Rollercoaster

Gao

2023-10-30

Import libraries or something

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
## v parsnip      1.1.1     v yardstick    1.2.0
## v recipes      1.0.8
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(ggforce)
library(yardstick)
library(car)
```

```
## Loading required package: carData
##
```

```
## Attaching package: ’car’
##
## The following object is masked from ’package:dplyr’:
##
##     recode
##
## The following object is masked from ’package:purrr’:
##
##     some
```

```r
library(moments)
library(GGally)
```

```
## Registered S3 method overwritten by ’GGally’:
##   method from
##   +.gg   ggplot2
```

```r
library(psych)
```

```
##
## Attaching package: ’psych’
##
## The following object is masked from ’package:car’:
##
##     logit
##
## The following objects are masked from ’package:scales’:
##
##     alpha, rescale
##
## The following objects are masked from ’package:ggplot2’:
##
##     %+%, alpha
```

```r
library(fastDummies)
```

```
## Thank you for using fastDummies!
## To acknowledge our work, please cite the package:
## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from
```

Import the data

```r
rollercoasters <- read_csv("rollercoasters.csv") %>% as_tibble()
```

```
## Rows: 101 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Name, Park, Track
## dbl (6): Speed, Height, Drop, Length, Duration, Inversions
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Find correlation between Speed and all numerical values

```
cor(rollercoasters$Speed, select_if(rollercoasters, is.numeric))
```

```
##      Speed    Height      Drop    Length  Duration Inversions
## [1,]     1 0.9170502 0.9543598 0.4562342 0.1704023 -0.4037699
```

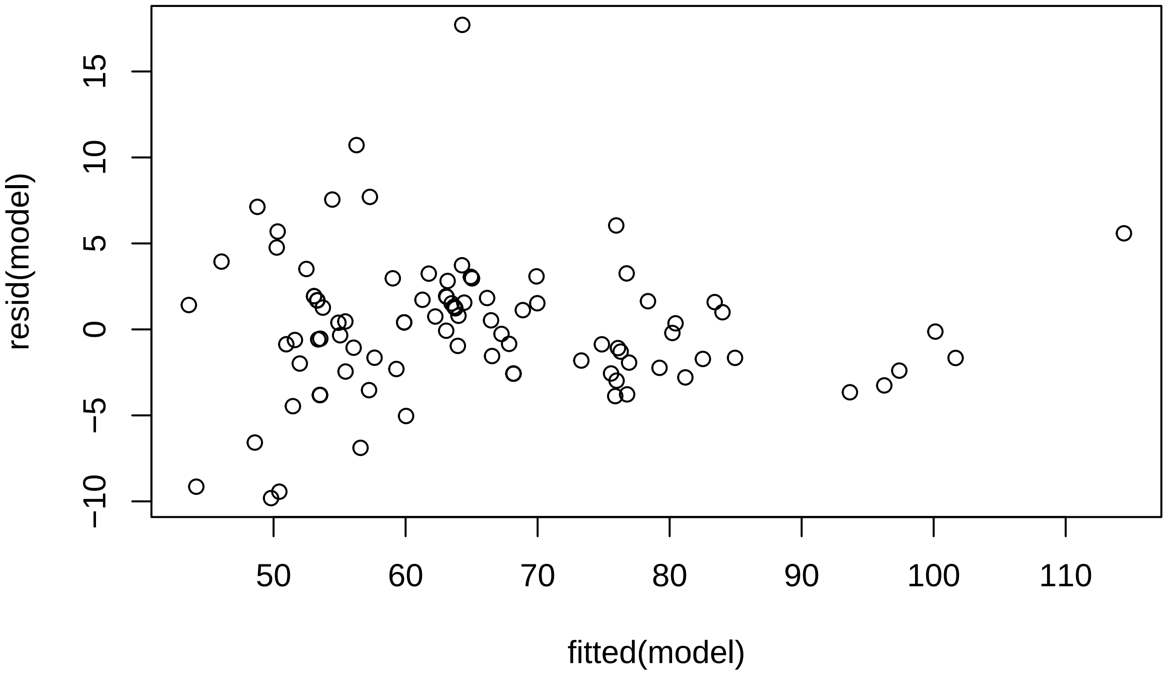
Find the regression model

```
model <- lm(Speed ~ Height + Drop + Length + Duration + Inversions, data = rollercoasters)
summary(model)
```

```
##
## Call:
## lm(formula = Speed ~ Height + Drop + Length + Duration + Inversions,
##     data = rollercoasters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8097 -2.4500 -0.1268  1.6911 17.7084
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 35.7956667  1.7295008  20.697  < 2e-16 ***
## Height       0.0406960  0.0161666   2.517   0.0135 *
## Drop         0.1468456  0.0178691   8.218 1.04e-12 ***
## Length       0.0013097  0.0005315   2.464   0.0155 *
## Duration    -0.0293653  0.0159234  -1.844   0.0683 .
## Inversions  -0.2885555  1.0018872  -0.288   0.7740
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 4.034 on 95 degrees of freedom
## Multiple R-squared:  0.9228, Adjusted R-squared:  0.9187
## F-statistic:   227 on 5 and 95 DF,  p-value: < 2.2e-16
```

Construct the residual plot

```
plot(fitted(model), resid(model))
```



Draw outlier boundaries

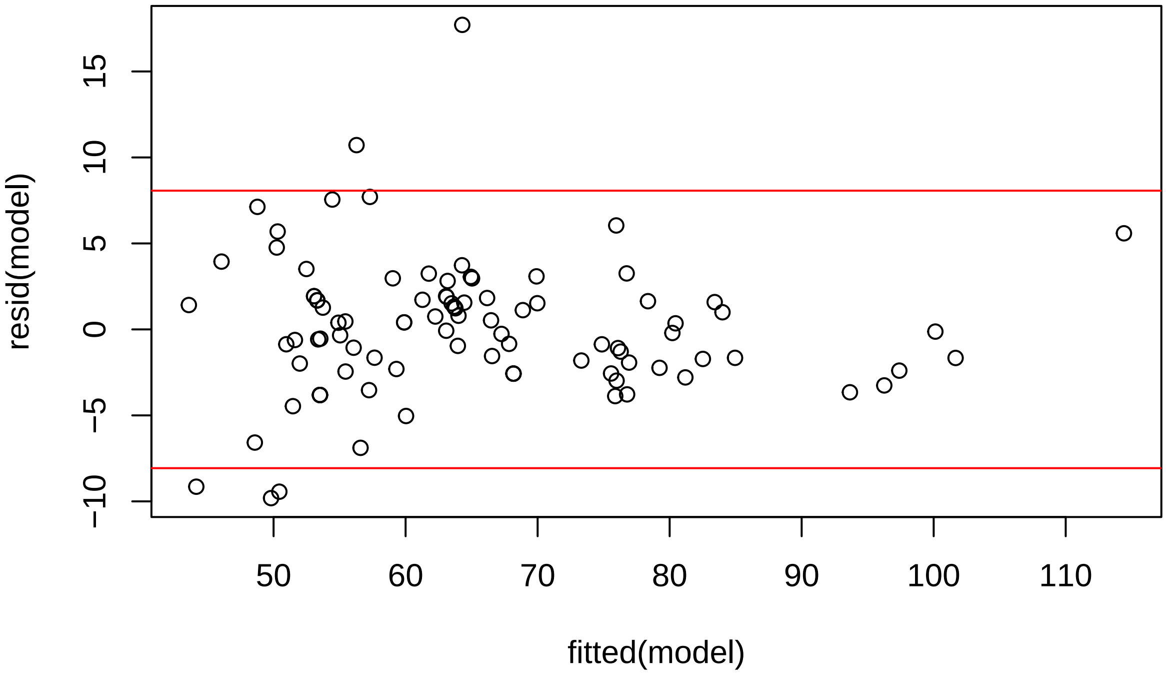
```
standard_error <- sqrt(deviance(model)/df.residual(model))
standard_error
```

```
## [1] 4.034104
```

```
2*standard_error
```

```
## [1] 8.068208
```

```
plot(fitted(model),resid(model))
abline(h=2*standard_error, col = "red")
abline(h=-2*standard_error, col = "red")
```



2x standard error of mean: 8.068208 There are 5 outliers that are outside of 2 standard errors of the mean. (Coasters 19, 2, 16, 43, 3) Display residual values

```
residuals <- resid(model)
sort(residuals)
```

```
##          19          2          16          17          24           9
## -9.80968692 -9.43582323 -9.14564310 -6.88655153 -6.57873045 -5.03502709
##           6          29          36          37          38          39
## -4.46403599 -3.87797696 -3.81426290 -3.81426290 -3.81426290 -3.81426290
##          40          41          59          80          11          67
## -3.81426290 -3.81426290 -3.77848582 -3.65532739 -3.53095820 -3.25728330
##          45          99          13          14          15          12
## -2.98118649 -2.79099941 -2.57140433 -2.57140433 -2.57140433 -2.56733473
##          84          42         101          89          61          20
## -2.56308696 -2.45003243 -2.39628114 -2.30258696 -2.23405392 -1.98531321
##          86          35          87          92          79          60
## -1.92910179 -1.81035928 -1.71792525 -1.65979658 -1.65366258 -1.64575621
##          82          75          76          28          57          74
## -1.54901403 -1.29024392 -1.08676379 -1.06388347 -0.95705821 -0.86666831
##          68          81          10          27          52          97
## -0.86365343 -0.84069790 -0.61621282 -0.57933503 -0.54028131 -0.34150390
##          64          65           1          70          96         100
## -0.26324045 -0.20740556 -0.12677727 -0.07609965  0.35354490  0.38856557
##          21          22          66          62          69          90
##  0.40898972  0.40898972  0.46269561  0.52785803  0.74986959  0.79463263
##          85          58          56           8          71          49
##  0.99778771  1.12009791  1.22804365  1.26706370  1.26723237  1.32461850
##           4          72          73          48          78          77
##  1.41862094  1.51296753  1.52178776  1.55878547  1.58726310  1.64038326
##          31          32          33          34          23          47
##  1.69113548  1.69113548  1.69113548  1.69113548  1.72204787  1.82570548
##          44          83          25          26          46          63
##  1.89005657  1.93160418  1.93289874  1.93289874  2.81728320  2.96707347
##          55           7          50          88          91          30
##  2.97044418  3.05763232  3.08368177  3.24381327  3.25561435  3.51387496
##          94          18           5          93          51          98
##  3.72883602  3.94274100  4.76066630  5.58758210  5.69255297  6.04419386
##          95          53          54          43           3
##  7.12571531  7.55162527  7.70435850 10.71597688 17.70841942
```

Cook's Distance

Jackknife

```
jac <- qt(df= 101 - 5 - 2, 0.95)
jac
```

```
## [1] 1.661226
```

```
sort(jackknife <- rstudent(model))
```

```
##          19           2          16          17          24           9
## -2.57052918 -2.52245096 -2.43993818 -1.77919507 -1.69086432 -1.28471954
##           6          29          36          37          38          39
## -1.14891029 -0.98813891 -0.95610805 -0.95610805 -0.95610805 -0.95610805
##          40          41          59          80          11          67
## -0.95610805 -0.95610805 -0.94932547 -0.93775067 -0.89029940 -0.84329770
##          45          99          13          14          15          12
## -0.76252985 -0.74778427 -0.65357322 -0.65357322 -0.65357322 -0.65253416
##          84         101          42          89          61          20
## -0.64584669 -0.62994342 -0.61819026 -0.58790073 -0.56226222 -0.50327540
##          86          92          35          87          79          60
## -0.49250273 -0.47683253 -0.45935716 -0.43919044 -0.41715334 -0.41233994
##          82          75          76          28          57          68
## -0.38895493 -0.32296354 -0.27207075 -0.26759471 -0.24004070 -0.22607518
##          74          81          10          27          52          97
## -0.21685351 -0.21344133 -0.15479042 -0.14492978 -0.13603176 -0.08524739
##          64          65           1          70          96         100
## -0.06597490 -0.05237858 -0.03491871 -0.01909104  0.09003219  0.09702678
##          21          22          66          62          69          90
##  0.10286584  0.10286584  0.11696652  0.13591656  0.18817305  0.20759802
##          85          58          56          71           8          49
##  0.25489372  0.28053277  0.30776545  0.31822233  0.32301855  0.34188350
##           4          73          72          48          78          77
##  0.35999469  0.38084818  0.38481067  0.39147570  0.40459346  0.41135923
##          31          32          33          34          23          47
##  0.42325485  0.42325485  0.42325485  0.42325485  0.43209228  0.45753479
##          25          26          44          83          46          63
##  0.48380471  0.48380471  0.48566289  0.48916676  0.71809364  0.74520295
##          55          50           7          88          91          30
##  0.74607235  0.77502879  0.81556456  0.83499983  0.85298574  0.88162447
##          18          94           5          51          93          98
##  1.00901671  1.11794151  1.21248390  1.46132284  1.60427800  1.61927363
##          95          53          54          43           3
##  1.86068145  1.94145663  1.96255671  2.79804194  5.32771209
```

Jackknife = 1.661226 Outliers: 19, 2, 16, 17, 24, 95, 53, 54, 43, 3

Leverage

```
lev <- 2 * (1+5)/101
lev
```

```
## [1] 0.1188119
```

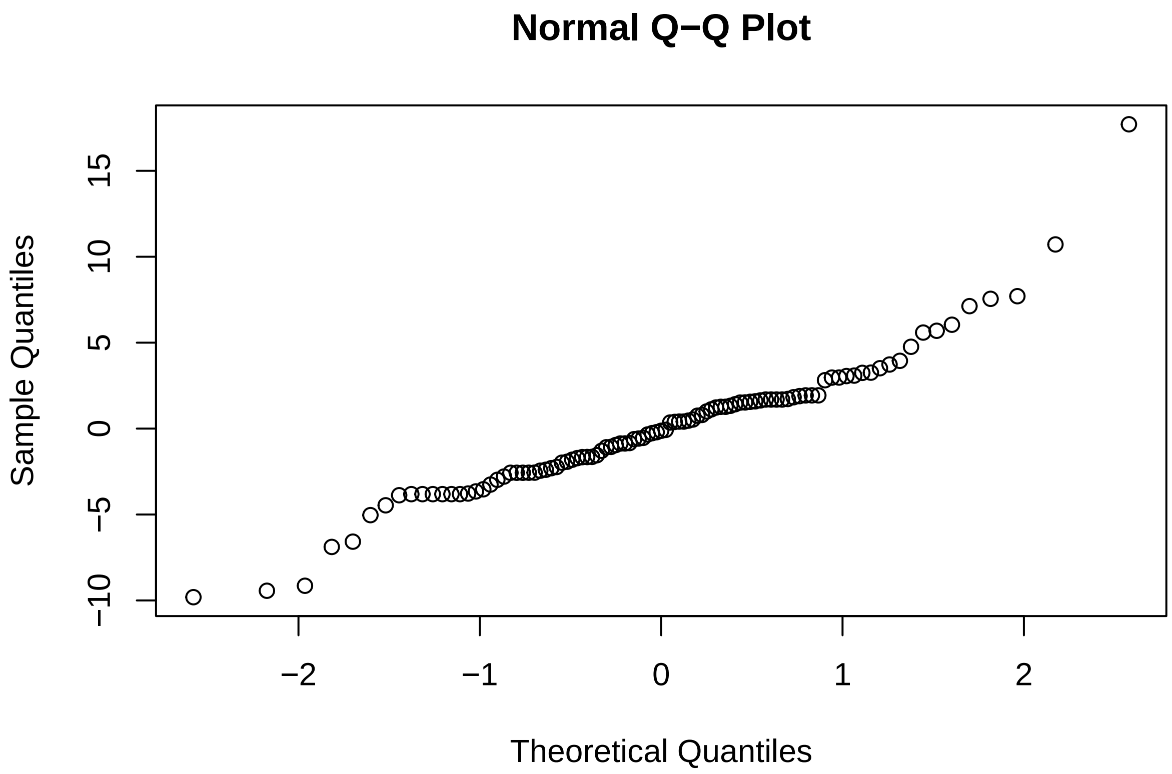
```
sort(hatvalues(model))
```

```
##         36         37         38         39         40         41         97
## 0.02294170 0.02294170 0.02294170 0.02294170 0.02294170 0.02294170 0.02417290
##         54        100         30         25         26         31         32
## 0.02460875 0.02478805 0.02615072 0.02709997 0.02709997 0.02749932 0.02749932
##         33         34         59         73         27         74         75
## 0.02749932 0.02749932 0.02756770 0.02773617 0.02825684 0.02837150 0.02853224
##         76         60         47         58         63         55         56
## 0.02913558 0.02967965 0.02973714 0.02989593 0.03043768 0.03048370 0.03097406
##         50         77         64         23         57         70         43
## 0.03132095 0.03141134 0.03199412 0.03237011 0.03287375 0.03390946 0.03392911
##         69         82         48         71         11         10         61
## 0.03410409 0.03412559 0.03443687 0.03477419 0.03557196 0.03618410 0.03688585
##         84         28         21         22         52         42         79
## 0.03816450 0.03822410 0.03873882 0.03873882 0.04069913 0.04110397 0.04277133
##         53         65          5         66         83          9         20
## 0.04322453 0.04664171 0.04800815 0.04842953 0.04953390 0.04970929 0.05130761
##         24         19         35         29          4         12         13
## 0.05160556 0.05228370 0.05351776 0.05382278 0.05452730 0.05456554 0.05456555
##         14         15         81         51         17         72         46
## 0.05456555 0.05456555 0.05627943 0.05640049 0.05843355 0.05863353 0.05901029
##         18         96         78          8         89         86         45
## 0.06159903 0.06234935 0.06259798 0.06344368 0.06388735 0.06476160 0.06491126
##         80         85         87          6         95         88         44
## 0.06753644 0.06767307 0.06781281 0.06921494 0.07544861 0.07560322 0.07683728
##         62         67         49          2         16         91         90
## 0.08275575 0.08602860 0.08614950 0.09161704 0.09165624 0.10743247 0.10875816
##         68        101          3         98          7         99          1
## 0.11219125 0.11648868 0.12544559 0.12924741 0.13935076 0.14797409 0.19853929
##         93         92         94
## 0.24224338 0.26152245 0.31458327
```

leverage = 0.1188119 Outliers: 3, 98, 7, 99, 1, 93, 92, 94

QQPlot

```r
qqnorm(resid(model))
```



```r
model <- lm(Speed ~ Height + Drop + Length + Duration + Inversions, data = rollercoasters)
model
```

```
##
## Call:
## lm(formula = Speed ~ Height + Drop + Length + Duration + Inversions,
##     data = rollercoasters)
##
## Coefficients:
## (Intercept)       Height         Drop       Length     Duration   Inversions
##    35.79567      0.04070      0.14685      0.00131     -0.02937     -0.28856
```

Speed-hat = 0.0407 * Height + 0.14685 * Drop + 0.00131 * Length + -0.2937 * Duration + -0.28856 * Inversions + 35.79567

```r
rollercoasters <- dummy_cols(rollercoasters, select_columns = "Track", remove_first_dummy = TRUE)
rollercoasters
```

```
## # A tibble: 101 x 10
##    Name      Park  Track Speed Height  Drop Length Duration Inversions Track_Wood
##    <chr>     <chr> <chr> <dbl>  <dbl> <dbl>  <dbl>    <dbl>      <dbl>      <int>
##  1 Tower o~ Drea~ Steel   100   377.  328.   1235       28          0          0
##  2 Canyon ~ Adve~ Steel    41    94    66    2423       60          1          0
##  3 Xcelera~ Knot~ Steel    82   205   130    2202       62          0          0
##  4 Afterbu~ Fun ~ Steel    45    56    47     635       66          1          0
##  5 Silver ~ Fron~ Steel    55    83    75    1942       75          1          0
##  6 New Mex~ Clif~ Wood     47    80    75    2750       75          0          1
##  7 Outlaw ~ Silv~ Wood     68   107   162    2937       87          1          1
##  8 Thunder~ Kenn~ Wood     55    70    95    2887       90          0          1
##  9 Inverti~ Para~ Steel    55   138   138     985       90          1          0
## 10 Freesty~ Cava~ Steel    51    88    84    2210       92          1          0
## # i 91 more rows
```

```
model2 <- lm(Speed ~ Height + Drop + Length + Duration + Inversions + Track_Wood, data = rollercoasters)
summary(model2)
```

```
##
## Call:
## lm(formula = Speed ~ Height + Drop + Length + Duration + Inversions +
##     Track_Wood, data = rollercoasters)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1249  -2.3337  -0.2069   1.8148  18.1630
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.8044115  2.1521406  16.172  < 2e-16 ***
## Height       0.0423121  0.0163335   2.591   0.0111 *
## Drop         0.1490159  0.0181231   8.222 1.09e-12 ***
## Length       0.0011928  0.0005534   2.155   0.0337 *
## Duration    -0.0264514  0.0163916  -1.614   0.1099
## Inversions   0.1339013  1.1417891   0.117   0.9069
## Track_Wood   1.0172394  1.3093781   0.777   0.4392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.043 on 94 degrees of freedom
## Multiple R-squared:  0.9232, Adjusted R-squared:  0.9183
## F-statistic: 188.5 on 6 and 94 DF,  p-value: < 2.2e-16
```

```
model2 %>% tidy()
```

```
## # A tibble: 7 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 34.8      2.15        16.2   6.68e-29
## 2 Height       0.0423   0.0163       2.59  1.11e- 2
## 3 Drop         0.149    0.0181       8.22  1.09e-12
## 4 Length       0.00119  0.000553     2.16  3.37e- 2
## 5 Duration    -0.0265   0.0164      -1.61  1.10e- 1
## 6 Inversions   0.134    1.14         0.117 9.07e- 1
## 7 Track_Wood   1.02     1.31         0.777 4.39e- 1
```