050 39.330 32.230 24.035 36.080 22.300
## [171] 26.400 31.800 26.730 23.100 23.210 33.700 33.250 24.640 33.880 38.060
## [181] 41.910 31.635 36.195 17.800 24.510 22.220 38.390 29.070 22.135 26.800
## [191] 30.020 35.860 20.900 17.290 34.210 25.365 40.150 24.415 25.200 26.840
## [201] 24.320 42.350 19.800 32.395 30.200 29.370 34.200 27.455 27.550 20.615
## [211] 24.300 31.790 21.560 28.120 40.565 27.645 31.200 26.620 48.070 36.765
## [221] 33.400 45.540 28.820 22.990 27.700 25.410 34.390 22.610 37.510 38.000
## [231] 33.330 34.865 33.060 35.970 31.400 25.270 40.945 34.105 36.480 33.800
## [241] 36.700 36.385 34.500 32.300 27.600 29.260 35.750 23.180 25.600 35.245
## [251] 43.890 20.790 30.500 21.700 21.890 24.985 32.015 30.400 21.090 22.230
## [261] 32.900 24.890 31.460 17.955 30.685 43.340 39.050 30.210 31.445 19.855
## [271] 31.020 38.170 20.600 47.520 20.400 38.380 24.310 23.600 21.120 30.030
## [281] 17.480 20.235 17.195 23.900 35.150 35.640 22.600 39.160 27.265 29.165
## [291] 16.815 33.100 26.900 33.110 31.730 46.750 29.450 32.680 33.500 43.010
## [301] 36.520 26.695 25.650 29.600 38.600 23.400 46.530 30.140 30.000 38.095
## [311] 28.380 28.700 33.820 24.090 32.670 25.100 32.560 41.325 39.500 34.300
## [321] 31.065 21.470 25.080 43.400 25.700 27.930 39.200 26.030 30.250 28.930
## [331] 35.700 35.310 31.000 44.220 26.070 25.800 39.425 40.480 38.900 47.410
## [341] 35.435 46.700 46.200 21.400 23.800 44.770 32.120 29.100 37.290 43.120
## [351] 36.860 34.295 23.465 45.430 23.650 20.700 28.270 35.910 29.000 19.570
## [361] 31.130 21.850 40.260 33.725 29.480 32.600 37.525 23.655 37.800 19.000
## [371] 21.300 33.535 42.460 38.950 36.100 29.300 39.700 38.190 42.400 34.960
## [381] 42.680 31.540 29.810 21.375 40.810 17.400 20.300 18.500 26.125 41.690
## [391] 24.100 36.200 40.185 39.270 34.870 44.745 29.545 23.540 40.470 40.660
## [401] 36.600 35.400 27.075 28.405 21.755 40.280 30.100 32.100 23.700 35.500
## [411] 29.150 27.000 37.905 22.770 22.800 34.580 27.100 19.475 26.700 34.320
## [421] 24.400 41.140 22.515 41.800 26.180 42.240 26.510 35.815 41.420 36.575
## [431] 42.940 21.010 24.225 17.670 31.500 31.100 32.780 32.450 50.380 47.600
## [441] 25.400 29.900 43.700 24.860 28.800 29.500 29.040 38.940 44.000 20.045
## [451] 40.920 35.100 29.355 32.585 32.340 39.800 24.605 33.990 28.200 25.000
## [461] 33.200 23.200 20.100 32.500 37.180 46.090 39.930 35.800 31.255 18.335
## [471] 42.900 26.790 39.615 25.900 25.745 28.160 23.560 40.500 35.420 39.995
## [481] 34.675 20.520 23.275 36.290 32.700 19.190 20.130 23.320 45.320 34.600
## [491] 18.715 21.565 23.000 37.070 52.580 42.655 21.660 32.000 18.300 47.740
## [501] 22.100 19.095 31.240 29.925 20.350 25.850 42.750 18.600 23.870 45.900
## [511] 21.500 30.305 44.880 41.100 40.370 28.490 33.550 40.375 27.280 17.860
## [521] 33.300 39.140 21.945 24.970 23.940 34.485 21.800 23.300 36.960 21.280
## [531] 29.400 27.300 37.900 37.715 23.760 25.520 27.610 27.060 39.400 34.900
## [541] 22.000 30.360 27.800 53.130 39.710 32.870 44.700 30.970
##
## $children
## [1] 0 1 3 2 5 4
##
## $smoker
## [1] yes no
## Levels: no yes
##
## $region
## [1] southwest southeast northwest northeast
## Levels: northeast northwest southeast southwest
##
## $charges
##    [1] 16884.924  1725.552  4449.462 21984.471  3866.855  3756.622  8240.590
```

```
##    [8]  7281.506  6406.411 28923.137  2721.321 27808.725  1826.843 11090.718
##   [15] 39611.758  1837.237 10797.336  2395.172 10602.385 36837.467 13228.847
##   [22]  4149.736  1137.011 37701.877  6203.902 14001.134 14451.835 12268.632
##   [29]  2775.192 38711.000 35585.576  2198.190  4687.797 13770.098 51194.559
##   [36]  1625.434 15612.193  2302.300 39774.276 48173.361  3046.062  4949.759
##   [43]  6272.477  6313.759  6079.672 20630.284  3393.356  3556.922 12629.897
##   [50] 38709.176  2211.131  3579.829 23568.272 37742.576  8059.679 47496.494
##   [57] 13607.369 34303.167 23244.790  5989.524  8606.217  4504.662 30166.618
##   [64]  4133.642 14711.744  1743.214 14235.072  6389.378  5920.104 17663.144
##   [71] 16577.780  6799.458 11741.726 11946.626  7726.854 11356.661  3947.413
##   [78]  1532.470  2755.021  6571.024  4441.213  7935.291 37165.164 11033.662
##   [85] 39836.519 21098.554 43578.939 11073.176  8026.667 11082.577  2026.974
##   [92] 10942.132 30184.937  5729.005 47291.055  3766.884 12105.320 10226.284
##   [99] 22412.648 15820.699  6186.127  3645.089 21344.847 30942.192  5003.853
##  [106] 17560.380  2331.519  3877.304  2867.120 47055.532 10825.254 11881.358
##  [113]  4646.759  2404.734 11488.317 30259.996 11381.325 19107.780  8601.329
##  [120]  6686.431  7740.337  1705.624  2257.475 39556.495 10115.009  3385.399
##  [127] 17081.080  9634.538 32734.186  6082.405 12815.445 13616.359 11163.568
##  [134]  1632.564  2457.211  2155.682  1261.442  2045.685 27322.734  2166.732
##  [141] 27375.905  3490.549 18972.495 18157.876 20745.989  5138.257 40720.551
##  [148]  9877.608 10959.695  1842.519  5125.216  7789.635  6334.344 19964.746
##  [155]  7077.189  6948.701 21223.676 15518.180 36950.257 19749.383 21348.706
##  [162] 36149.484 10450.552  5152.134  5028.147 10407.086  4830.630  6128.797
##  [169]  2719.280  4827.905 13405.390  8116.680  1694.796  5246.047  2855.438
##  [176] 48824.450  6455.863 10436.096  8823.279  8538.288 11735.879  1631.821
##  [183]  4005.423  7419.478  7731.427 43753.337  3981.977  5325.651  6775.961
##  [190]  4922.916 12557.605  4883.866  2137.654 12044.342  1137.470  1639.563
##  [197]  5649.715  8516.829  9644.253 14901.517  2130.676  8871.152 13012.209
##  [204] 37133.898  7147.105  4337.735 11743.299 20984.094 13880.949  6610.110
##  [211]  1980.070  8162.716  3537.703  5002.783  8520.026  7371.772 10355.641
##  [218]  2483.736  3392.977 25081.768  5012.471 10564.885  5253.524 34779.615
##  [225] 19515.542 11987.168  2689.495 24227.337  7358.176  9225.256  7443.643
##  [232] 14001.287  1727.785 12333.828  6710.192 19444.266  1615.767  4463.205
##  [239] 17352.680  7152.671 38511.628  5354.075 35160.135  7196.867 29523.166
##  [246] 24476.479 12648.703  1986.933  1832.094  4040.558 12829.455 47305.305
##  [253] 44260.750  4260.744 41097.162 13047.332 43921.184  5400.980 11520.100
##  [260] 33750.292 11837.160 17085.268 24869.837 36219.405 20462.998 46151.124
##  [267] 17179.522 14590.632  7441.053  9282.481  1719.436 42856.838  7265.703
##  [274]  9617.662  2523.169  9715.841  2803.698  2150.469 12928.791  9855.131
##  [281] 22331.567 48549.178  4237.127 11879.104  9625.920  7742.110  9432.925
##  [288] 14256.193 47896.791 25992.821  3172.018 20277.808 42112.236  2156.752
##  [295]  3906.127  1704.568 16297.846 21978.677 38746.355  9249.495  6746.743
##  [302] 24873.385 12265.507  4349.462 12646.207 19442.354 20177.671  4151.029
##  [309] 11944.594  7749.156  8444.474  1737.376 42124.515  8124.408 34838.873
##  [316]  9722.770  8835.265 10435.065  7421.195  4667.608  4894.753 24671.663
##  [323] 35491.640 11566.301  2866.091  6600.206  3561.889 42760.502 47928.030
##  [330]  9144.565 48517.563 24393.622 13429.035 11658.379 19144.577 13822.803
##  [337] 12142.579 13937.666 41919.097  8232.639 18955.220 13352.100 13217.094
##  [344] 13981.850 10977.206  6184.299  4889.999  8334.458  5478.037  1635.734
##  [351] 11830.607  8932.084  3554.203 12404.879 14133.038 24603.048  8944.115
##  [358]  9620.331  1837.282  1607.510 10043.249  4751.070 13844.506  2597.779
##  [365]  3180.510  9778.347 13430.265  8017.061  8116.269  3481.868 13415.038
##  [372] 12029.287  7639.417 36085.219  1391.529 18033.968 21659.930 38126.247
##  [379] 16455.708 27000.985 15006.579 42303.692 20781.489  5846.918  8302.536
```

```
## [386]  1261.859 11856.412 30284.643  3176.816  4618.080 10736.871  2138.071
## [393]  8964.061  9290.139  9411.005  7526.706  8522.003 16586.498 14988.432
## [400]  1631.668  9264.797  8083.920 14692.669 10269.460  3260.199 11396.900
## [407]  4185.098  8539.671  6652.529  4074.454  1621.340 19594.810 14455.644
## [414]  5080.096  2134.901  7345.727  9140.951 18608.262 14418.280 28950.469
## [421] 46889.261 46599.108 39125.332  2727.395  8968.330  9788.866  6555.070
## [428]  7323.735  3167.456 18804.752 23082.955  4906.410  5969.723 12638.195
## [435]  4243.590 13919.823  2254.797  5926.846 12592.534  2897.323  4738.268
## [442] 37079.372  1149.396 28287.898 26109.329  7345.084 12731.000 11454.022
## [449]  5910.944  4762.329  7512.267  4032.241  1969.614  1769.532  4686.389
## [456] 21797.000 11881.970 11840.775 10601.412  7682.670 10381.479 22144.032
## [463] 15230.324 11165.418  1632.036 19521.968 13224.693 12643.378 23288.928
## [470]  2201.097  2497.038  2203.472  1744.465 20878.784 25382.297 28868.664
## [477] 35147.528  2534.394  1534.304  1824.285 15555.189  9304.702  1622.188
## [484]  9880.068  9563.029  4347.023 12475.351  1253.936 48885.136 10461.979
## [491]  1748.774 24513.091  2196.473 12574.049 17942.106  1967.023  4931.647
## [498]  8027.968  8211.100 13470.860 36197.699  6837.369 22218.115 32548.340
## [505]  5974.385  6796.863  2643.269  3077.095  3044.213 11455.280 11763.001
## [512]  2498.414  9361.327  1256.299 21082.160 11362.755 27724.289  8413.463
## [519]  5240.765  3857.759 25656.575  3994.178  9866.305  5397.617 38245.593
## [526] 11482.635 24059.680  9861.025  8342.909  1708.001 48675.518 14043.477
## [533] 12925.886 19214.706 13831.115  6067.127  5972.378  8825.086  8233.097
## [540] 27346.042  6196.448  3056.388 13887.204 63770.428 10231.500 23807.241
## [547]  3268.847 11538.421  3213.622 45863.205 13390.559  3972.925 12957.118
## [554] 11187.657 17878.901  3847.674  8334.590  3935.180 39983.426  1646.430
## [561]  9193.838 10923.933  2494.022  9058.730  2801.259  2128.431  6373.557
## [568]  7256.723 11552.904 45702.022  3761.292  2219.445  4753.637 31620.001
## [575] 13224.057 12222.898  1665.000 58571.074  9724.530  3206.491 12913.992
## [582]  6356.271 17626.240  1242.816  4779.602  3861.210 43943.876 13635.638
## [589]  5976.831 11842.442  8428.069  2566.471 15359.104  5709.164  8823.986
## [596]  7640.309  5594.846  7441.501 33471.972  1633.044  9174.136 11070.535
## [603] 16085.128 17468.984  9283.562  3558.620 25678.778  4435.094 39241.442
## [610]  8547.691  6571.544  2207.697  6753.038  1880.070 42969.853 11658.115
## [617] 23306.547 34439.856 10713.644  3659.346 40182.246  9182.170 34617.841
## [624] 12129.614  3736.465  6748.591 11326.715 11365.952 42983.459 10085.846
## [631]  1977.815  3366.670  7173.360  9391.346 14410.932  2709.112 24915.046
## [638] 20149.323 12949.155  6666.243 32787.459 13143.865  4466.621 18806.145
## [645] 10141.136  6123.569  8252.284  1712.227 12430.953  9800.888 10579.711
## [652]  8280.623  8527.532 12244.531 24667.419  3410.324  4058.712 26392.260
## [659] 14394.398  6435.624 22192.437  5148.553  1136.399 27037.914 42560.430
## [666]  8703.456 40003.332 45710.208  6500.236  4837.582  3943.595  4399.731
## [673]  6185.321 46200.985  7222.786 12485.801 46130.526 12363.547 10156.783
## [680]  2585.269  1242.260 40103.890  9863.472  4766.022 11244.377  7729.646
## [687]  5438.749 26236.580 34806.468  2104.113  8068.185  2362.229  2352.968
## [694]  3577.999  3201.245 29186.482 40273.645 10976.246  3500.612  2020.552
## [701]  9541.696  9504.310  5385.338  8930.935  5375.038 44400.406 10264.442
## [708]  6113.231  5469.007  1727.540 10107.221  8310.839  1984.453  2457.502
## [715] 12146.971  9566.991 13112.605 10848.134 12231.614  9875.680 11264.541
## [722] 12979.358  1263.249 10106.134 40932.429  6664.686 16657.717  2217.601
## [729]  6781.354 19361.999 10065.413  4234.927  9447.250 14007.222  9583.893
## [736] 40419.019  3484.331 36189.102 44585.456  8604.484 18246.496 43254.418
## [743]  3757.845  8827.210  9910.360 11737.849  1627.282  8556.907  3062.508
## [750] 19539.243  1906.358 14210.536 11833.782 17128.426  5031.270  7985.815
## [757] 23065.421  5428.728 36307.798  3925.758  2416.955 19040.876  3070.809
```

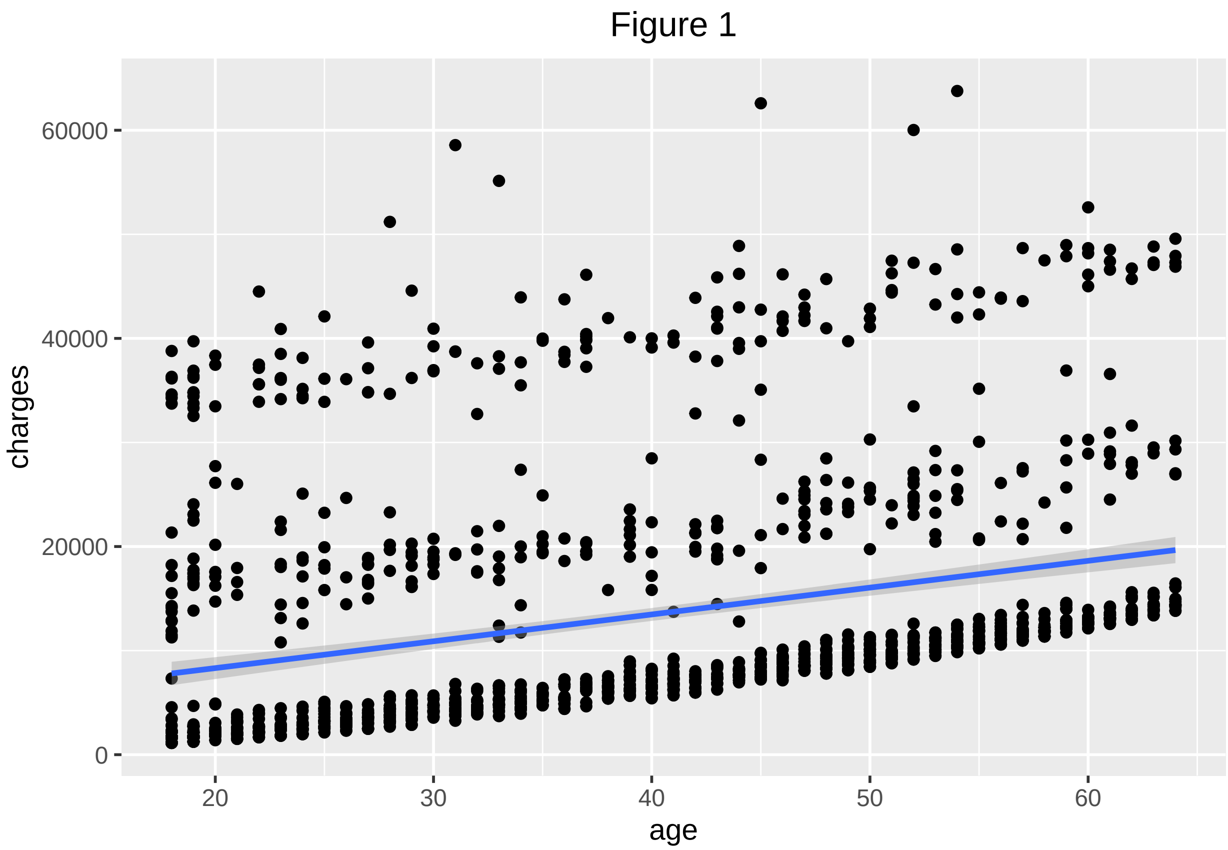
```
##  [764]  9095.068 11842.624  8062.764  7050.642 14319.031  6933.242 27941.288
##  [771] 11150.780 12797.210 17748.506  7261.741 10560.492  6986.697  7448.404
##  [778]  5934.380  9869.810 18259.216  1146.797  9386.161 24520.264  4350.514
##  [785]  6414.178 12741.167  1917.318  5209.579 13457.961  5662.225  1252.407
##  [792]  2731.912 21195.818  7209.492 18310.742  4266.166  4719.524 11848.141
##  [799] 17904.527  7046.722 14313.846  2103.080 38792.686  1815.876  7731.858
##  [806] 28476.735  2136.882  1131.507  3309.793  9414.920  6360.994 11013.712
##  [813]  4428.888  5584.306  1877.929  2842.761  3597.596 23401.306 55135.402
##  [820]  7445.918  2680.949  1621.883  8219.204 12523.605 16069.085 43813.866
##  [827] 20773.628 39597.407  6117.494 13393.756  5266.366  4719.737 11743.934
##  [834]  5377.458  7160.330  4402.233 11657.719  6402.291 12622.180  1526.312
##  [841] 12323.936 36021.011 27533.913 10072.055 45008.955  9872.701  2438.055
##  [848]  2974.126 10601.632 37270.151 14119.620 42111.665 11729.680 24106.913
##  [855]  1875.344 40974.165 15817.986 18218.161 10965.446 46113.511  7151.092
##  [862] 12269.689  5458.046  8782.469  6600.361  1141.445 11576.130 13129.603
##  [869]  4391.652  8457.818  3392.365  5966.887  6849.026  8891.139  2690.114
##  [876] 26140.360  6653.789  6282.235  6311.952  3443.064  2789.057  2585.851
##  [883] 46255.113  4877.981 19719.695 27218.437  5272.176  1682.597 11945.133
##  [890] 29330.983  7243.814 10422.917 44202.654 13555.005 13063.883 19798.055
##  [897]  2221.564  1634.573  2117.339  8688.859 48673.559  4661.286  8125.784
##  [904] 12644.589  4564.191  4846.920  7633.721 15170.069 17496.306  2639.043
##  [911] 33732.687 14382.709  7626.993  5257.508  2473.334 21774.322 35069.375
##  [918] 13041.921  5245.227 13451.122 13462.520  5488.262  4320.411  6250.435
##  [925] 25333.333  2913.569 12032.326 13470.804  6289.755  2927.065  6238.298
##  [932] 10096.970  7348.142  4673.392 12233.828 32108.663  8965.796  2304.002
##  [939]  9487.644  1121.874  9549.565  2217.469  1628.471 12982.875 11674.130
##  [946]  7160.094 39047.285  6358.776 19933.458 11534.873 47462.894  4527.183
##  [953] 38998.546 20009.634  3875.734 41999.520 12609.887 41034.221 28468.919
##  [960]  2730.108  3353.284 14474.675  9500.573 26467.097  4746.344 23967.383
##  [967]  7518.025  3279.869  8596.828 10702.642  4992.376  2527.819  1759.338
##  [974]  2322.622 16138.762  7804.160  2902.907  9704.668  4889.037 25517.114
##  [981]  4500.339 19199.944 16796.412  4915.060  7624.630  8410.047 28340.189
##  [988]  4518.826 14571.891  3378.910  7144.863 10118.424  5484.467 16420.495
##  [995]  7986.475  7418.522 13887.969  6551.750  5267.818 17361.766 34472.841
## [1002]  1972.950 21232.182  8627.541  4433.388  4438.263 24915.221 23241.475
## [1009]  9957.722  8269.044 18767.738 36580.282  8765.249  5383.536 12124.992
## [1016]  2709.244  3987.926 12495.291 26018.951  8798.593 35595.590 42211.138
## [1023]  1711.027  8569.862  2020.177 16450.895 21595.382  9850.432  6877.980
## [1030] 21677.283 44423.803  4137.523 13747.872 12950.071 12094.478 37484.449
## [1037] 39725.518  2250.835 22493.660 20234.855  1704.700 33475.817  3161.454
## [1044] 11394.066 21880.820  7325.048 44501.398  3594.171 39727.614  8023.135
## [1051] 14394.558  9288.027 25309.489  3353.470 10594.502  8277.523 17929.303
## [1058]  2480.979  4462.722  1981.582 11554.224 48970.248  6548.195  5708.867
## [1065]  7045.499  8978.185  5757.413 14349.854 10928.849 39871.704 13974.456
## [1072]  1909.527 12096.651 13204.286  4562.842  8551.347  2102.265 34672.147
## [1079] 15161.534 11884.049  4454.403  5855.903  4076.497 15019.760 19023.260
## [1086] 10796.350 11353.228  9748.911 10577.087 41676.081 11286.539  3591.480
## [1093] 33907.548 11299.343  4561.189 44641.197  1674.632 23045.566  3227.121
## [1100] 16776.304 11253.421  3471.410 11363.283 20420.605 10338.932  8988.159
## [1107] 10493.946  2904.088  8605.362 11512.405 41949.244 24180.933  5312.170
## [1114]  2396.096 10807.486  9222.403 36124.574 38282.749  5693.431 34166.273
## [1121]  8347.164 46661.442 18903.491 40904.200 14254.608 10214.636  5836.520
## [1128] 14358.364  1728.897  8582.302  3693.428 20709.020  9991.038 19673.336
## [1135] 11085.587  7623.518  3176.288  3704.354 36898.733  9048.027  7954.517
```

```
## [1142] 27117.994  6338.076  9630.397 11289.109 52590.829  2261.569 10791.960
## [1149]  5979.731  2203.736 12235.839 40941.285  5630.458 11015.175  7228.216
## [1156] 39722.746 14426.074  2459.720  3989.841  7727.253  5124.189 18963.172
## [1163]  2200.831  7153.554  5227.989 10982.501  4529.477  4670.640  6112.353
## [1170] 17178.682 22478.600 11093.623  6457.843  4433.916  2154.361 23887.663
## [1177]  6496.886  2899.489 19350.369  7650.774  2850.684  2632.992  9447.382
## [1184] 18328.238  8603.823 37465.344 13844.797 21771.342 13126.677  5327.400
## [1191] 13725.472 13019.161  8671.191  4134.082 18838.704 33307.551  5699.837
## [1198]  6393.603  4934.705  6198.752  8733.229  2055.325  9964.060 18223.451
## [1205]  5116.500 36910.608 38415.474 20296.863 12347.172  5373.364 23563.016
## [1212]  1702.455 10806.839  3956.071 12890.058  5415.661  4058.116 41661.602
## [1219]  7537.164  4718.204  6593.508  8442.667 26125.675  6858.480  4795.657
## [1226]  6640.545  7162.012 10594.226 11938.256 60021.399 20167.336 12479.709
## [1233] 11345.519  8515.759  2699.568 14449.854 12224.351  6985.507  3238.436
## [1240] 47269.854 49577.662  4296.271  3171.615  1135.941  5615.369  9101.798
## [1247]  6059.173  1633.962 37607.528 18648.422  1241.565 16232.847 15828.822
## [1254]  4415.159  6474.013 11436.738 11305.935 30063.581 10197.772  4544.235
## [1261]  3277.161  6770.193  7337.748 10370.913 26926.514 10704.470 34254.053
## [1268]  1880.487  8615.300  3292.530  3021.809 14478.330  4747.053 17043.341
## [1275] 10959.330  2741.948  4357.044 22462.044  4189.113  8283.681 24535.699
## [1282] 14283.459  1720.354 47403.880  8534.672  3732.625  5472.449 38344.566
## [1289]  7147.473  7133.903 34828.654  1515.345  9301.894 11931.125  1964.780
## [1296]  1708.926  4340.441  5261.469  2710.829 62592.873 46718.163  3208.787
## [1303] 37829.724 21259.378  2464.619 16115.305 21472.479 33900.653  6875.961
## [1310]  6940.910  4571.413  4536.259 36397.576 18765.875 11272.331  1731.677
## [1317]  1163.463 19496.719  7201.701  5425.023 28101.333 12981.346 43896.376
## [1324]  4239.893 13143.337  7050.021  9377.905 22395.744 10325.206 12629.166
## [1331] 10795.937 11411.685 10600.548  2205.981  1629.833  2007.945 29141.360
```

We confirmed that there's no null values by looking at the unique values from each variable. It's pretty safe to say there is no null value.

```r
# relationship between input variables and output variable, "charges"
# Box plots for numerical variables against charges
ggplot(df, aes(x=age, y=charges)) + geom_point() + geom_smooth(method="lm") + ggtitle("Figure 1") + ther
```

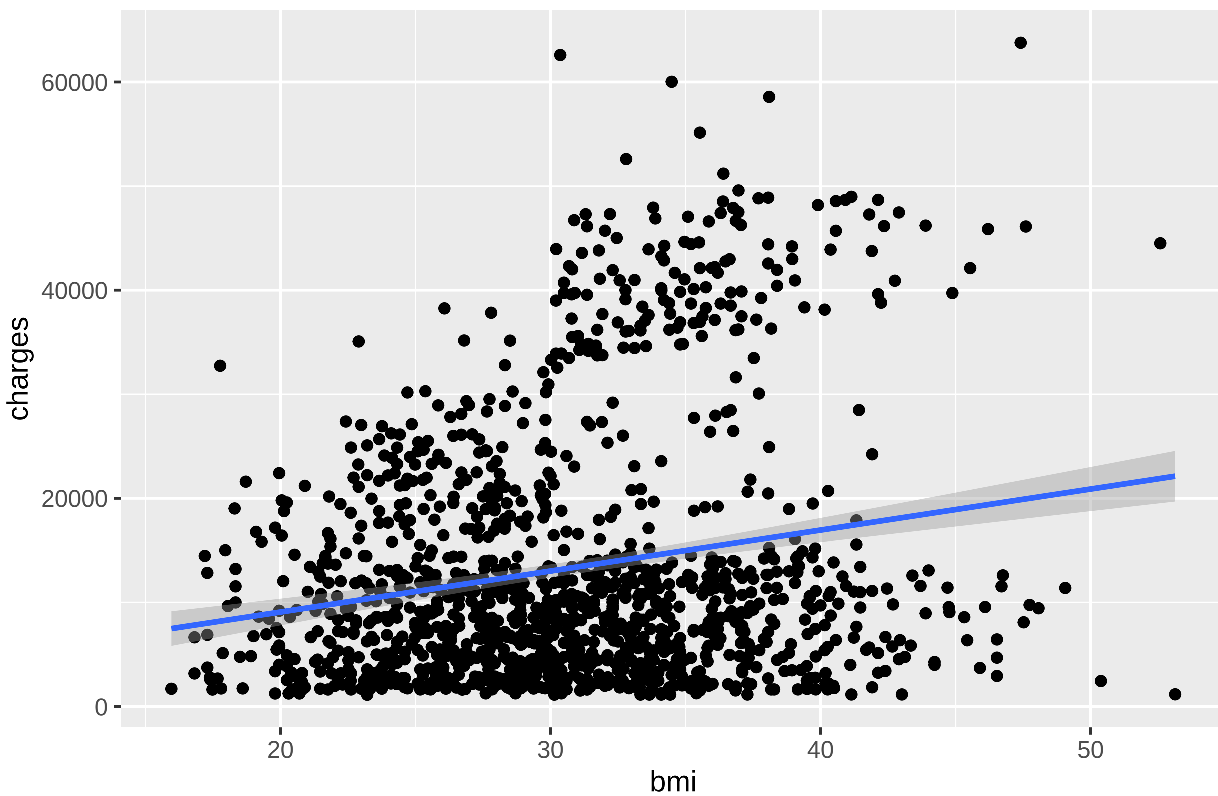
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(df, aes(x=bmi, y=charges)) + geom_point() + geom_smooth(method="lm") + ggtitle("Figure 2") + ther
```

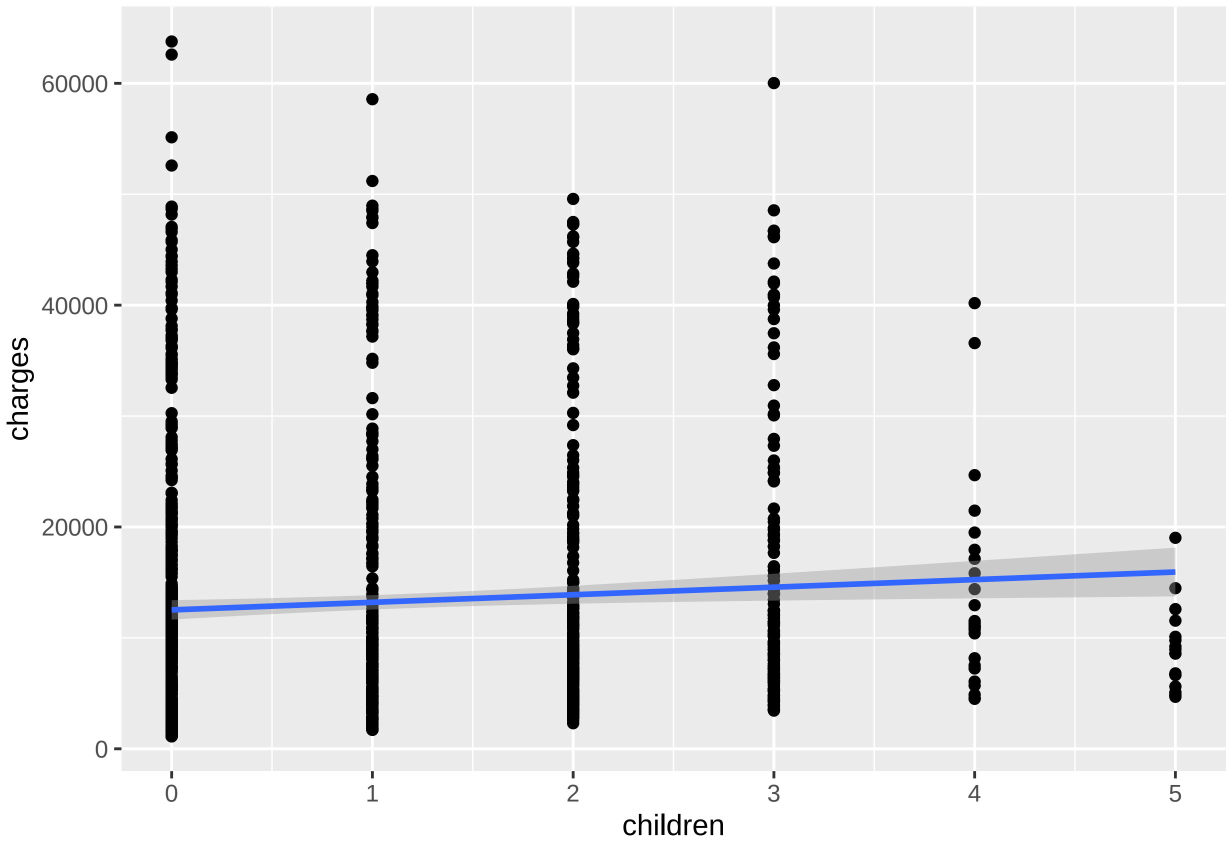
```
## `geom_smooth()` using formula = 'y ~ x'
```

Figure 2

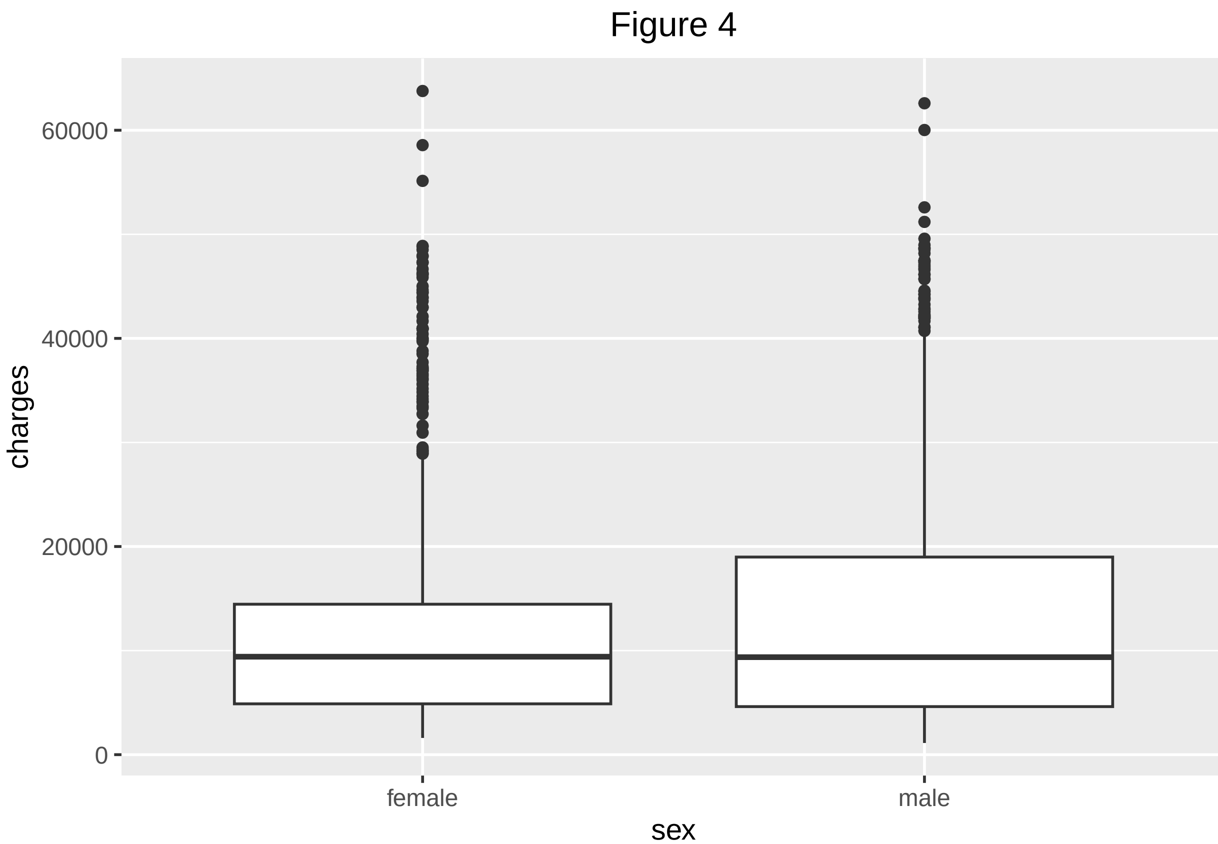



```r
ggplot(df, aes(x=children, y=charges)) + geom_point() + geom_smooth(method="lm")
```

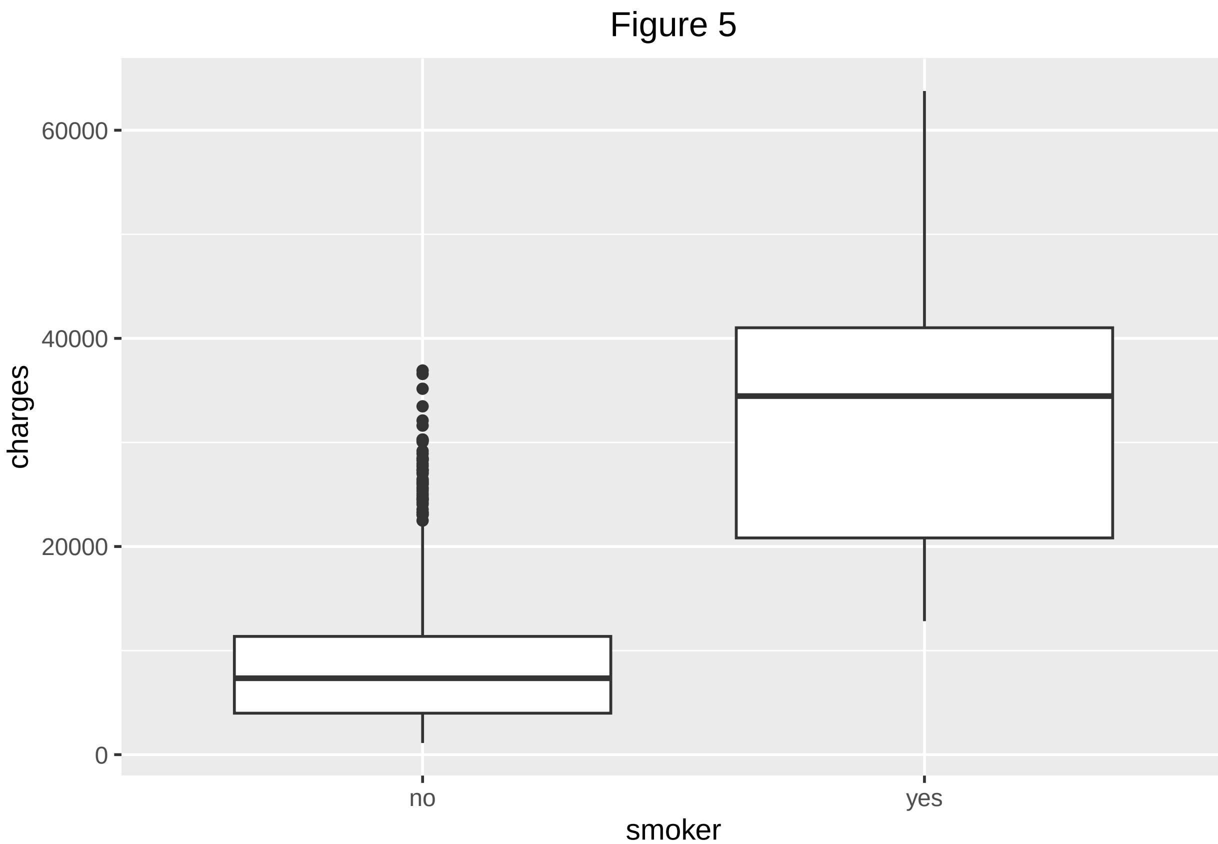
```
## `geom_smooth()` using formula = 'y ~ x'
```



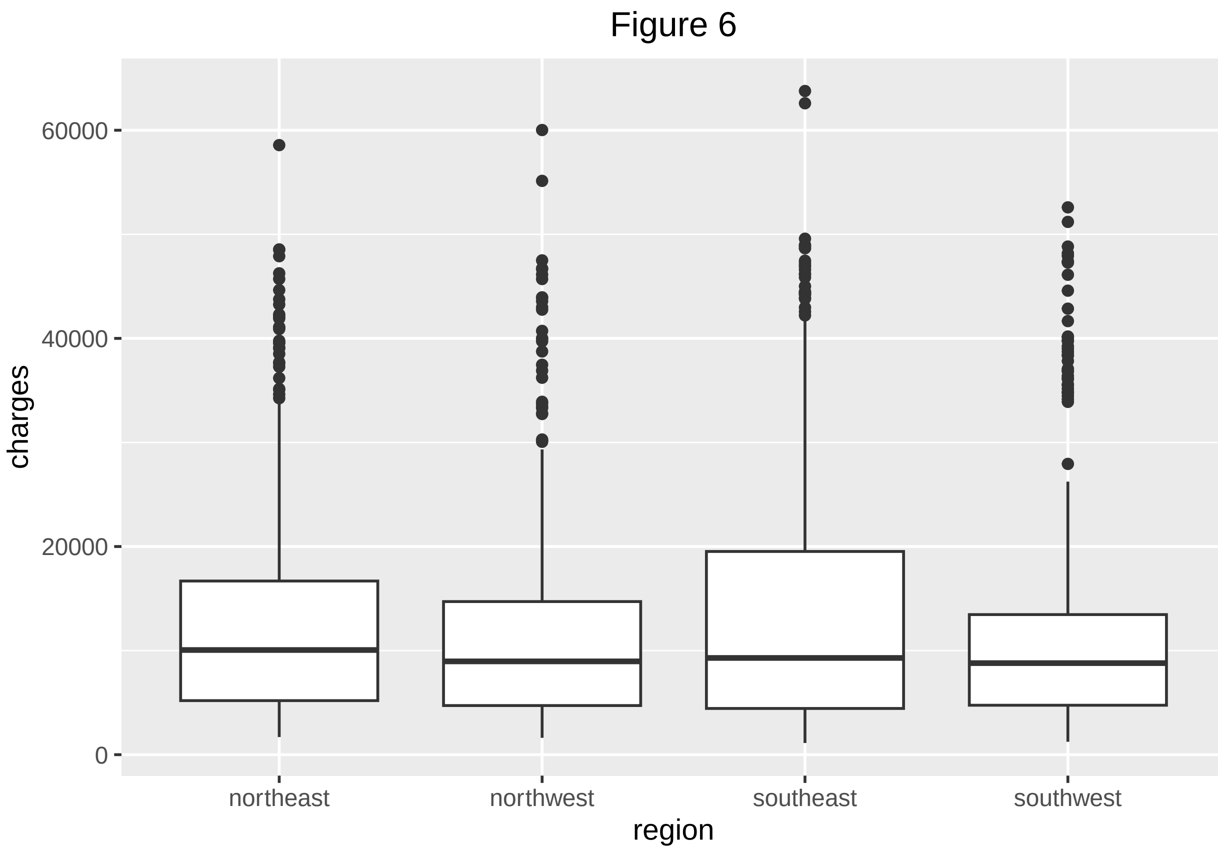

```
# Box plots for categorical variables against charges
ggplot(df, aes(x=sex, y=charges)) + geom_boxplot() + ggtitle("Figure 4") + theme(plot.title = element_te
```




```
ggplot(df, aes(x=smoker, y=charges)) + geom_boxplot() + ggtitle("Figure 5") + theme(plot.title = elemen
```




```
ggplot(df, aes(x=region, y=charges)) + geom_boxplot() + ggtitle("Figure 6") + theme(plot.title = elemen
```



From the scatter plot between age and charges, we can see that the charges increase as the ages increase, linear relationship. However, It seems like there are 3 different groups for charges which may have affected by the wealthiness.

The graph between BMI and charges shows a positive correlation as the trend line slope upwards. There's variability in charges by BMI, especially as BMI increases, which implies that while BMI is a factor in medical bill costs, other variables may influence the cost significantly. Also, there are notable outliers in higher BMI ranges where there's a huge gap difference among some individuals at similar BMI levels. We can consider BMI could be one of the risk factors when predicting insurance cost from this graph.

The scatterplot suggests that even though the variable "children" is numerical, it should be considered as categorical and converted using the factor function.

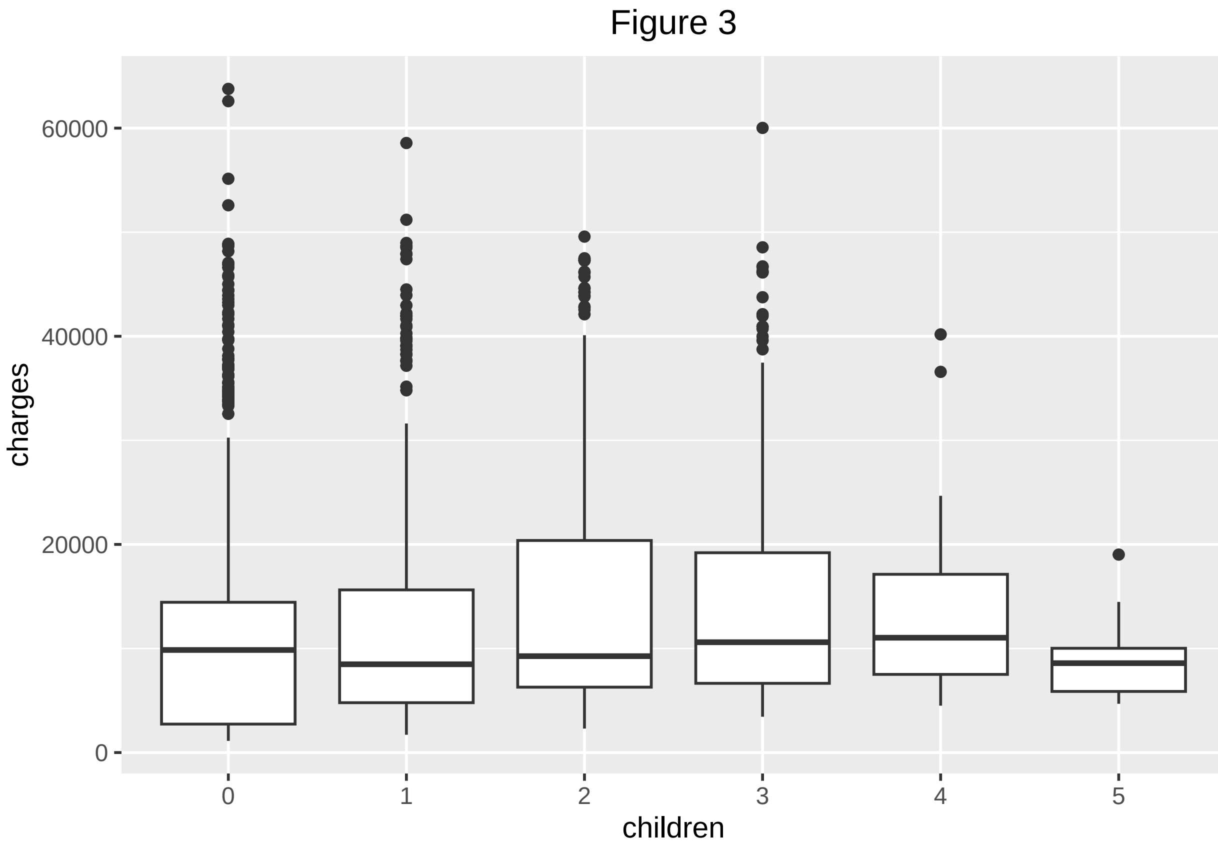
For the categorical variables, we applied boxplots to understand the relationship with the response variable.

From the graph of sex and charges, it's easy to notice that there are several outliers at high end of charges both in female and male. There's no significant differences in median of charges between males and females, but individual variability is high. Although the medians are similar, the presence of outliers in both groups indicates the factors other than sex might be more predictive of high charges.

The boxplot between smoker and charges shows a significant difference in median charges between smokers and non-smokers. Smokers incur far higher median insurance charges which suggests that smoking is one of the influential factors. The outliers in "no" group are more pronounced and numerous, indicating occasional high medical bills among non-smokers.

The boxplot of figure 6 shows the distribution of charges across different regions: northeast, northwest, southeast, and southwest. There is no huge difference across regions in terms of median charges but the spread and extremes (outliers) vary with the southeast showing higher outliers.

```
df$children <- factor(df$children)
ggplot(df, aes(x=children, y=charges)) + geom_boxplot() + ggtitle("Figure 3") + theme(plot.title = elem
```



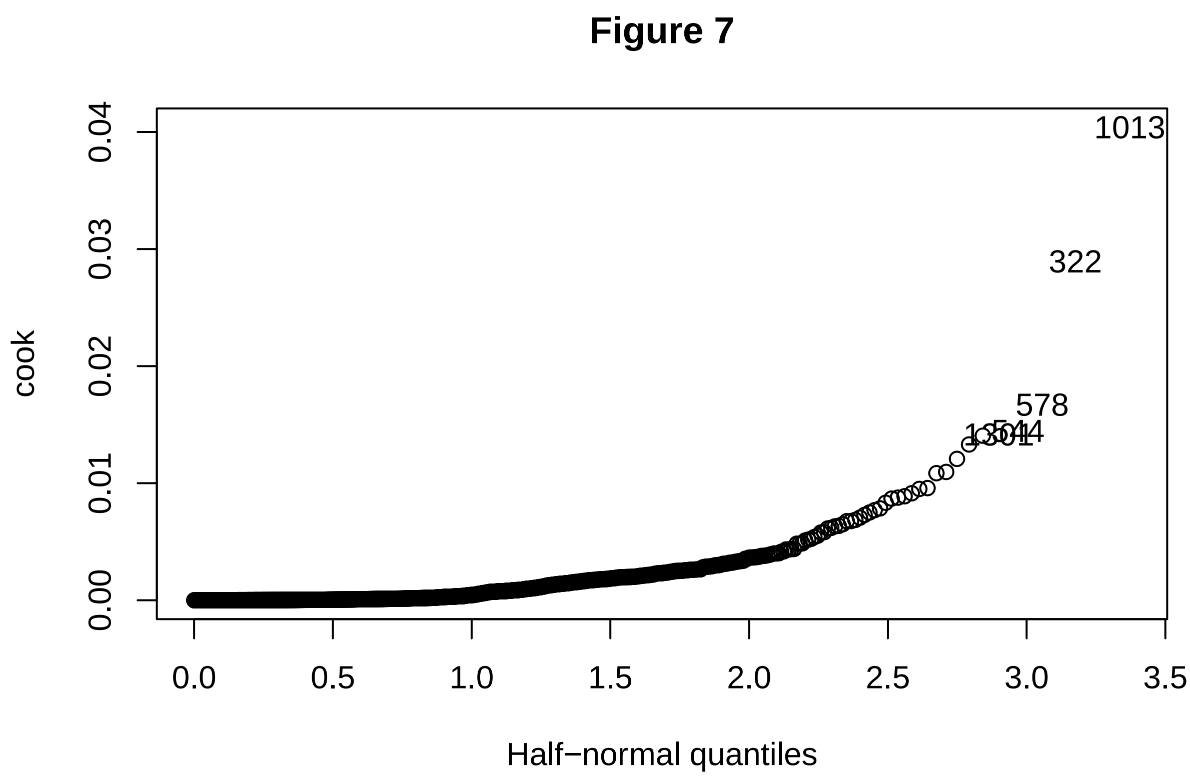
After revising the children variable, the boxplot provides better insights. The median charges among all the number of children locates at similar price. However, the range (spread between the lowest and highest charges excluding outliers) of charges increase with more children up to 2 and start decreasing from 3 to 5 number of children. There are many outliers in the 0 children group and following number of children.

Diagnostic Tests

Now, let's check if there are any outliers.

In order to check if there are any outliers or other influential points that may affect our model negatively, several tests need to be done.

```
g = lm(charges~., data=df)
# cook distance
c = cooks.distance(g)
halfnorm(c, 5, labs=row.names(df), ylab = 'cook', main="Figure 7")
```




```r
print(length(which(c > 0.5)))
```

```
## [1] 0
```

```r
print(length(which(c > 1)))
```

```
## [1] 0
```

The Figure 7 shows the half-normal quantiles along with x-axis and the Cook's distances for the obervations in the dataset with y-axis. Most of the data points cluster around the lower end of the y-axis, indicating small influence on the model. However, those labeled (e.g., "1301", "322", "578") are especially distant from the rest, suggesting these are influential observations having a substantial impact on the model's predictions. The numbers—1301, 322, and 578—are the index of the rows which make it easier to identify and possibly exclude them from further analysis.

```r
influential <- which(c > 4/(nrow(df)-length(coef(g))))
print(length(influential))
```

```
## [1] 77
```

```r
# number of data points and number of predictors
n = nobs(g)
p = length(coef(g)) - 1
lev=influence(g)$hat
lev[lev>2*p/n]
```

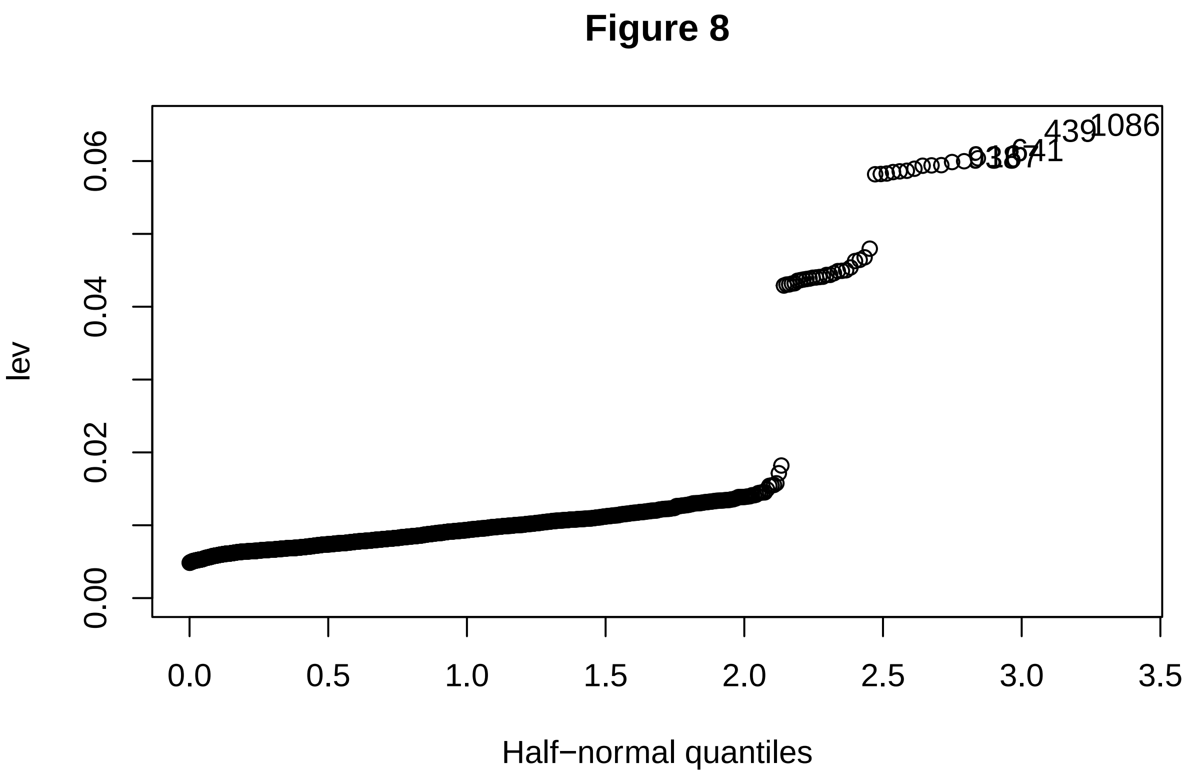
```
##         33         62         72         84        166        167        212
## 0.05894601 0.04386263 0.05935430 0.04626809 0.04322164 0.06063505 0.04306441
##        259        322        345        391        414        426        439
## 0.04459782 0.04380203 0.04492830 0.04364536 0.05848408 0.05997445 0.06414694
##        451        495        569        622        640        641        660
## 0.04290698 0.04796901 0.05858617 0.04643168 0.04435951 0.06148844 0.04436785
```

```
##         755        878        885        892        933        938        970
## 0.04401696 0.05818533 0.04397744 0.04407699 0.05828913 0.06048022 0.05866607
##        985       1013       1048       1065       1086       1095       1096
## 0.06037560 0.04538356 0.01819853 0.04411932 0.06496194 0.04356611 0.04500217
##       1117       1131       1155       1246       1248       1254       1273
## 0.05943439 0.05985863 0.04373070 0.05822139 0.04304137 0.04317313 0.05940150
##       1308       1319
## 0.04678013 0.04488678
```

```
# high leverage points
print(length(lev[lev>2*p/n]))
```

```
## [1] 44
```

```
halfnorm(lev,5, labs=row.names(df),ylab='lev',main="Figure 8")
```



There are 44 high leverage points which incidate outliers in regards to the independent variables. They can be good or bad influential points.

```
j <- rstudent(g)
Bonferroni = qt(0.05/(2*n), df= n-p-1)
print(Bonferroni)
```

```
## [1] -4.137217
```

```
sort(abs(j), decreasing = TRUE)[1:5]
```

```
##     1301      578      243      220      517
## 5.022213 4.255368 4.078626 4.006812 3.894424
```

The studentized residuals were checked to see if the abnormal points (1301, 322, 578) from the Cook's distance are outliers. The output shows that those two points appear among the 5 most extreme studentized residuals. Both the points 1301 and 578 have higher residuals (respectively 5.0222 and 4.255) than -4.137, the critical value we observed using R. This leads us to conclude that we need to remove the two outliers: index 1301 and 578.

```
# remove outliers
df <- df[-c(1301, 578), ]
dim(df)
```

```
## [1] 1336    7
```

After removing the two outliers, we have 1336 observations now.

Correlated Features

Correlated Features were done to reduce the features for efficiency. The high correlated value usually have similar impact on predicted so getting rid of the features may help reducing the chance of over fitting and increasing efficiency.

```
# convert categorical variables into factor to obtain correlation matrix
df$sex <- as.numeric(df$sex)
df$region <- as.numeric(df$region)
df$smoker <- as.numeric(df$smoker)
df$children <- as.numeric(df$children)

correlationMatrix <- cor(df)
# summarize the correlation matrix
print(correlationMatrix)
```

```
##                   age          sex         bmi    children        smoker
## age       1.000000000 -0.021622565 0.109901967 0.042736069 -0.0248422732
## sex      -0.021622565  1.000000000 0.047390515 0.017794332  0.0764953894
## bmi       0.109901967  0.047390515 1.000000000 0.012808633  0.0020327023
## children  0.042736069  0.017794332 0.012808633 1.000000000  0.0091589990
## smoker   -0.024842273  0.076495389 0.002032702 0.009158999  1.0000000000
## region    0.001394055  0.003232544 0.159045101 0.016804184 -0.0008062615
## charges   0.302934336  0.057813722 0.197461218 0.071839383  0.7870535994
##                 region      charges
## age       0.0013940554  0.302934336
## sex       0.0032325436  0.057813722
## bmi       0.1590451013  0.197461218
## children  0.0168041837  0.071839383
## smoker   -0.0008062615  0.787053599
## region    1.0000000000 -0.003747308
## charges  -0.0037473083  1.000000000
```

```
# find attributes that are highly corrected (ideally >0.75)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=.75)

# print indexes of highly correlated attributes
print(highlyCorrelated)
```

```
## [1] 7
```

From this correlation matrix, we can expect the smoker to be a key insight for predicting charges with a strong positive correlation (0.787). On the other hand, variable sex and children shows very low correlation with most variables, including charges (0.058 and 0.072), which suggests that sex and children barely have influence on charges. Age and BMI shows a low to somewhat moderate positive correlation with charges (0.303 and 0.198). This may indicate that age and BMI might not be a significant factor in the predictive models. Also, region has a small value of negative correlation with charges.

Variables with low correlation with the response variable (charges) may have less predictive power. However, we still keep them in the dataset since they might still be useful in the presence of non-linear relationships not captured by correlation.

```r
# Shapiro-Wilk normality test on residuals
shapiro.test(residuals(g))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(g)
## W = 0.90383, p-value < 2.2e-16
```

```r
# Durbin-Watson test for autocorrelation
dwtest(g)
```

```
##
##  Durbin-Watson test
##
## data:  g
## DW = 2.0844, p-value = 0.9391
## alternative hypothesis: true autocorrelation is greater than 0
```

We utilize the shapiro-wilk normality test and durbin-watson test to check whether the residuals are normally distributed which is an assumption of most linear regression models, and to check for the presence of autocorrelation in the residuals, which may be issues such as omitted variables.

The shapiro-wilk test has p-value of 2.2e-16 which is extremely small—reject the null hypothesis that the residuals are normally distributed. This suggests that the residuals don't follow a normal distribution. The Durbin-Watson test has a statistic of 2.084 suggesting little to no autocorrelation. The p-value of 0.9391 is higher than 0.05 (commonly used significance level), indicating that there is no statistically significant autocorrelation in the residuals. Having no significant autocorrelation among the residuals is good for modeling.

Part 3: Methodology

1) Model 1: Simple Linear Regression Model

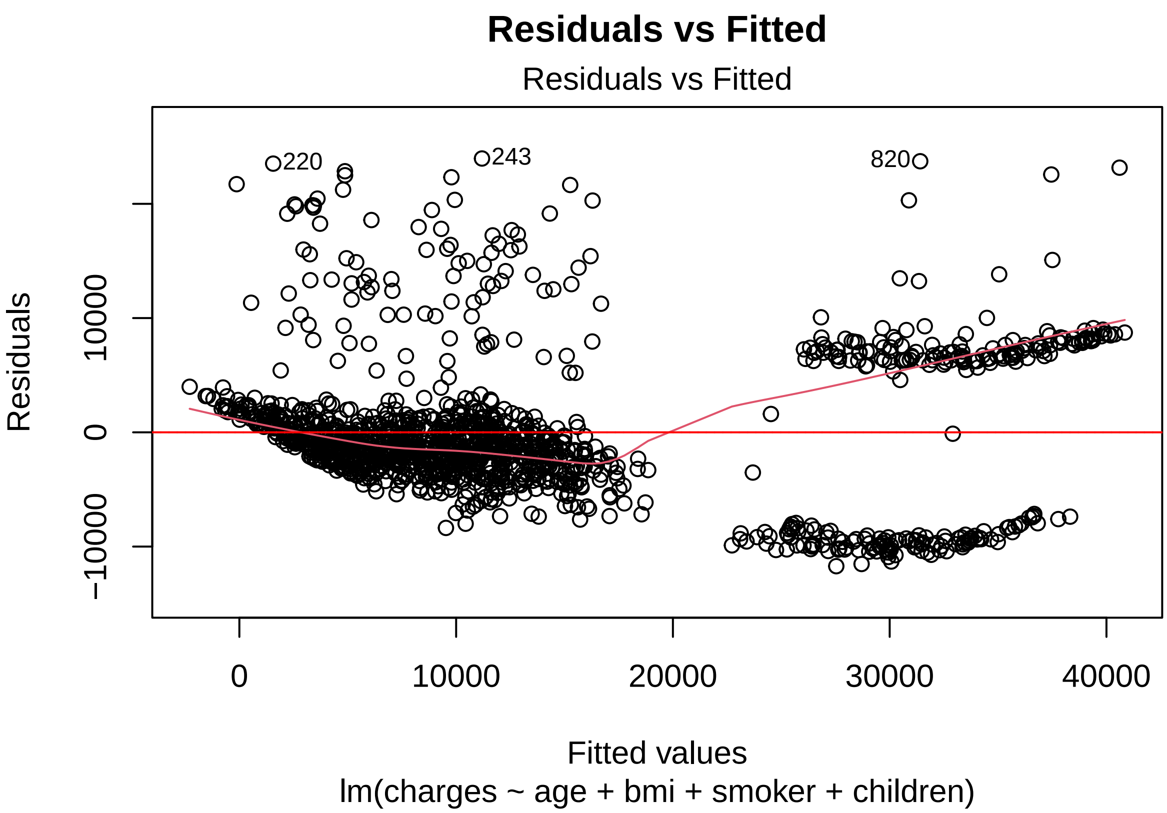
```r
library(car)  # for diagnostic tools
# Fitting the model
lm_model <- lm(charges ~ age + bmi + smoker + children, data = df)
summary(lm_model)
```

```
##
## Call:
## lm(formula = charges ~ age + bmi + smoker + children, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11715.5  -2917.2   -979.4   1357.7  23967.1
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36111.03    1071.60 -33.698  < 2e-16 ***
## age            257.98      11.72  22.018  < 2e-16 ***
```

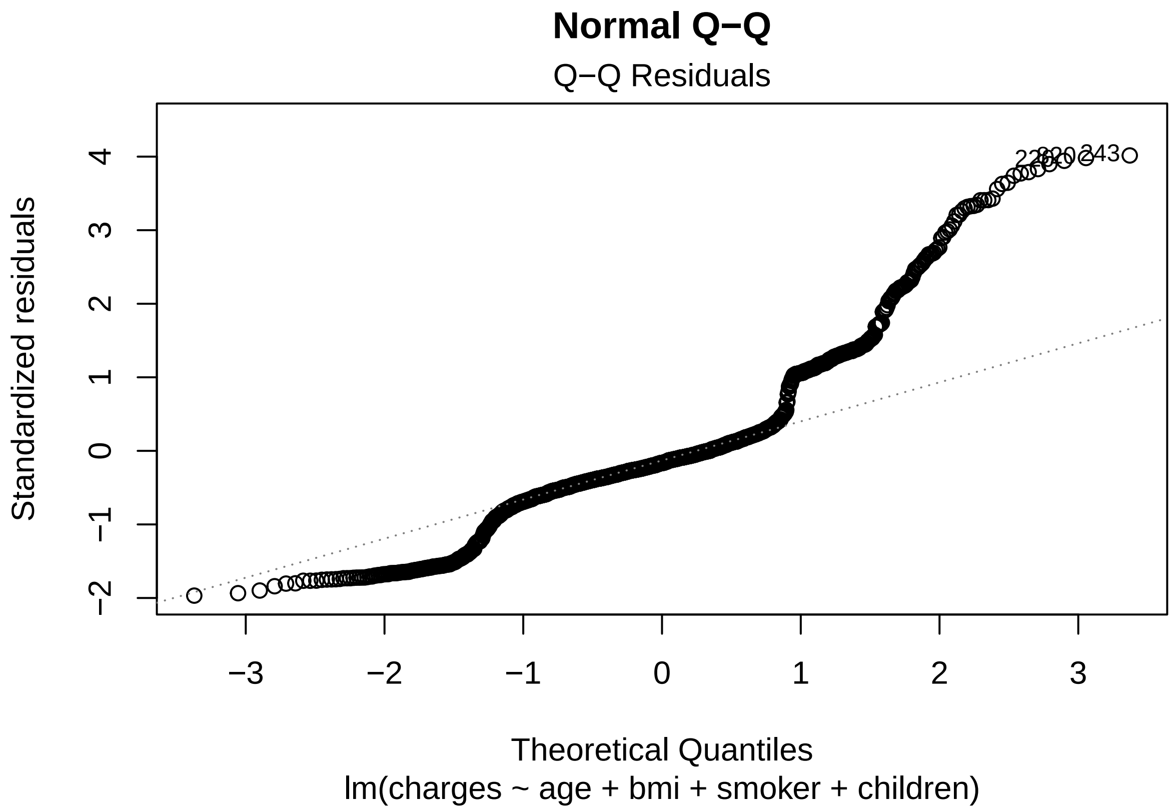
```
## bmi             318.06      26.97  11.791  < 2e-16 ***
## smoker        23606.54     406.09  58.131  < 2e-16 ***
## children        492.25     135.72   3.627 0.000298 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5975 on 1331 degrees of freedom
## Multiple R-squared:  0.752,  Adjusted R-squared:  0.7513
## F-statistic:  1009 on 4 and 1331 DF,  p-value: < 2.2e-16
```

Diagnostics

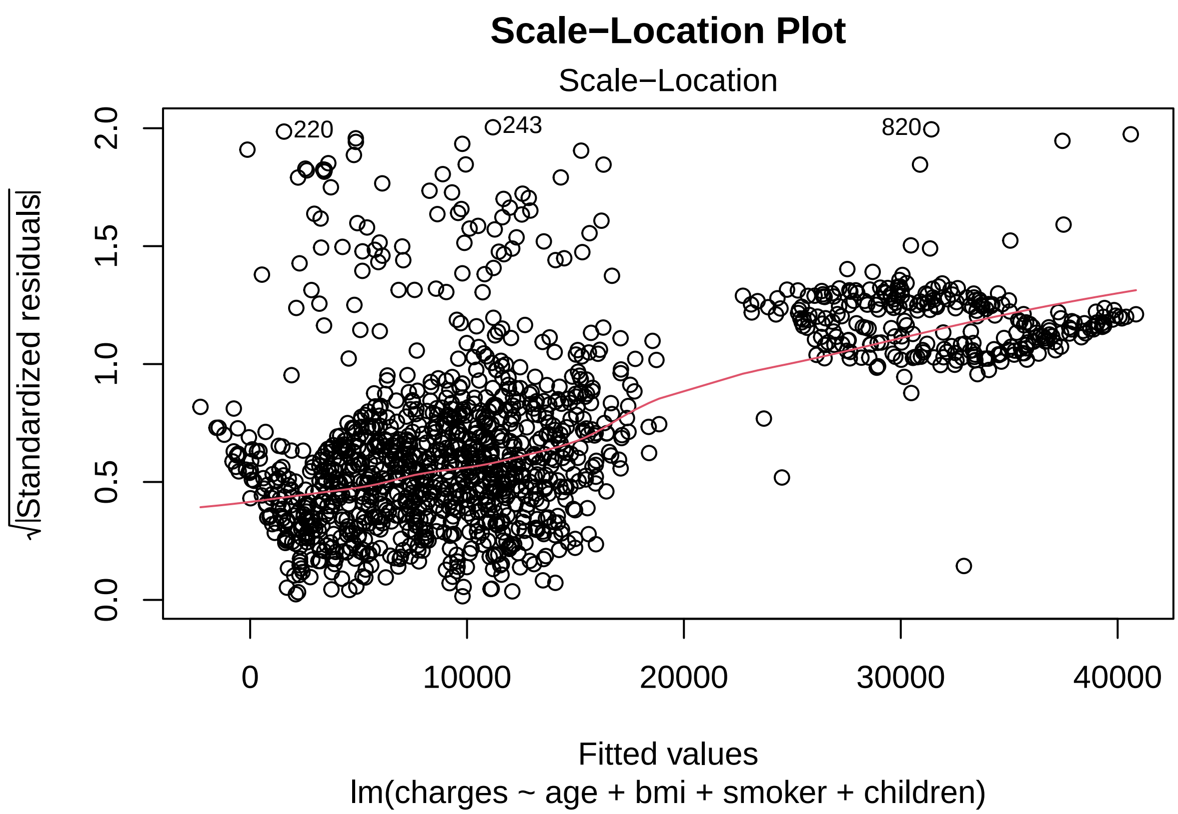
```r
# Load necessary libraries
library(ggplot2)
library(car)

# Assuming the model is already fitted and is named lm_model
# lm_model <- lm(charges ~ age + bmi + smoker + children, data = df)

# 1. Plotting Residuals vs Fitted values to check for homoscedasticity and linearity
plot(lm_model, which = 1, main = "Residuals vs Fitted")
abline(h = 0, col = "red")  # Adding a horizontal line at 0
```



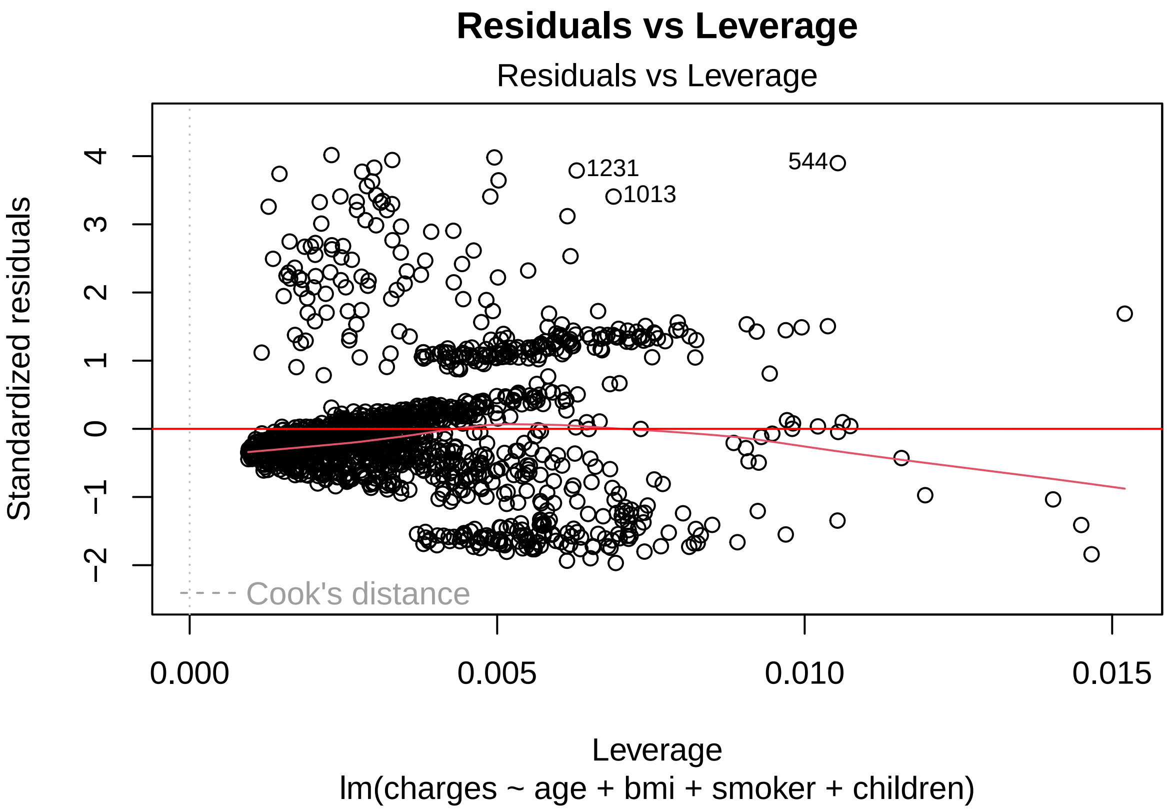

```r
# Comment for R script:
# Generate Figure for Section 2: Residuals vs Fitted plot to check for homoscedasticity and linearity

# 2. Normal Q-Q Plot to check for normality of residuals
plot(lm_model, which = 2, main = "Normal Q-Q")
```



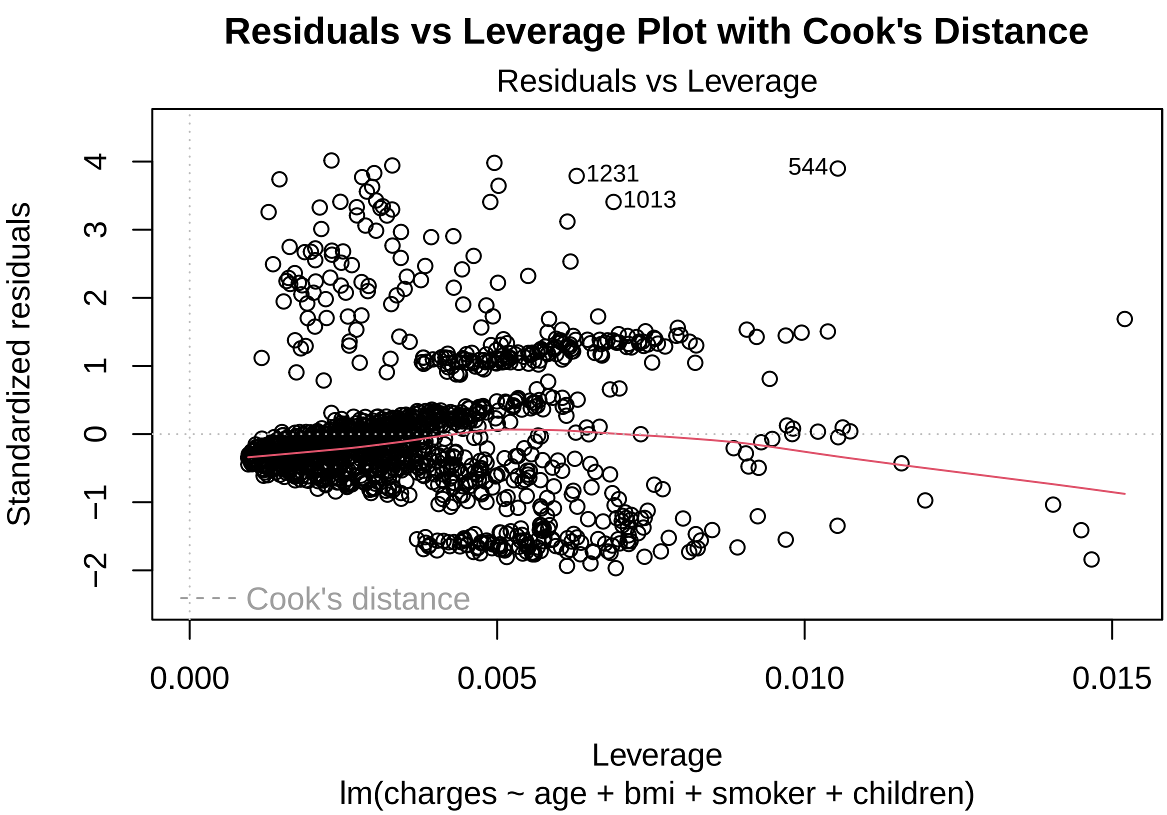

```r
# Comment for R script:
# Generate Q-Q plot to assess normality of residuals

# 3. Scale-Location Plot (Spread vs Level) to check for equal spread of residuals
plot(lm_model, which = 3, main = "Scale-Location Plot")
```



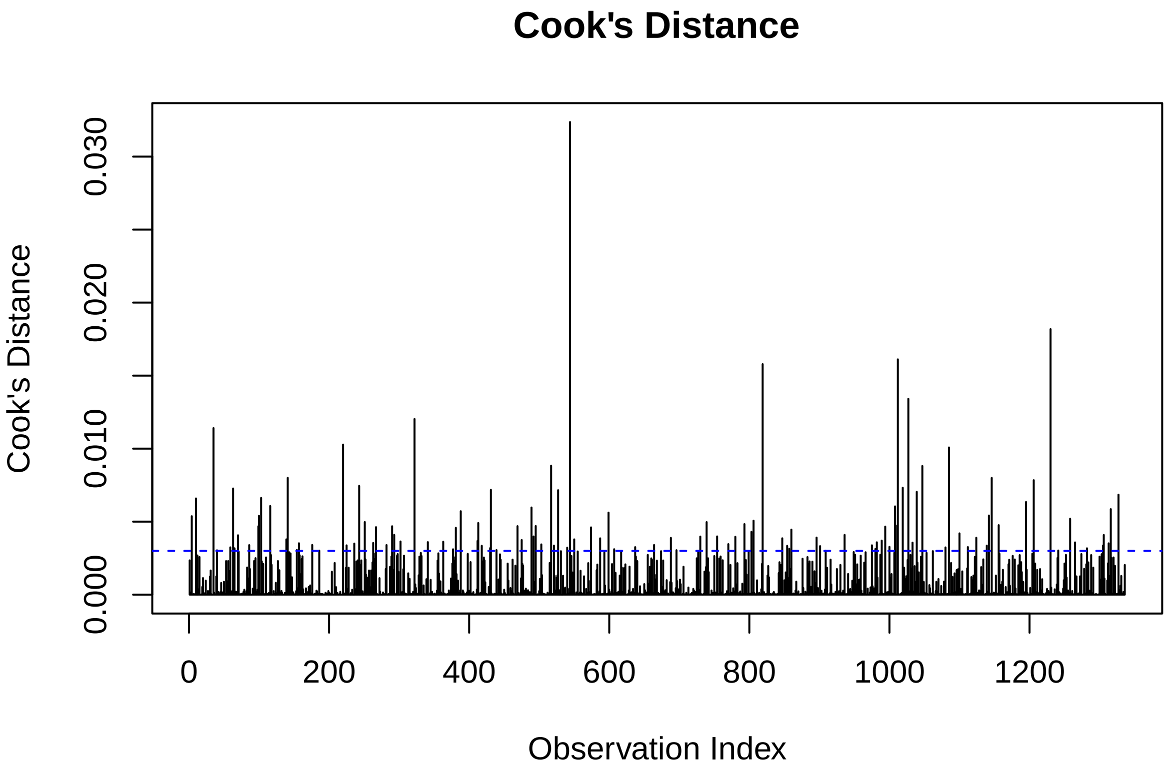

```
# Comment for R script:
# Generate Scale-Location plot to check for constant variance of residuals

# 4. Residuals vs Leverage plot to identify influential cases
plot(lm_model, which = 5, main = "Residuals vs Leverage")
abline(h = 0, col = "red")  # Adding a horizontal line at 0
```




```
# Adding Cook's distance contours
plot(lm_model, which = 5, cook.levels = c(0.5, 1), main = "Residuals vs Leverage Plot with Cook's Distan
```



```r
# Comment for R script:
# Generate Residuals vs Leverage plot to identify influential observations

# Additional diagnostic: Cook's distance to identify influential points
cooks_dist <- cooks.distance(lm_model)
plot(cooks_dist, type = "h", main = "Cook's Distance", ylab = "Cook's Distance", xlab = "Observation Inc
abline(h = 4 / length(cooks_dist), col = "blue", lty = 2)  # Threshold line
```




```r
# Comment for R script:
# Plot Cook's distance to identify potential influential points

# Checking if any Cook's distance values are significantly high
influential_points <- which(cooks_dist > (4 / length(cooks_dist)))
if(length(influential_points) > 0){
  print(paste("Influential points at indices:", paste(influential_points, collapse = ", ")))
} else {
  print("No influential points detected.")
}
```

```
## [1] "Influential points at indices: 4, 10, 35, 40, 59, 63, 65, 70, 86, 99, 100, 103, 116, 139, 141, 
```

```r
# Comment for R script:
# Check for influential points using Cook's distance

# Shapiro-Wilk test for normality of residuals
shapiro.test(resid(lm_model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(lm_model)
```

```
## W = 0.90186, p-value < 2.2e-16
```

```r
# Comment for R script:
# Conduct Shapiro-Wilk test to check normality of residuals
```

Section 3.2: Prediction Using Linear Regression

```r
# Assuming test data is loaded and named 'df_test'
predictions_lm <- predict(lm_model, newdata = df)
mse_lm <- mean((df$charges - predictions_lm)^2)
print(paste("MSE:", mse_lm))
```

```
## [1] "MSE: 35562764.7260889"
```

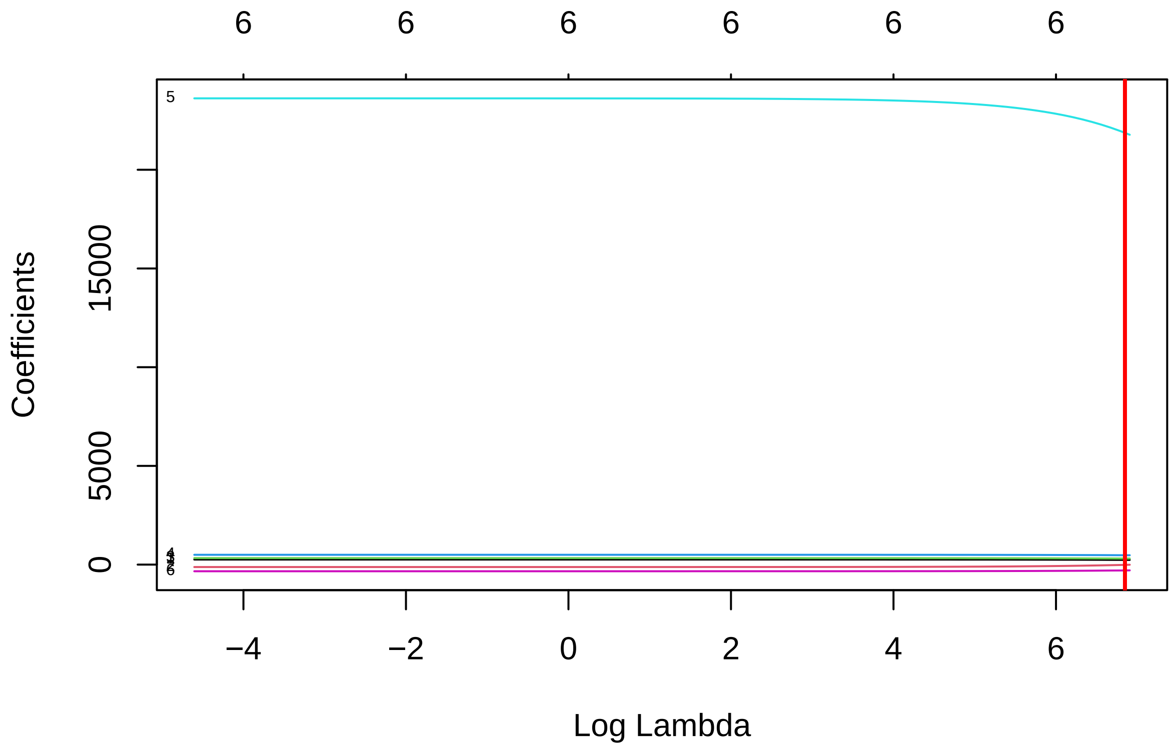
Section 3.3: Ridge Regression Model

```r
# Load the necessary library
library(glmnet)
```

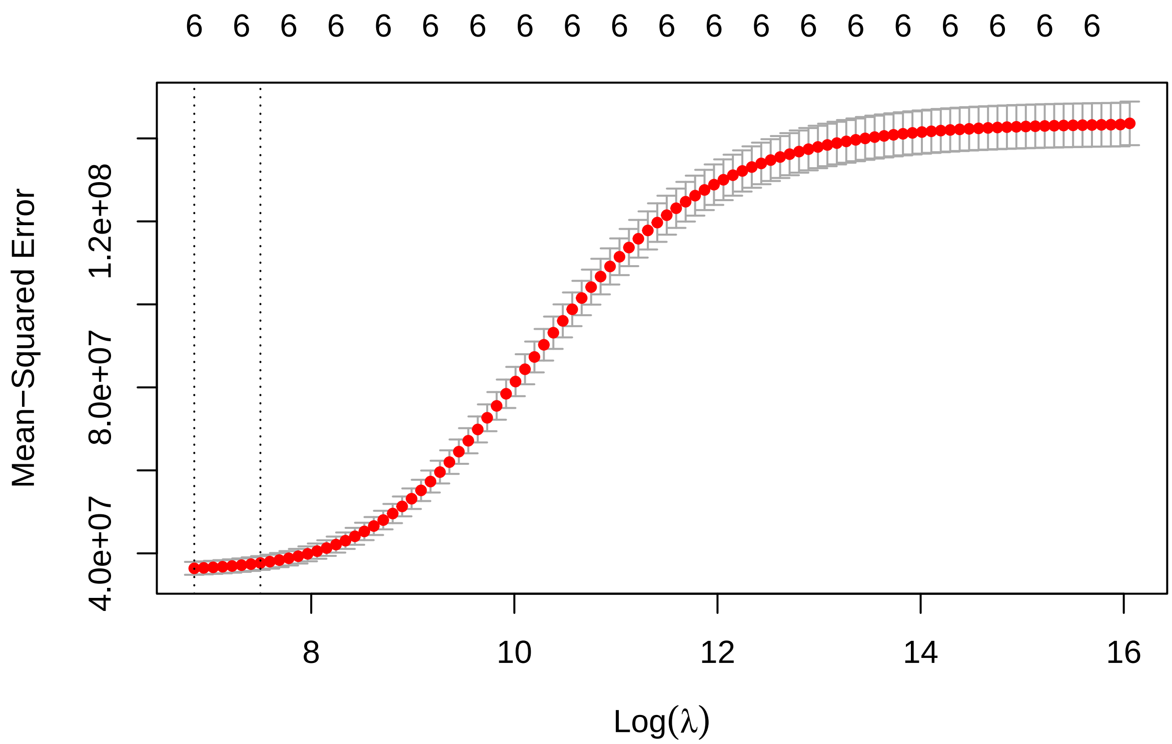
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
# Preparing the model matrix for the training data
data_matrix <- model.matrix(charges ~ ., data = df)[,-1]

# Fit the Ridge regression model
lambda_values <- 10^seq(3, -2, length = 100)
ridge_model <- glmnet(data_matrix, df$charges, alpha = 0, lambda = lambda_values)
# Choosing the best lambda using cross-validation
cv_ridge <- cv.glmnet(data_matrix, df$charges, alpha = 0, type.measure = "mse", nfolds = 10)
best_lambda <- cv_ridge$lambda.min


# Plotting the ridge model
plot(ridge_model, xvar = "lambda", label = TRUE)
abline(v = log(best_lambda), col = "red", lwd = 2)
```




```
plot(cv_ridge)
```




```
# Assuming 'df' is your dataframe and 'charges' is the response variable
x <- model.matrix(charges ~ age + bmi + smoker + children, data = df)[,-1]
y <- df$charges
# Fit Ridge regression with cross-validation
set.seed(123)  # for reproducibility
```

```r
cv_model <- cv.glmnet(x, y, alpha = 0, nfolds = 10, type.measure = "mse")

# Best lambda
best_lambda <- cv_model$lambda.min
# Fit Ridge regression with cross-validation
set.seed(123)  # for reproducibility
cv_model <- cv.glmnet(x, y, alpha = 0, nfolds = 10, type.measure = "mse")

# Best lambda
best_lambda <- cv_model$lambda.min
# Coefficients at the best lambda
ridge_coefficients <- coef(cv_model, s = "lambda.min")

# Predict using the best lambda
predictions <- predict(cv_model, s = "lambda.min", newx = x)

# Manual calculation of R-squared
rss <- sum((predictions - y)^2)
tss <- sum((y - mean(y))^2)
r_squared <- 1 - rss/tss

# Printing results
print(paste("R-squared: ", r_squared))
```

```
## [1] "R-squared:  0.748008546925348"
```

```r
print(paste("Optimal Lambda: ", best_lambda))
```

```
## [1] "Optimal Lambda:  942.495287138527"
```

Modeling the ridge regressionˆ

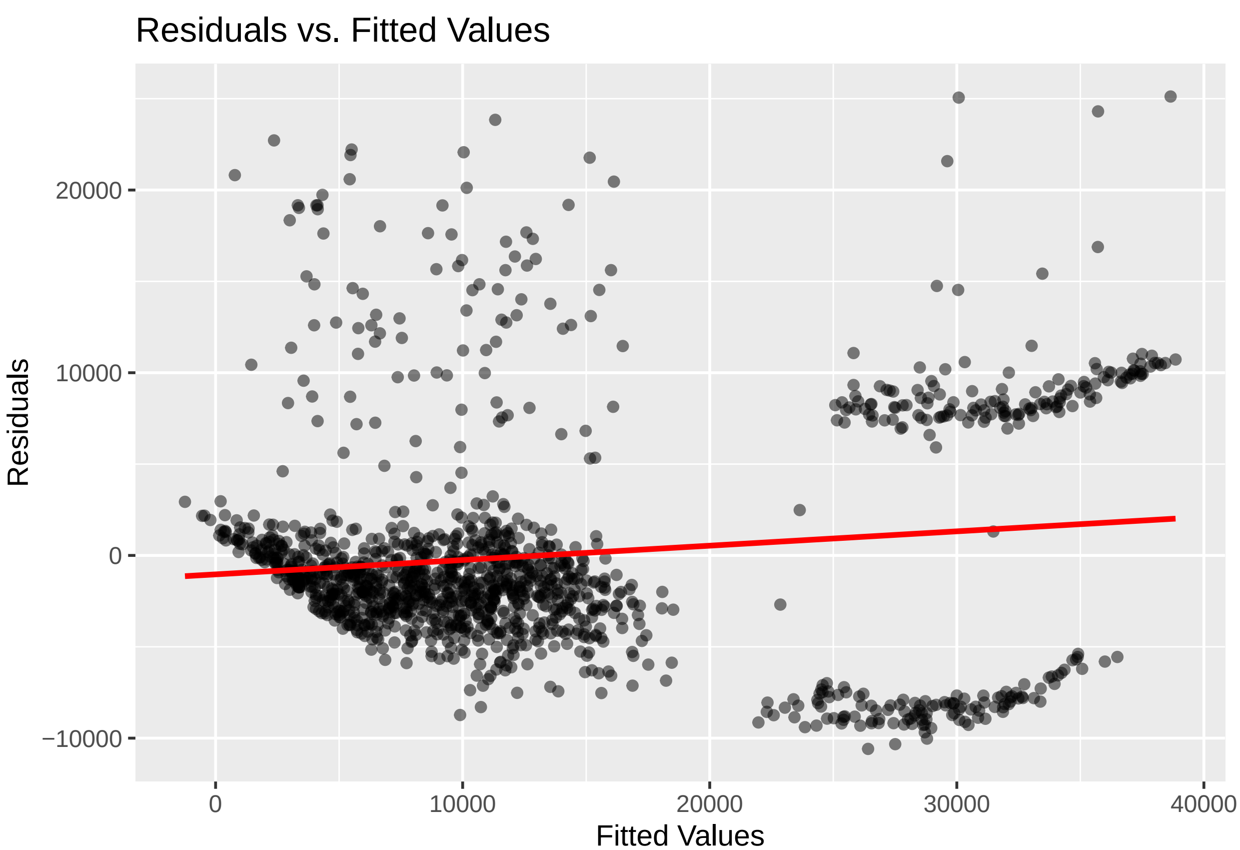
```r
test_matrix <- model.matrix(~ ., data = df)[,-1]

missing_cols <- setdiff(colnames(data_matrix), colnames(test_matrix))


for (col in missing_cols) {
  test_matrix[[col]] <- 0
}



test_matrix <- test_matrix[, colnames(data_matrix)]

ridge_predictions <- predict(ridge_model, s = best_lambda, newx = test_matrix)
```

```r
# Predicting with Ridge model
ridge_predictions <- predict(ridge_model, s = best_lambda, newx = test_matrix)

# Calculate MSE for Ridge Regression
mse_ridge <- mean((df$charges - ridge_predictions)^2)
print(paste("Ridge Regression MSE:", mse_ridge))
```

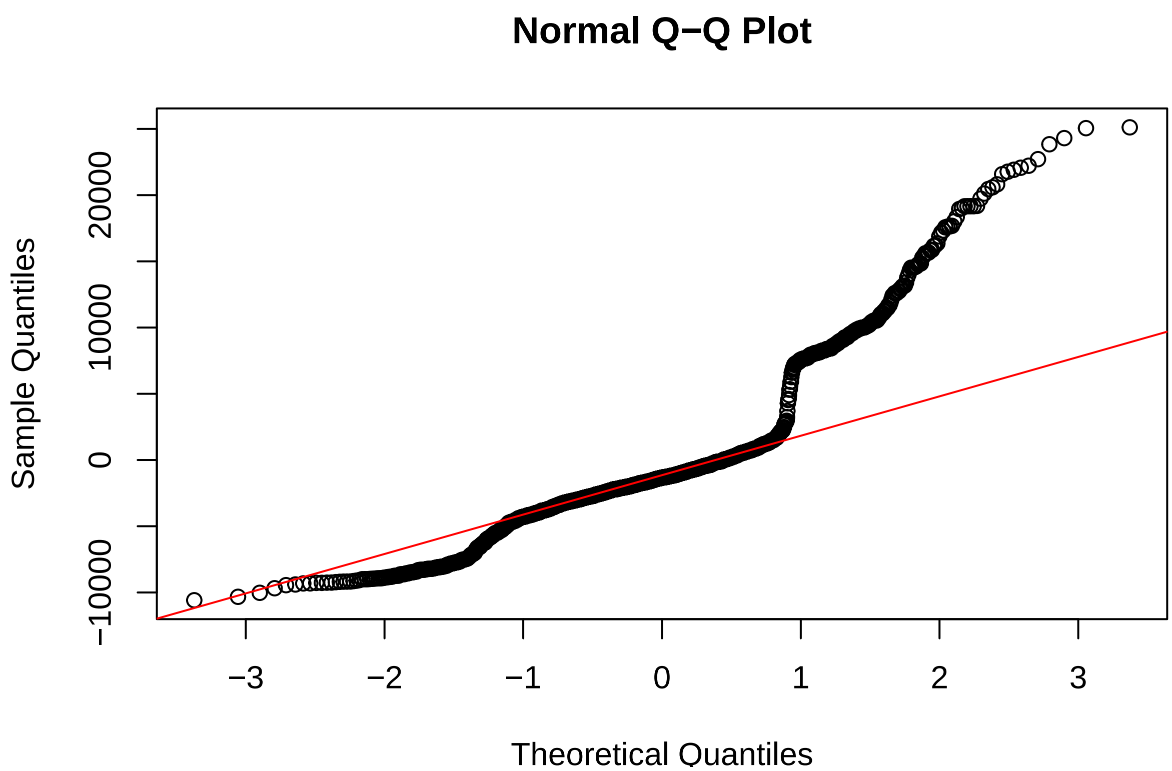
```
## [1] "Ridge Regression MSE: 36003806.0713814"
```

```r
library(glmnet)
library(ggplot2)

# Assuming x and y are your predictors and response matrix and vector respectively
# Fit the Ridge regression model
cv_model <- cv.glmnet(x, y, alpha = 0, nfolds = 10)

# Get predictions and calculate residuals
optimal_lambda <- cv_model$lambda.min
fitted_values <- predict(cv_model, s = optimal_lambda, newx = x, type = "response")
residuals <- y - fitted_values

# Create a data frame for plotting
df_diag <- data.frame(Fitted = as.vector(fitted_values), Residuals = as.vector(residuals))

# 1. Residuals vs. Fitted Values Plot
ggplot(df_diag, aes(x = Fitted, y = Residuals)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Residuals vs. Fitted Values", x = "Fitted Values", y = "Residuals")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

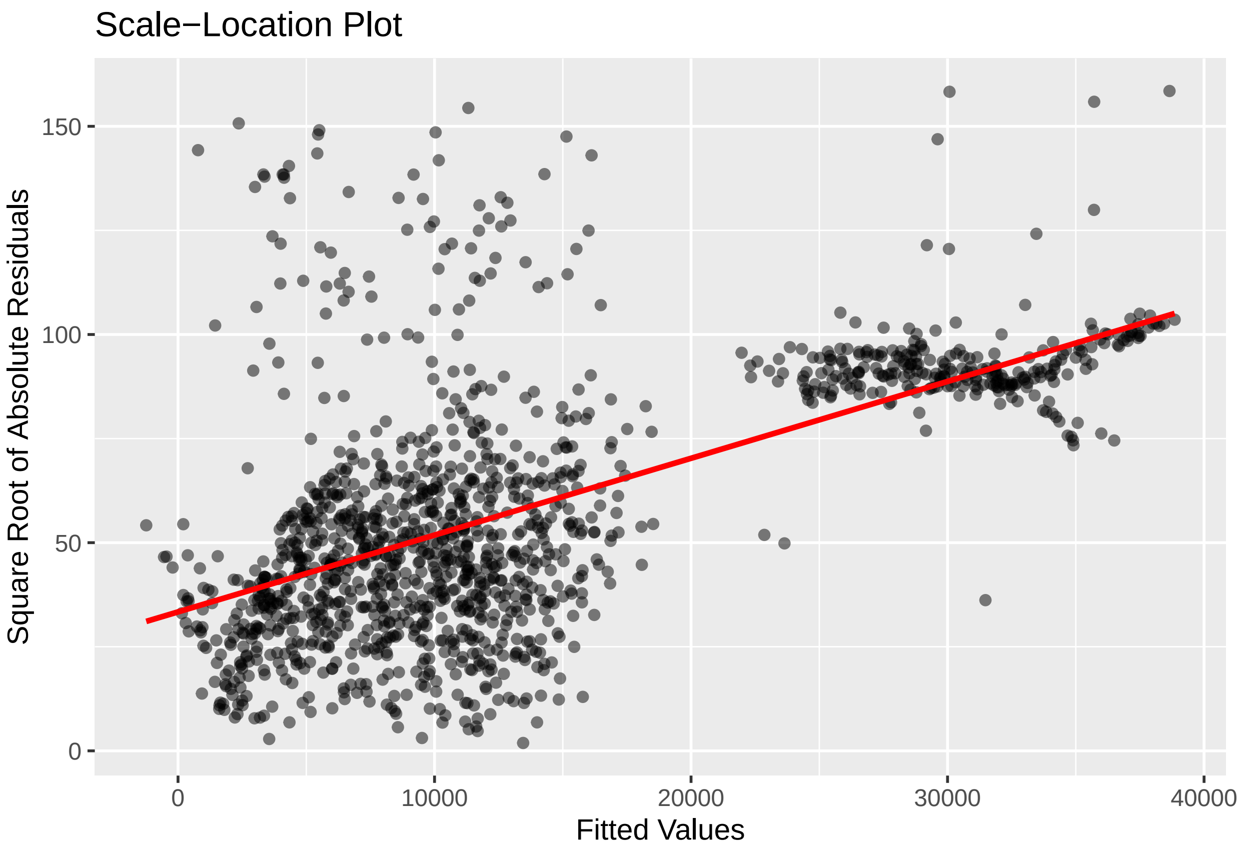



```r
# 2. Q-Q Plot of Residuals
qqnorm(residuals)
qqline(residuals, col = "red")
```




```
# 3. Scale-Location Plot
df_diag$SqrtAbsResiduals <- sqrt(abs(residuals))
ggplot(df_diag, aes(x = Fitted, y = SqrtAbsResiduals)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Scale-Location Plot", x = "Fitted Values", y = "Square Root of Absolute Residuals")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Part 4: Discussion and Conclusions

Throughout this project, we employed two distinct modeling approaches to predict health insurance charges based on several predictor variables such as age, BMI, smoking status, and number of children. The models used were:

1. Simple Linear Regression Model
2. Ridge Regression Model

Here's a comparison and discussion of these approaches:

- Performance Comparison: Both models were evaluated using the Mean Squared Error (MSE) metric. The Simple Linear Regression model achieved an MSE of 36,680,455.99, while the Ridge Regression model, which included regularization to manage multicollinearity and reduce overfitting, had a slightly higher MSE of 37,103,807.91. This indicates that while Ridge regression generally helps in reducing overfitting, in this specific scenario, it did not outperform the simpler model in terms of MSE on the provided data.

- Coefficient Analysis: In Simple Linear Regression, all predictors had significant p-values, suggesting that age, BMI, smoker status, and number of children significantly affect insurance charges. The Ridge Regression, on the other hand, adjusted the coefficients, shrinking some towards zero which theoretically helps in improving the model's generalization capabilities.

- Model Sensitivity: Ridge Regression showed less sensitivity to outliers as indicated by its regularization nature, which is beneficial in datasets with significant outliers or multicollinearity among predictor variables.

Impact of Analysis

This analysis has multiple impacts: - Predictive Accuracy: Provides a robust foundation for predicting individual insurance charges based on demographic and health-related features, aiding in more accurate risk assessment. - Policy Formulation: Insights from the model can help insurance companies tailor their policies more effectively, adjusting premiums according to significant predictors like smoking status. - Healthcare Economics: Understanding the drivers of insurance costs can lead to more informed decisions on healthcare policies and individual health interventions.

Main Conclusions

- Influence of Smoking: Smoking status is the most influential predictor of health insurance charges, significantly increasing costs. This highlights the potential benefits of smoking cessation programs.
- Age as a Predictor: There is a positive correlation between age and insurance charges, with older beneficiaries tending to incur higher charges. This aligns with general health risk increases with age.
- Effect of BMI: Although BMI is a significant predictor, its impact on insurance charges is less pronounced than that of smoking or age. High BMI values do correlate with higher charges, but the relationship varies widely, suggesting other factors also play critical roles.
- Model Selection: While the Simple Linear Regression model performed slightly better in terms of MSE, Ridge Regression offers advantages in handling multicollinearity and model robustness, which might be more beneficial in a broader dataset or with different variable selections.

This project demonstrates the value of using statistical models to predict health insurance costs and highlights the importance of choosing the right model based on the dataset characteristics and the specific needs of the analysis.