Week 2 - Data Science II

Phileas Dazeley-Gaist

11/01/2022

```r
# # ----------------------------------------------------------------------
#
# Simple Linear Regression
#
# # ----------------------------------------------------------------------

# First, let's check out this Boston data
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

```r
# Cool...but what do these variables mean? Let's use our ?? command to check it out and see if
??Boston

# So medv is a column that stands for the median value of owner-occupied homes, as measured in
# That could be a good initial variable to check out as a dependent variable.
# Let's also check out that variable lstat - "lower status of population". This variable has s
# but it basically means "lower socioeconomic status". Let's build a model that sees how well

# We can use the lm ("linear model") command to build a linear regression:

# lm_fit <- lm(medv ~ lstat)

# why didn't this work?

?lm

#...ahhh, we forgot to tell it which dataset we were using!

lm_fit <- lm(medv ~ lstat, data = Boston)
```

```r
# Let's see what is in this new object we just created
lm_fit
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```r
# That's not super, uh, informative. Let's check it out even further with summary()

summary(lm_fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
# We can extract stuff from this fitted model fairly easily.

# ...we can grab the names of all the internal elements if we want:
names(lm_fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
# ...and thus call them!

as.data.frame(lm_fit$residuals)
```

```
##    lm_fit$residuals
## 1       -5.822595098
## 2       -4.270389786
## 3        3.974858016
## 4        1.639304221
## 5        6.709922176
## 6       -0.904083746
## 7        0.155272588
## 8       10.739604245
## 9       10.381136279
## 10       0.592003070
## 11      -0.125331595
## 12      -3.046685955
## 13       2.071434468
## 14      -6.306433217
## 15      -6.606334510
## 16      -6.606922853
## 17      -5.202516132
## 18      -3.116616860
## 19      -3.247763934
## 20      -5.637284169
## 21      -0.983803463
## 22      -1.814658317
## 23      -1.568916977
## 24      -1.166859727
## 25      -3.468036413
## 26      -4.968526049
## 27      -3.883609950
## 28      -3.336988046
## 29      -3.993209151
## 30      -2.172249621
## 31      -0.382725484
## 32      -7.665197306
## 33       4.972026713
## 34      -4.020435238
## 35      -1.729837024
## 36      -6.457363135
## 37      -3.713777753
## 38      -5.221908047
## 39      -0.229840926
## 40       0.350372329
## 41       2.227256841
## 42      -3.355602007
```

```
## 43       -3.734054134
## 44       -2.785473687
## 45       -4.280869551
## 46       -5.553836978
## 47       -1.110642524
## 48       -0.092913029
## 49        9.117179710
## 50        0.236958651
## 51       -2.075677071
## 52       -5.094875473
## 53       -4.537580292
## 54       -3.144924827
## 55       -1.593110444
## 56        5.415896512
## 57       -4.372056108
## 58        0.798854068
## 59       -4.736502313
## 60       -6.194385838
## 61       -3.360691877
## 62       -4.835128211
## 63       -5.960008729
## 64       -0.528372019
## 65        6.094056418
## 66       -6.617110397
## 67       -5.425335497
## 68       -4.858441114
## 69       -4.717694839
## 70       -5.302907060
## 71       -3.969509222
## 72       -3.467353264
## 73       -6.509568447
## 74       -3.990468752
## 75       -4.012506261
## 76       -4.660399657
## 77       -3.181750115
## 78       -3.996834016
## 79       -1.630231854
## 80       -5.608391760
## 81       -1.528079798
## 82       -3.794484545
## 83       -3.369509222
## 84       -4.518970233
## 85       -1.514366096
## 86       -1.750018599
## 87        0.163793810
## 88       -4.335424334
## 89       -5.728569434
## 90       -0.438559563
```

```
## 91       -3.583906073
## 92       -4.763436179
## 93       -3.901438153
## 94       -3.654034393
## 95       -3.892818223
## 96        0.163987323
## 97       -2.380281208
## 98        8.145866900
## 99       12.637835314
## 100       4.526964620
## 101       1.895624033
## 102      -0.766962336
## 103      -5.854816249
## 104      -2.485177565
## 105      -2.739732348
## 106       0.593471977
## 107       2.674080062
## 108      -0.767645485
## 109      -3.096735309
## 110      -0.380573428
## 111      -0.503199281
## 112      -2.101339445
## 113      -0.353540855
## 114       0.382502576
## 115      -6.125825133
## 116      -1.281063064
## 117      -1.915246660
## 118      -5.568332536
## 119       0.448417688
## 120      -2.323669175
## 121       1.098368334
## 122      -0.696636601
## 123       2.980544033
## 124       6.886913200
## 125       0.948026760
## 126       0.916390050
## 127       7.044504504
## 128      -2.022492488
## 129      -1.932581325
## 130      -2.829935731
## 131      -3.383219022
## 132      -3.306235802
## 133      -0.989292066
## 134      -1.874599092
## 135      -2.508486566
## 136      -0.341003840
## 137      -1.098006801
## 138      -3.592620808
```

```
## 139     -0.998788657
## 140      0.784070191
## 141      2.399351507
## 142     12.537357383
## 143      4.326482788
## 144      6.146463047
## 145      5.073104692
## 146      5.657531155
## 147     -3.135519139
## 148      8.101116537
## 149     10.151556819
## 150      1.224717759
## 151      0.341855009
## 152     -2.337185461
## 153     -7.739242712
## 154     -0.152561584
## 155     -3.189094651
## 156     -4.684099586
## 157     -6.120044310
## 158     11.106885654
## 159     -4.145023535
## 160     -4.232976155
## 161     -2.328569434
## 162     17.089744503
## 163     17.270253880
## 164     18.600322975
## 165     -0.795266402
## 166     -0.233856719
## 167     18.961341730
## 168      0.779758275
## 169     -0.208293053
## 170     -1.499282195
## 171     -3.444628705
## 172     -4.024747154
## 173      2.502384127
## 174     -2.365394721
## 175     -2.795365109
## 176     -0.090077824
## 177     -1.748841913
## 178     -3.978030444
## 179      1.920500649
## 180      7.434407864
## 181     12.428532235
## 182     10.624125514
## 183      7.925397006
## 184      3.342439450
## 185      5.127849086
## 186      7.539308123
```

```
## 187      19.673878745
## 188       3.792488804
## 189      -0.421615826
## 190       5.466925137
## 191       7.291410825
## 192       0.401890590
## 193       4.572800766
## 194       1.324907370
## 195      -1.292624710
## 196      18.267805701
## 197       2.622360484
## 198       3.926084056
## 199       6.335485842
## 200       4.678384174
## 201       2.573878745
## 202      -3.394974181
## 203      10.700812611
## 204      17.565847158
## 205      18.182301259
## 206      -1.626804404
## 207       0.268200531
## 208       5.104050449
## 209       3.773882647
## 210       7.382798699
## 211       3.553511460
## 212       7.528342624
## 213       3.075450261
## 214       2.457622059
## 215      17.220117524
## 216      -0.556873499
## 217       1.581325890
## 218       3.352137359
## 219       3.971043540
## 220      -1.578322665
## 221       1.371138346
## 222       7.534218252
## 223       2.380149203
## 224       2.766534209
## 225      14.179363445
## 226      19.844887629
## 227       6.019813598
## 228       3.088473011
## 229      15.870352587
## 230       0.518344691
## 231       0.814234092
## 232       2.133918228
## 233       9.492781024
## 234      17.498854068
```

```
## 235       2.094056418
## 236      -0.217303910
## 237      -0.390370045
## 238       1.439892564
## 239      -4.811526989
## 240      -4.251977142
## 241      -1.742279234
## 242      -2.673228893
## 243      -1.694287130
## 244      -5.923084733
## 245      -5.078223957
## 246       1.484070191
## 247      -1.551388799
## 248      -4.410839939
## 249      -1.009371032
## 250      -2.121517119
## 251      -4.548549692
## 252      -6.343163699
## 253      -1.600166661
## 254      11.609333833
## 255      -6.412016625
## 256      -4.865884357
## 257      12.400812611
## 258      20.310411812
## 259       8.847043586
## 260       2.101499662
## 261       8.357132423
## 262      15.443517429
## 263      19.860950801
## 264       7.134214350
## 265       9.641558886
## 266      -1.825825133
## 267      10.197389063
## 268      22.514526313
## 269      11.948315078
## 270      -0.885667201
## 271      -1.103199281
## 272      -3.093015638
## 273      -2.809959375
## 274       6.897483868
## 275       1.199833339
## 276       0.277306195
## 277       4.393957711
## 278       2.498364432
## 279       1.377013974
## 280       5.153898486
## 281      14.418344691
## 282       5.206885654
```

```
## 283      14.305807675
## 284      18.448315078
## 285       5.104046548
## 286      -4.734934698
## 287      -2.169702735
## 288      -4.570488494
## 289      -5.033465791
## 290      -0.718871525
## 291      -2.890176531
## 292       6.128334820
## 293      -2.188608917
## 294      -2.502417424
## 295      -2.973327600
## 296       0.002968569
## 297      -0.432976155
## 298       0.794940884
## 299      -7.332095591
## 300      -1.050606943
## 301      -3.987041302
## 302      -3.528372019
## 303       0.083087018
## 304       3.163398980
## 305       8.130001142
## 306       2.330099850
## 307       4.992978439
## 308       0.800030754
## 309      -7.440616813
## 310      -4.781848822
## 311      -6.445217048
## 312      -6.772545744
## 313      -4.019262453
## 314      -5.448450985
## 315      -1.937382877
## 316      -7.428273311
## 317       0.660563775
## 318       0.389945820
## 319      -1.611329574
## 320      -1.459712606
## 321      -3.913485532
## 322      -4.927001819
## 323      -6.838460855
## 324      -4.900261466
## 325      -3.739538834
## 326      -5.127590162
## 327      -5.711037354
## 328      -0.202709645
## 329      -5.781848822
## 330      -4.980478623
```

```
## 331      -6.117892254
## 332      -5.644727412
## 333      -7.714954439
## 334      -6.957560550
## 335      -7.441007742
## 336      -5.843945556
## 337      -5.743357213
## 338      -6.021319704
## 339      -5.868920879
## 340      -6.300360174
## 341      -7.027882383
## 342       3.361930073
## 343      -9.835913969
## 344      -3.832486519
## 345       1.025886641
## 346      -7.049821184
## 347      -5.316715567
## 348      -5.411526989
## 349      -4.363045250
## 350      -2.358050186
## 351      -5.972545744
## 352      -5.238069927
## 353      -8.552956414
## 354      -0.178618787
## 355      -8.705943582
## 356      -8.662065979
## 357      -0.032972253
## 358      -0.246685955
## 359      -0.947274298
## 360       0.083284433
## 361      -2.152956414
## 362      -1.172640550
## 363      -4.072837965
## 364      -3.845118340
## 365      -7.628079798
## 366      -0.289489481
## 367       0.646850073
## 368       1.210317006
## 369      18.543320014
## 370      18.989843210
## 371      18.258305208
## 372      24.500129462
## 373      23.882597382
## 374      12.279375151
## 375      15.319533083
## 376      -6.785177565
## 377       1.425306102
## 378      -1.074792606
```

```
## 379     1.052828311
## 380    -3.661765955
## 381    -7.803491501
## 382    -3.626800502
## 383    -0.832676131
## 384     1.079371249
## 385     3.346170826
## 386     1.917179710
## 387     2.813554845
## 388     3.238237947
## 389     4.736670333
## 390    -3.245311854
## 391    -3.198496437
## 392     6.469084997
## 393    -0.456573475
## 394    -6.341592183
## 395    -6.320533945
## 396    -5.188995943
## 397    -3.651384897
## 398    -7.128857753
## 399    -0.491831148
## 400     0.219138253
## 401    -3.521019679
## 402    -8.048838011
## 403    -3.158338505
## 404    -7.471365156
## 405    -0.041489573
## 406    -7.721706730
## 407    -0.479688963
## 408     4.870257782
## 409     7.727462060
## 410    11.738135338
## 411    -9.948841913
## 412     2.806206407
## 413    15.999355409
## 414     0.823150144
## 415     7.578984223
## 416     0.245092847
## 417    -2.552068046
## 418     1.155473905
## 419    -6.163823205
## 420    -4.549718575
## 421    -3.584099586
## 422    -5.438066025
## 423    -0.358144991
## 424     0.972808570
## 425    -6.550993969
## 426    -3.082137141
```

```
## 427     -9.447566519
## 428     -9.859124263
## 429     -3.108778787
## 430     -2.176652441
## 431     -3.294970279
## 432     -1.747369104
## 433     -7.024747154
## 434     -4.844040361
## 435     -8.441592183
## 436      0.953807583
## 437     -7.805450044
## 438     -0.725035472
## 439      6.166838135
## 440     -0.016711665
## 441     -3.048249668
## 442      1.091122506
## 443     -0.392522101
## 444     -1.245410561
## 445     -1.152166753
## 446      0.028342624
## 447     -2.752462876
## 448     -6.335029504
## 449     -3.229446096
## 450     -3.208387858
## 451     -4.584980150
## 452     -2.509465837
## 453     -2.046488540
## 454     -0.850014697
## 455     -1.878417471
## 456     -3.229446096
## 457     -3.793402664
## 458     -4.960004827
## 459     -4.234539868
## 460     -0.588115379
## 461     -2.554030491
## 462     -2.935617847
## 463     -1.762650420
## 464     -4.577833029
## 465     -0.594188423
## 466     -1.229643511
## 467      0.739505538
## 468      4.801211343
## 469      1.770553904
## 470     -0.431112418
## 471      0.822463093
## 472     -2.726705697
## 473      2.288867841
## 474      6.323734585
```

```
## 475     -3.519945602
## 476      1.642348546
## 477     -0.106918951
## 478      1.111888523
## 479     -2.824451031
## 480     -0.698693852
## 481     -1.350310820
## 482     -3.500458881
## 483     -2.893994910
## 484     -2.854326613
## 485     -1.280182500
## 486     -3.302318717
## 487     -1.222101560
## 488     -3.075775779
## 489     -2.195949551
## 490     -4.781157870
## 491      1.743623940
## 492     -3.786449057
## 493     -1.770682007
## 494     -1.343748141
## 495      2.857329838
## 496      5.267027747
## 497      5.230202459
## 498     -2.858144991
## 499     -1.079203229
## 500     -2.708095638
## 501     -4.139633640
## 502     -2.966863629
## 503     -5.327392747
## 504     -5.295562524
## 505     -6.397521067
## 506    -15.167451972
```

```r
# We can also extract the coefficients super easily
coef(lm_fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

```r
# and the confidence intervals!
confint(lm_fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

```r
# most importantly, we can PREDICT stuff very, very easily with this fitted model.
# The predict() function can be used to produce confidence intervals and
# prediction intervals for the prediction of medv for a given value of lstat.

predict(lm_fit, data.frame(lstat = (c(5, 10, 15))),
        interval = "confidence")
```

```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

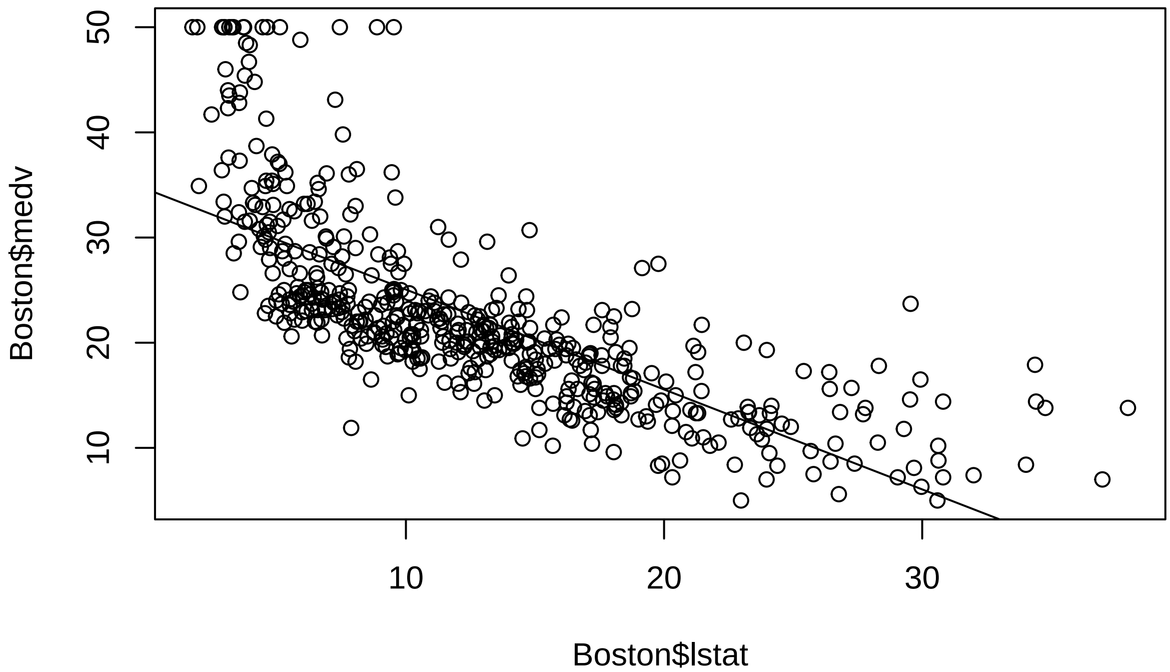
```r
predict(lm_fit, data.frame(lstat = (c(5, 10, 15))),
        interval = "prediction")
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

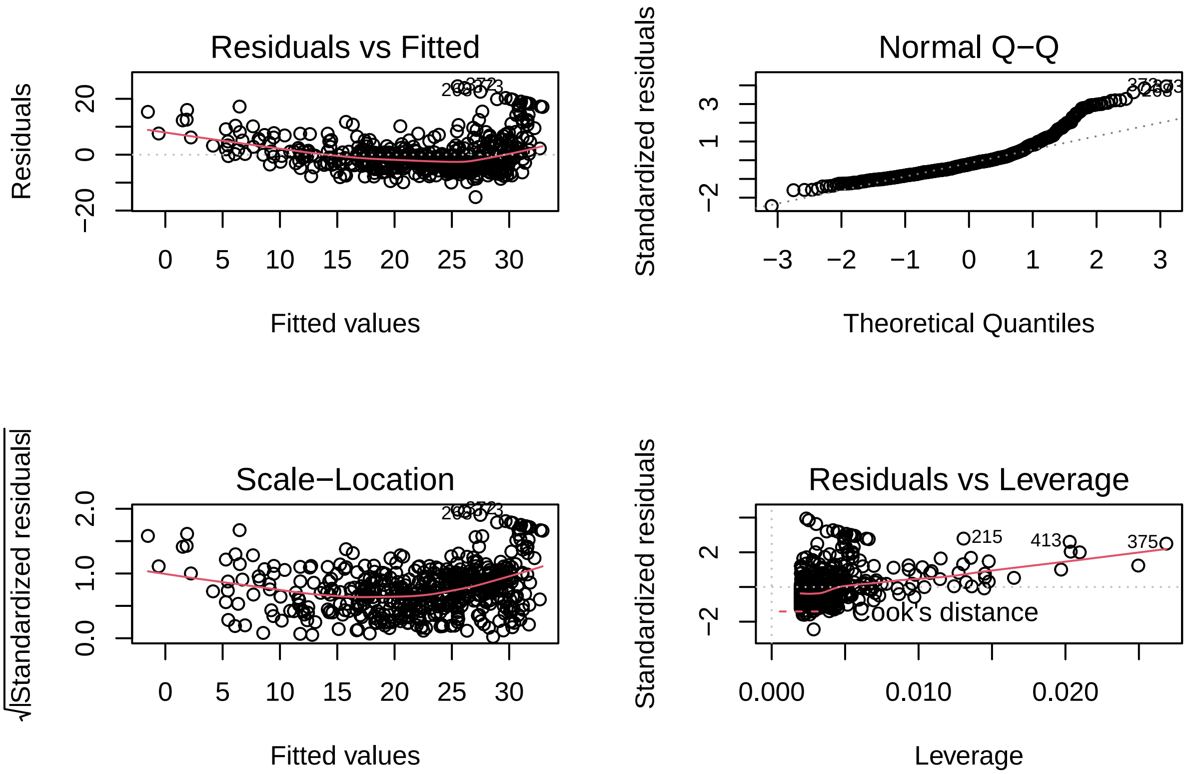
```r
# # ----------------------------------------------------------------
#
# STOP! Your turn. Predict the value of medv for a given lstat value of 8.
# Explain the difference
#
# # ----------------------------------------------------------------

predict(lm_fit, data.frame(lstat = c(8)))
```

```
##        1
## 26.95345
```

```r
# Let's plot these two variables and add the fitted line! We won't use ggplot2 here at the mom
# as standard plotting will work okay.

plot(Boston$lstat, Boston$medv)
abline(lm_fit, col =)
```




```r
# Let's take a peek at some diagnostic plots.
# Four diagnostic plots are automatically produced by applying the plot() function directly to
# command will produce one plot at a time, and hitting Enter will generate
# the next plot. However, it is often convenient to view all four plots together.
# We can achieve this in base R graphics by using the par() and mfrow() functions, which tell R
# to split the display screen into separate panels so that multiple plots can
# be viewed simultaneously. For example, par(mfrow = c(2, 2)) divides the
# plotting region into a 2 × 2 grid of panels.
par(mfrow = c(2, 2))
plot(lm_fit)
```




```
# What are these diagnostics?

#### Residuals vs. Fitted
# This plot shows if residuals have non-linear patterns.
# There could be a non-linear relationship between predictor variables and an outcome variable
# show up in this plot if the model doesn't capture the non-linear relationship. If you find e
# around a horizontal line without distinct patterns, that is a good indication you don't have

#### Normal Q-Q
# This plot shows if residuals are normally distributed. Do residuals follow a straight line w
# It's good if residuals are lined well on the straight dashed line.

#### Scale-Location
# It's also called Spread-Location plot. This plot shows if residuals are spread equally along
# This is how you can check the assumption of equal variance (homoscedasticity). It's good if
# (randomly) spread points.

#### Residuals vs. Leverage
# This plot helps us to find influential cases (i.e., subjects) if any. Not all outliers are i
# (whatever outliers mean). Even though data have extreme values, they might not be influentia
# the results wouldn't be much different if we either include or exclude them from analysis. T
# and they don't really matter; they are not influential. On the other hand, some cases could
# within a reasonable range of the values. They could be extreme cases against a regression li
#them from analysis. Another way to put it is that they don't get along with the trend in the
```

```r
# Unlike the other plots, this time patterns are not relevant. We watch out for outlying values
# right corner. Those spots are the places where cases can be influential against a regression
#Cook’s distance. When cases are outside of the Cook’s distance (meaning they have high Cook’s
# the regression results. The regression results will be altered if we exclude those cases.

# Let's keep going, so let's turn the plotting function off!
dev.off()
```

```
## null device
##           1
```

```r
# # ----------------------------------------------------------------
#
# Multiple Linear Regression
#
# # ----------------------------------------------------------------

# Let's add age!
lm_fit_multiple <- lm(medv ~ lstat + age, data = Boston)
summary(lm_fit_multiple)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

```r
# Heck, let's add EVERYTHING!
lm_fit_multiple <- lm(medv ~ ., data = Boston)
summary(lm_fit_multiple)
```

```
##
```

```
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1304  -2.7673  -0.5814   1.9414  26.2526
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.617270   4.936039   8.431 3.79e-16 ***
## crim         -0.121389   0.033000  -3.678 0.000261 ***
## zn            0.046963   0.013879   3.384 0.000772 ***
## indus         0.013468   0.062145   0.217 0.828520
## chas          2.839993   0.870007   3.264 0.001173 **
## nox         -18.758022   3.851355  -4.870 1.50e-06 ***
## rm            3.658119   0.420246   8.705  < 2e-16 ***
## age           0.003611   0.013329   0.271 0.786595
## dis          -1.490754   0.201623  -7.394 6.17e-13 ***
## rad           0.289405   0.066908   4.325 1.84e-05 ***
## tax          -0.012682   0.003801  -3.337 0.000912 ***
## ptratio      -0.937533   0.132206  -7.091 4.63e-12 ***
## lstat        -0.552019   0.050659 -10.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.798 on 493 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7278
## F-statistic: 113.5 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
# is this better than our simple model?
summary(lm_fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
# We can test if it is SIGNIFICANTLY better with an F-test!
# Use the anova function, and use the form anova(simpler_model, more complicated_model).
# Order matters!
anova(lm_fit, lm_fit_multiple)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + lstat
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 19472
## 2    493 11349 11      8123 32.077 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# We can check if our variables are highly collinear by using the vif()
# command from the car package.

vif(lm_fit_multiple)
```

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.767486 2.298459 3.987181 1.071168 4.369093 1.912532 3.088232 3.954037
##      rad      tax  ptratio    lstat
## 7.445301 9.002158 1.797060 2.870777
```

```
# A rule of thumb commonly used in practice is if a VIF is > 10, you have high multicollineari
# In our case, with values around 1-3, we are in good shape, and can proceed with our regressi
```

```
# # ----------------------------------------------------------------
#
# STOP! Your turn. Generate a model that predicts medv on
# the variables crim, age, and tax. Report the Adjusted R-Squared for
# this model.
#
# # ----------------------------------------------------------------

lm_fit_multiple_2 <- lm(medv ~ crim + age + tax, data = Boston)
summary(lm_fit)
```

```
##
```

```
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
# # ----------------------------------------------------------------
#
# Interaction terms and non-linear transformations!
#
# # ----------------------------------------------------------------

# Perhaps we sometimes want to add in the interaction effects between variables.
# You can specify an interaction term by using either the * or : character between variables.
# : generates JUST the interaction, whereas * generates the whole list (variables + interactio

summary(lm(medv ~ lstat * age, data = Boston))
```

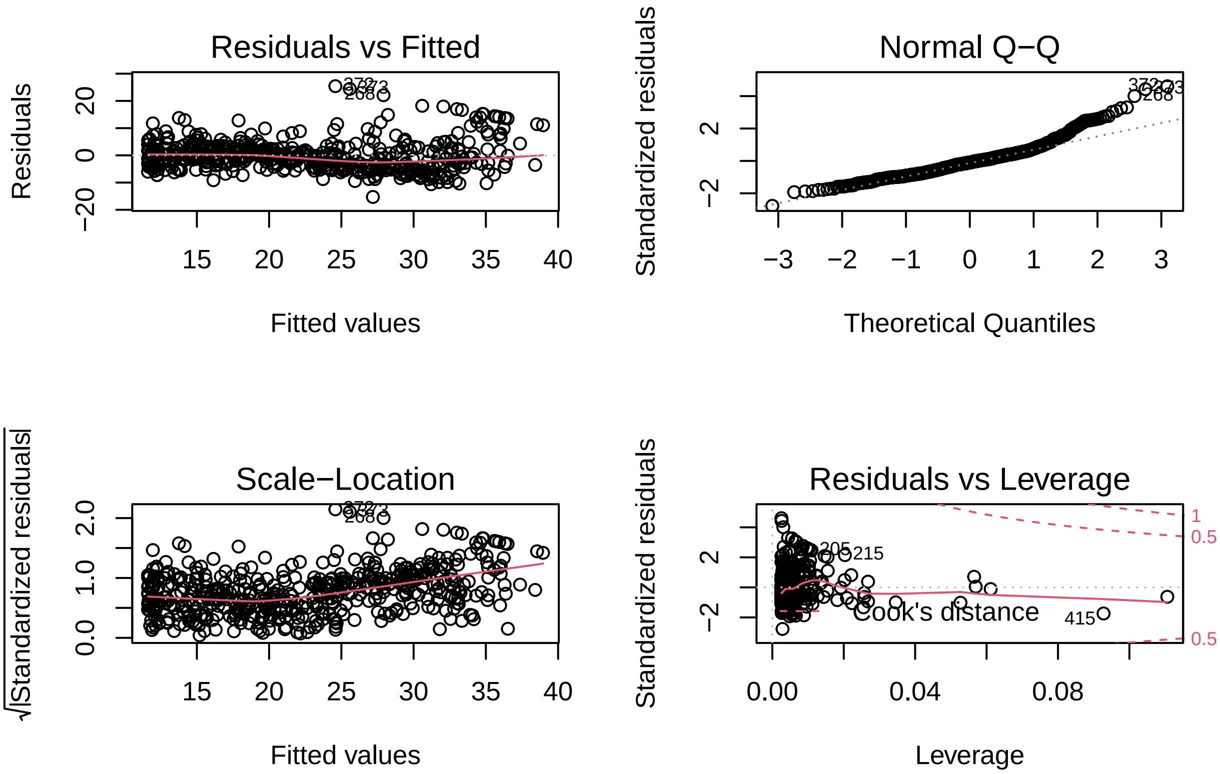
```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
# The lm() function can also accommodate non-linear transformations of the
# predictors even though it's building a linear model. For instance, given a predictor X,
# we can (for example) create a predictor X^2 using I(X^2).
# The function I() is needed since the ^ has a special meaning I() in a formula object;
# wrapping as we do allows the standard usage in R,
# which is to raise X to the power 2.

# We now perform a regression of medv onto lstat and lstat2.
lm_fit_fancy <- lm(medv ~ lstat + I(lstat^2), data=Boston)
summary(lm_fit_fancy)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
# # ----------------------------------------------------------------
#
# STOP! Your turn. Plot the diagnostics for the model with the quadratic term.
# Discuss what's different about them.
#
# # ----------------------------------------------------------------
#

par(mfrow = c(2, 2))
plot(lm_fit_fancy)
```




```r
# We can add an arbitrary number of polynomial variables. Let's add a 5th order polynomial!
lm_fit5 <- lm(medv ~ poly(lstat, 5), data=Boston)
summary(lm_fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
```

```
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

```r
# We can also take the log of a given variable. If you know me, you know I love logs.
summary(lm(medv ~ log(rm), data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
# # -----------------------------------------------------------------
#
# Qualitative Predictors
#
# # -----------------------------------------------------------------

#We're going to switch to a different data set now - one about carseats!
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

```r
# Let's built a model that fits to every varaible, the interaction effects between Income
# and Advertising, and the interaction effects between Price and Age.
careseats_fit <- lm(Sales ~ . + Income:Advertising + Price:Age,
             data = Carseats)
summary(careseats_fit)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9208 -0.7503  0.0177  0.6754  3.3413
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         6.5755654  1.0087470   6.519 2.22e-10 ***
## CompPrice           0.0929371  0.0041183  22.567  < 2e-16 ***
## Income              0.0108940  0.0026044   4.183 3.57e-05 ***
## Advertising         0.0702462  0.0226091   3.107 0.002030 **
## Population          0.0001592  0.0003679   0.433 0.665330
## Price              -0.1008064  0.0074399 -13.549  < 2e-16 ***
## ShelveLocGood       4.8486762  0.1528378  31.724  < 2e-16 ***
## ShelveLocMedium     1.9532620  0.1257682  15.531  < 2e-16 ***
## Age                -0.0579466  0.0159506  -3.633 0.000318 ***
## Education          -0.0208525  0.0196131  -1.063 0.288361
## UrbanYes            0.1401597  0.1124019   1.247 0.213171
## USYes              -0.1575571  0.1489234  -1.058 0.290729
## Income:Advertising  0.0007510  0.0002784   2.698 0.007290 **
## Price:Age           0.0001068  0.0001333   0.801 0.423812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 386 degrees of freedom
## Multiple R-squared:  0.8761, Adjusted R-squared:  0.8719
## F-statistic:   210 on 13 and 386 DF,  p-value: < 2.2e-16
```

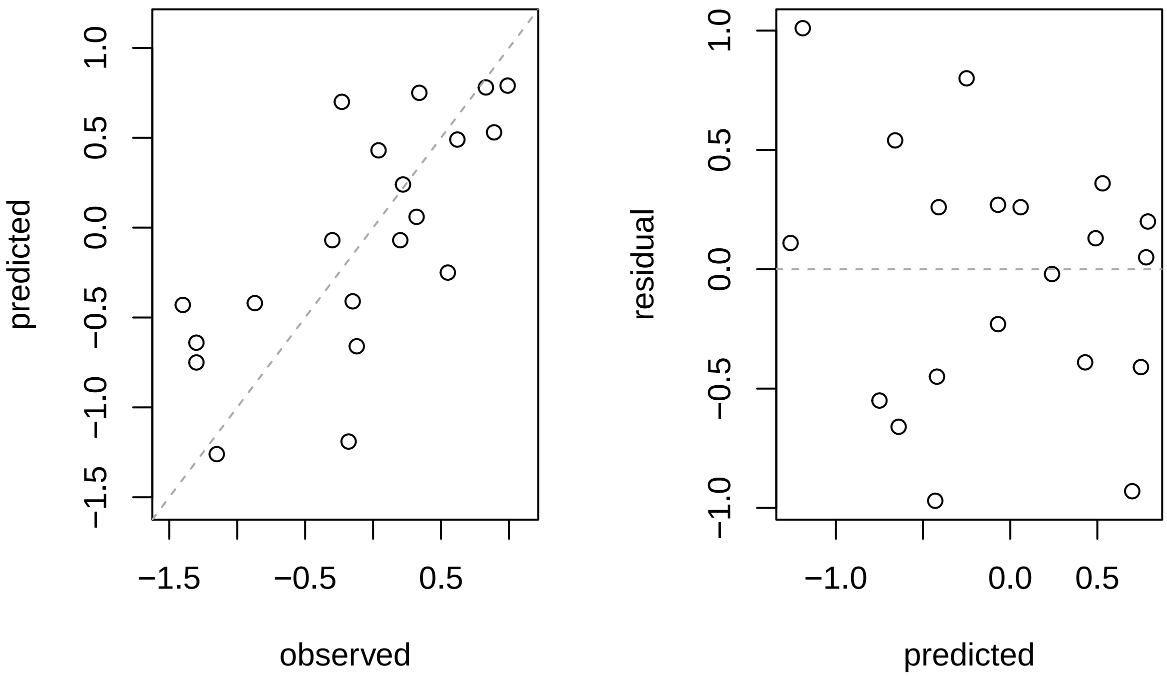
```r
# Notice how each "level" of the variable for ShelveLoc is present except for one?
# R has created a ShelveLocGood dummy variable that takes on a value of
# 1 if the shelving location is good, and 0 otherwise. It has also created a
# ShelveLocMedium dummy variable that equals 1 if the shelving location is
# medium, and 0 otherwise. A bad shelving location corresponds to a zero
# for each of the two dummy variables. The fact that the coefficient for
# ShelveLocGood in the regression output is positive indicates that a good
# shelving location is associated with high sales (relative to a bad location).
# And ShelveLocMedium has a smaller positive coefficient, indicating that a
# medium shelving location is associated with higher sales than a bad shelving location
```

```
# but lower sales than a good shelving location.

# You can see that by checking this out:
contrasts(Carseats$ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

```
# # --------------------------------------------------------------
#
# Let's take a look at manually predicting things and calculating
# RMSE
#
# # --------------------------------------------------------------

# Let's make up some pretend observations:

observed <- c(0.22, 0.83, -0.12, 0.89, -0.23, -1.30, -0.15, -1.4,
              + 0.62, 0.99, -0.18, 0.32, 0.34, -0.30, 0.04, -0.87,
              + 0.55, -1.30, -1.15, 0.20)

# Now, some pretend predictions:

predicted <- c(0.24, 0.78, -0.66, 0.53, 0.70, -0.75, -0.41, -0.43,
               + 0.49, 0.79, -1.19, 0.06, 0.75, -0.07, 0.43, -0.42,
               + -0.25, -0.64, -1.26, -0.07)

# Calculate the residuals:

residuals <- observed - predicted

# An important step in evaluating the quality of the model is to visualize
# the results. First, a plot of the observed values against the predicted values
# helps one to understand how well the model fits. Also, a plot of the residuals
# versus the predicted values can help uncover systematic patterns in the model
# predictions. Let's make some plots!

# this puts things on the same axis
axis_range <- extendrange(c(observed, predicted))
par(mfrow = c(1,2))
# show Predicted vs. Observed
plot(observed, predicted,
     ylim = axis_range,
     xlim = axis_range,
     ylab = "predicted",
```

```
     xlab = "observed")
# Add a 45 degree reference line
abline(0, 1, col = "darkgrey", lty = 2)
# show predicted values versus residuals
plot(predicted, residuals, ylab = "residual", xlab = "predicted")
abline(h = 0, col = "darkgrey", lty = 2)
```




```
# turn off the par plot
dev.off()
```

```
## null device
##           1
```

```
# Calculate R2
caret::R2(predicted, observed)
```

```
## [1] 0.5170123
```

```
# Calculate RMSE
caret::RMSE(predicted, observed)
```

```
## [1] 0.5234883
```