Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure

Arnaud Legrand

23 September 2018

In this document we reperform some of the analysis provided in *Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure* by *Siddhartha R. Dalal, Edward B. Fowlkes, Bruce Hoadley* published in *Journal of the American Statistical Association*, Vol. 84, No. 408 (Dec., 1989), pp. 945-957 and available at http://www.jstor.org/stable/2290069.

On the fourth page of this article, they indicate that the maximum likelihood estimates of the logistic regression using only temperature are: $\hat{\alpha} = \mathbf{5.085}$ and $\hat{\beta} = \mathbf{-0.1156}$ and their asymptotic standard errors are $s_{\hat{\alpha}} = \mathbf{3.052}$ and $s_{\hat{\beta}} = \mathbf{0.047}$. The Goodness of fit indicated for this model was $G^2 = \mathbf{18.086}$ with **21** degrees of freedom. Our goal is to reproduce the computation behind these values and the Figure 4 of this article, possibly in a nicer looking way.

Technical information on the computer on which the analysis is run

We will be using the R language using the ggplot2 library.

```r
library(ggplot2)
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.18     withr_2.1.2      crayon_1.3.4     digest_0.6.16
##  [5] rprojroot_1.3-2  plyr_1.8.4       grid_3.5.1       gtable_0.2.0
##  [9] backports_1.1.2  magrittr_1.5     evaluate_0.11    scales_1.0.0
## [13] pillar_1.3.0     rlang_0.2.2      stringi_1.1.7    lazyeval_0.2.1
## [17] rmarkdown_1.10   tools_3.5.1      stringr_1.3.1    munsell_0.5.0
## [21] yaml_2.2.0       compiler_3.5.1   colorspace_1.3-2 htmltools_0.3.6
## [25] knitr_1.20       tibble_1.4.2
```

Here are the available libraries

```r
devtools::session_info()
```

```
## Session info -------------------------------------------------------------

##  setting  value
##  version  R version 3.5.1 (2018-07-02)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  French_France.1252
##  tz       Europe/Paris
##  date     2018-10-10

## Packages -----------------------------------------------------------------

##  package    * version date       source
##  backports    1.1.2   2017-12-13 CRAN (R 3.5.0)
##  base       * 3.5.1   2018-07-02 local
##  colorspace   1.3-2   2016-12-14 CRAN (R 3.5.1)
##  compiler     3.5.1   2018-07-02 local
##  crayon       1.3.4   2017-09-16 CRAN (R 3.5.1)
##  datasets   * 3.5.1   2018-07-02 local
##  devtools     1.13.6  2018-06-27 CRAN (R 3.5.1)
##  digest       0.6.16  2018-08-22 CRAN (R 3.5.1)
##  evaluate     0.11    2018-07-17 CRAN (R 3.5.1)
##  ggplot2    * 3.0.0   2018-07-03 CRAN (R 3.5.1)
##  graphics   * 3.5.1   2018-07-02 local
##  grDevices  * 3.5.1   2018-07-02 local
##  grid         3.5.1   2018-07-02 local
##  gtable       0.2.0   2016-02-26 CRAN (R 3.5.1)
##  htmltools    0.3.6   2017-04-28 CRAN (R 3.5.1)
##  knitr        1.20    2018-02-20 CRAN (R 3.5.1)
##  lazyeval     0.2.1   2017-10-29 CRAN (R 3.5.1)
##  magrittr     1.5     2014-11-22 CRAN (R 3.5.1)
##  memoise      1.1.0   2017-04-21 CRAN (R 3.5.1)
##  methods    * 3.5.1   2018-07-02 local
##  munsell      0.5.0   2018-06-12 CRAN (R 3.5.1)
##  pillar       1.3.0   2018-07-14 CRAN (R 3.5.1)
##  plyr         1.8.4   2016-06-08 CRAN (R 3.5.1)
##  Rcpp         0.12.18 2018-07-23 CRAN (R 3.5.1)
##  rlang        0.2.2   2018-08-16 CRAN (R 3.5.1)
##  rmarkdown    1.10    2018-06-11 CRAN (R 3.5.1)
##  rprojroot    1.3-2   2018-01-03 CRAN (R 3.5.1)
##  scales       1.0.0   2018-08-09 CRAN (R 3.5.1)
##  stats      * 3.5.1   2018-07-02 local
##  stringi      1.1.7   2018-03-12 CRAN (R 3.5.0)
##  stringr      1.3.1   2018-05-10 CRAN (R 3.5.1)
##  tibble       1.4.2   2018-01-22 CRAN (R 3.5.1)
##  tools        3.5.1   2018-07-02 local
##  utils      * 3.5.1   2018-07-02 local
##  withr        2.1.2   2018-03-15 CRAN (R 3.5.1)
##  yaml         2.2.0   2018-07-25 CRAN (R 3.5.1)
```

Loading and inspecting data

Let's start by reading data:

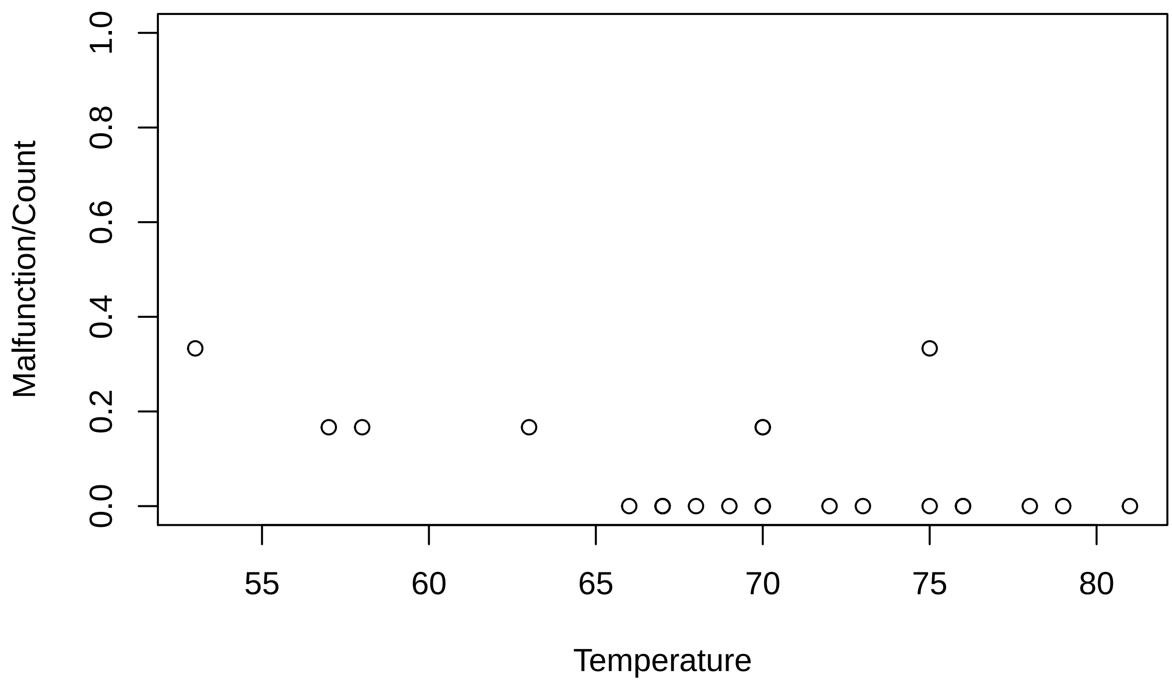
```
data = read.csv("https://app-learninglab.inria.fr/gitlab/moocrr-session1/moocrr-reproducibility-study/ra
data
```

```
##         Date Count Temperature Pressure Malfunction
## 1    4/12/81     6          66       50           0
## 2   11/12/81     6          70       50           1
## 3    3/22/82     6          69       50           0
## 4   11/11/82     6          68       50           0
## 5    4/04/83     6          67       50           0
## 6    6/18/82     6          72       50           0
## 7    8/30/83     6          73      100           0
## 8   11/28/83     6          70      100           0
## 9    2/03/84     6          57      200           1
## 10   4/06/84     6          63      200           1
## 11   8/30/84     6          70      200           1
## 12  10/05/84     6          78      200           0
## 13  11/08/84     6          67      200           0
## 14   1/24/85     6          53      200           2
## 15   4/12/85     6          67      200           0
## 16   4/29/85     6          75      200           0
## 17   6/17/85     6          70      200           0
## 18 7/2903/85     6          81      200           0
## 19   8/27/85     6          76      200           0
## 20  10/03/85     6          79      200           0
## 21  10/30/85     6          75      200           2
## 22  11/26/85     6          76      200           0
## 23   1/12/86     6          58      200           1
```

We know from our previous experience on this data set that filtering data is a really bad idea. We will therefore process it as such.

Let's visually inspect how temperature affects malfunction:

```
plot(data=data, Malfunction/Count ~ Temperature, ylim=c(0,1))
```



Logistic regression

Let's assume O-rings independently fail with the same probability which solely depends on temperature. A logistic regression should allow us to estimate the influence of temperature.

```
logistic_reg = glm(data=data, Malfunction/Count ~ Temperature, weights=Count,
                   family=binomial(link='logit'))
summary(logistic_reg)
```

```
##
## Call:
## glm(formula = Malfunction/Count ~ Temperature, family = binomial(link = "logit"),
##     data = data, weights = Count)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.95227  -0.78299  -0.54117  -0.04379   2.65152
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.08498    3.05247   1.666   0.0957 .
## Temperature -0.11560    0.04702  -2.458   0.0140 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

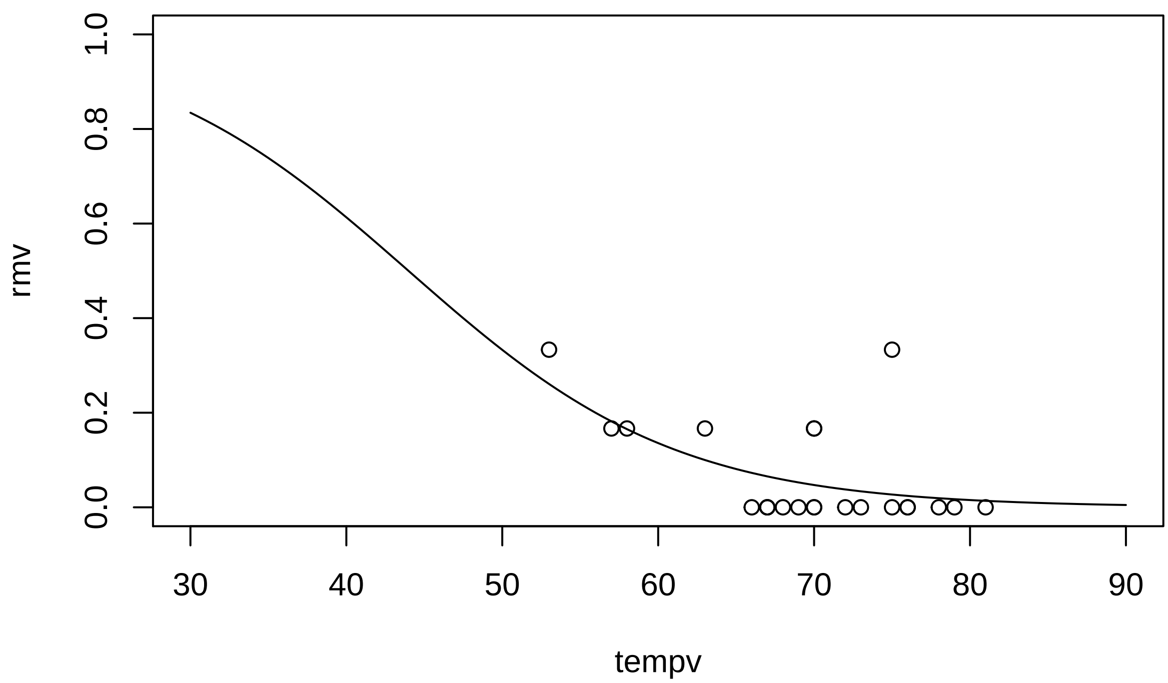
```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.230  on 22  degrees of freedom
## Residual deviance: 18.086  on 21  degrees of freedom
## AIC: 35.647
##
## Number of Fisher Scoring iterations: 5
```

The maximum likelyhood estimator of the intercept and of Temperature are thus $\hat{\alpha} = \mathbf{5.0850}$ and $\hat{\beta} = \mathbf{-0.1156}$ and their standard errors are $s_{\hat{\alpha}} = \mathbf{3.052}$ and $s_{\hat{\beta}} = \mathbf{0.04702}$. The Residual deviance corresponds to the Goodness of fit $G^2 = \mathbf{18.086}$ with **21** degrees of freedom. **I have therefore managed to replicate the results of the Dalal *et al.* article**.

Predicting failure probability

The temperature when launching the shuttle was 31°F. Let's try to estimate the failure probability for such temperature using our model :

```
# shuttle=shuttle[shuttle$r!=0,]
tempv = seq(from=30, to=90, by = .5)
rmv <- predict(logistic_reg,list(Temperature=tempv),type="response")
plot(tempv,rmv,type="l",ylim=c(0,1))
points(data=data, Malfunction/Count ~ Temperature)
```

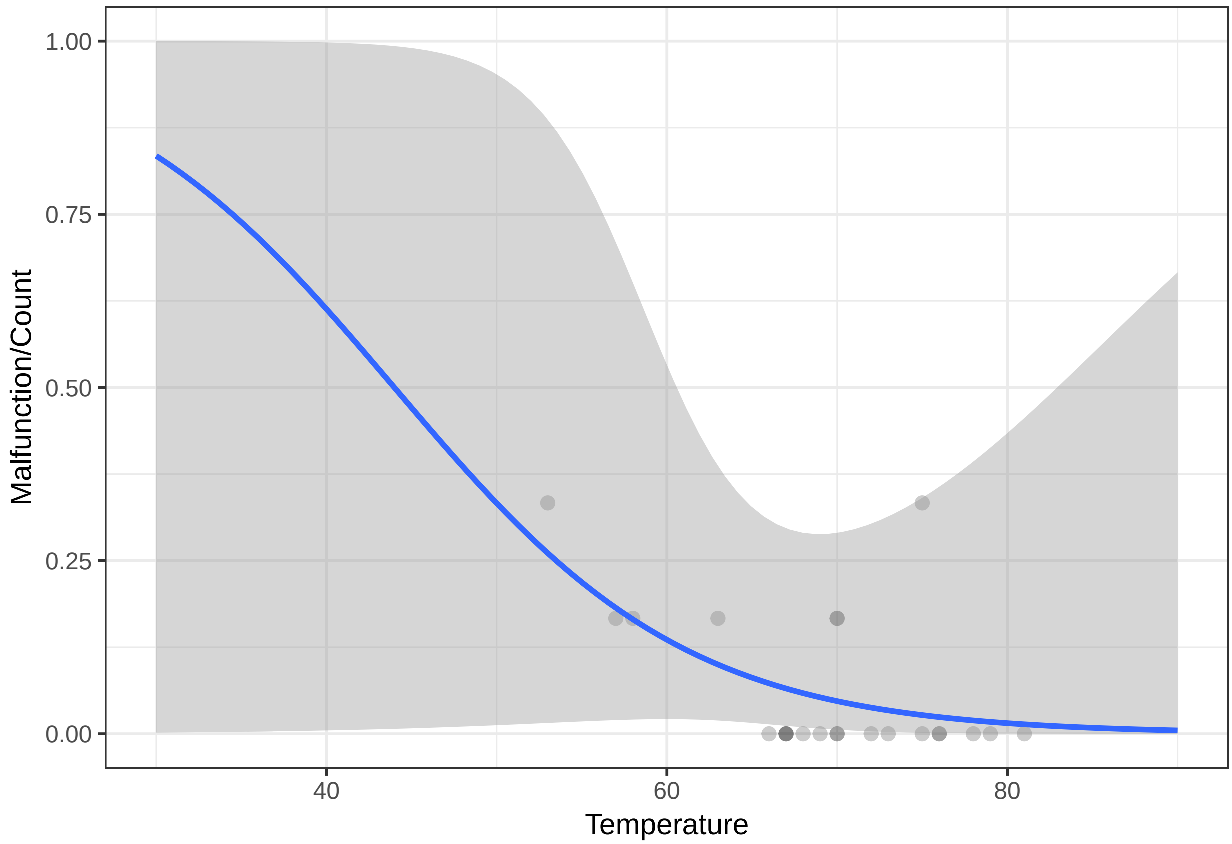


This figure is very similar to the Figure 4 of Dalal *et al.* **I have managed to replicate the Figure 4 of the Dalal *et al.* article.**

Let's try to plot confidence intervals although I am not sure exactly how they are computed.

```
ggplot(data, aes(y=Malfunction/Count, x=Temperature)) + geom_point(alpha=.2, size = 2) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"), fullrange=T) +
  xlim(30,90) + ylim(0,1) + theme_bw()
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial
## glm!
```



No confidence region was given in the original article. **Let's hope this confidence region estimation is correct.**