

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

Arnaud Legrand

25 October 2018 / new run by Agathe S on 12 April 2020

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:  $\hat{\alpha} = 5.085$  and  $\hat{\beta} = -0.1156$  and their asymptotic standard errors are  $s_{\hat{\alpha}} = 3.052$  and  $s_{\hat{\beta}} = 0.047$ . The Goodness of fit indicated for this model was  $G^2 = 18.086$  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.

AgatheS edit : missing devtools package installed (first use of R language)

```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
```

```
##   method      from
##   [.quosures   rlang
##   c.quosures   rlang
##   print.quosures rlang
```

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
```

```
## Platform: x86_64-conda_cos6-linux-gnu (64-bit)
```

```
## Running under: Ubuntu 18.04.4 LTS
```

```
##
```

```
## Matrix products: default
```

```
## BLAS/LAPACK: /home/aschmide/miniconda3/envs/mooc-rr-rstudio/lib/R/lib/libRblas.so
```

```
##
```

```
## locale:
```

```
## [1] LC_CTYPE=fr_FR.UTF-8      LC_NUMERIC=C
```

```
## [3] LC_TIME=fr_FR.UTF-8      LC_COLLATE=fr_FR.UTF-8
```

```
## [5] LC_MONETARY=fr_FR.UTF-8  LC_MESSAGES=fr_FR.UTF-8
```

```
## [7] LC_PAPER=fr_FR.UTF-8     LC_NAME=C
```

```
## [9] LC_ADDRESS=C             LC_TELEPHONE=C
```

```
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
```

```
##
```

```
## attached base packages:
```

```
## [1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
##
```

```
## other attached packages:
```

```
## [1] ggplot2_3.1.1
##
## loaded via a namespace (and not attached):
## [1] Rcpp_1.0.1      knitr_1.22      magrittr_1.5    tidyselect_0.2.5
## [5] munsell_0.5.0   colorspace_1.4-1 R6_2.4.0        rlang_0.4.5
## [9] stringr_1.4.0   plyr_1.8.4      dplyr_0.8.0.1   tools_3.6.1
## [13] grid_3.6.1      gtable_0.3.0    xfun_0.6        withr_2.1.2
## [17] htmltools_0.3.6 assertthat_0.2.1 yaml_2.2.0       lazyeval_0.2.2
## [21] digest_0.6.25   tibble_2.1.1    crayon_1.3.4    purrr_0.3.3
## [25] glue_1.4.0      evaluate_0.13    rmarkdown_1.12  stringi_1.4.3
## [29] compiler_3.6.1 pillar_1.3.1     scales_1.0.0    pkgconfig_2.0.2
```

Here are the available libraries

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

```
## - Session info -----
## setting value
## version R version 3.6.1 (2019-07-05)
## os      Ubuntu 18.04.4 LTS
## system  x86_64, linux-gnu
## ui      X11
## language (EN)
## collate fr_FR.UTF-8
## ctype   fr_FR.UTF-8
## tz      Europe/Paris
## date    2020-04-12
##
## - Packages -----
## package      * version date      lib source
## assertthat    0.2.1   2019-03-21 [1] CRAN (R 3.6.0)
## backports     1.1.4   2019-04-10 [1] CRAN (R 3.6.0)
## callr         3.4.3   2020-03-28 [1] CRAN (R 3.6.1)
## cli           2.0.2   2020-02-28 [1] CRAN (R 3.6.1)
## colorspace    1.4-1   2019-03-18 [1] CRAN (R 3.6.0)
## crayon        1.3.4   2017-09-16 [1] CRAN (R 3.6.0)
## desc          1.2.0   2018-05-01 [1] CRAN (R 3.6.1)
## devtools      2.3.0   2020-04-10 [1] CRAN (R 3.6.1)
## digest        0.6.25  2020-02-23 [1] CRAN (R 3.6.1)
## dplyr         0.8.0.1 2019-02-15 [1] CRAN (R 3.6.0)
## ellipsis      0.3.0   2019-09-20 [1] CRAN (R 3.6.1)
## evaluate      0.13    2019-02-12 [1] CRAN (R 3.6.0)
## fansi         0.4.0   2018-10-05 [1] CRAN (R 3.6.0)
## fs            1.4.1   2020-04-04 [1] CRAN (R 3.6.1)
## ggplot2       * 3.1.1   2019-04-07 [1] CRAN (R 3.6.0)
## glue          1.4.0   2020-04-03 [1] CRAN (R 3.6.1)
## gtable        0.3.0   2019-03-25 [1] CRAN (R 3.6.0)
## htmltools     0.3.6   2017-04-28 [1] CRAN (R 3.6.0)
## knitr         1.22    2019-03-08 [1] CRAN (R 3.6.0)
## lazyeval      0.2.2   2019-03-15 [1] CRAN (R 3.6.0)
## magrittr      1.5     2014-11-22 [1] CRAN (R 3.6.0)
## memoise       1.1.0   2017-04-21 [1] CRAN (R 3.6.1)
## munsell       0.5.0   2018-06-12 [1] CRAN (R 3.6.0)
## pillar        1.3.1   2018-12-15 [1] CRAN (R 3.6.0)
## pkgbuild      1.0.6   2019-10-09 [1] CRAN (R 3.6.1)
```

```
## pkgconfig      2.0.2    2018-08-16 [1] CRAN (R 3.6.0)
## pkgload        1.0.2    2018-10-29 [1] CRAN (R 3.6.1)
## plyr           1.8.4    2016-06-08 [1] CRAN (R 3.6.0)
## prettyunits    1.0.2    2015-07-13 [1] CRAN (R 3.6.0)
## processx       3.4.2    2020-02-09 [1] CRAN (R 3.6.1)
## ps             1.3.2    2020-02-13 [1] CRAN (R 3.6.1)
## purrr          0.3.3    2019-10-18 [1] CRAN (R 3.6.1)
## R6             2.4.0    2019-02-14 [1] CRAN (R 3.6.0)
## Rcpp          1.0.1    2019-03-17 [1] CRAN (R 3.6.0)
## remotes        2.1.1    2020-02-15 [1] CRAN (R 3.6.1)
## rlang          0.4.5    2020-03-01 [1] CRAN (R 3.6.1)
## rmarkdown      1.12     2019-03-14 [1] CRAN (R 3.6.0)
## rprojroot      1.3-2    2018-01-03 [1] CRAN (R 3.6.0)
## scales         1.0.0    2018-08-09 [1] CRAN (R 3.6.0)
## sessioninfo    1.1.1    2018-11-05 [1] CRAN (R 3.6.1)
## stringi        1.4.3    2019-03-12 [1] CRAN (R 3.6.0)
## stringr        1.4.0    2019-02-10 [1] CRAN (R 3.6.0)
## testthat       2.3.2    2020-03-02 [1] CRAN (R 3.6.1)
## tibble         2.1.1    2019-03-16 [1] CRAN (R 3.6.0)
## tidyselect     0.2.5    2018-10-11 [1] CRAN (R 3.6.0)
## usethis        1.6.0    2020-04-09 [1] CRAN (R 3.6.1)
## withr          2.1.2    2018-03-15 [1] CRAN (R 3.6.0)
## xfun           0.6      2019-04-02 [1] CRAN (R 3.6.0)
## yaml           2.2.0    2018-07-25 [1] CRAN (R 3.6.0)
##
## [1] /home/aschmide/miniconda3/envs/mooc-rr-rstudio/lib/R/library
```

## Loading and inspecting data

Let's start by reading data:

Fist the data was downloaded from the gitlab platform and saved locally on 12/04/2020

```
data = read.csv("./shuttle.csv",header=T)
data
```

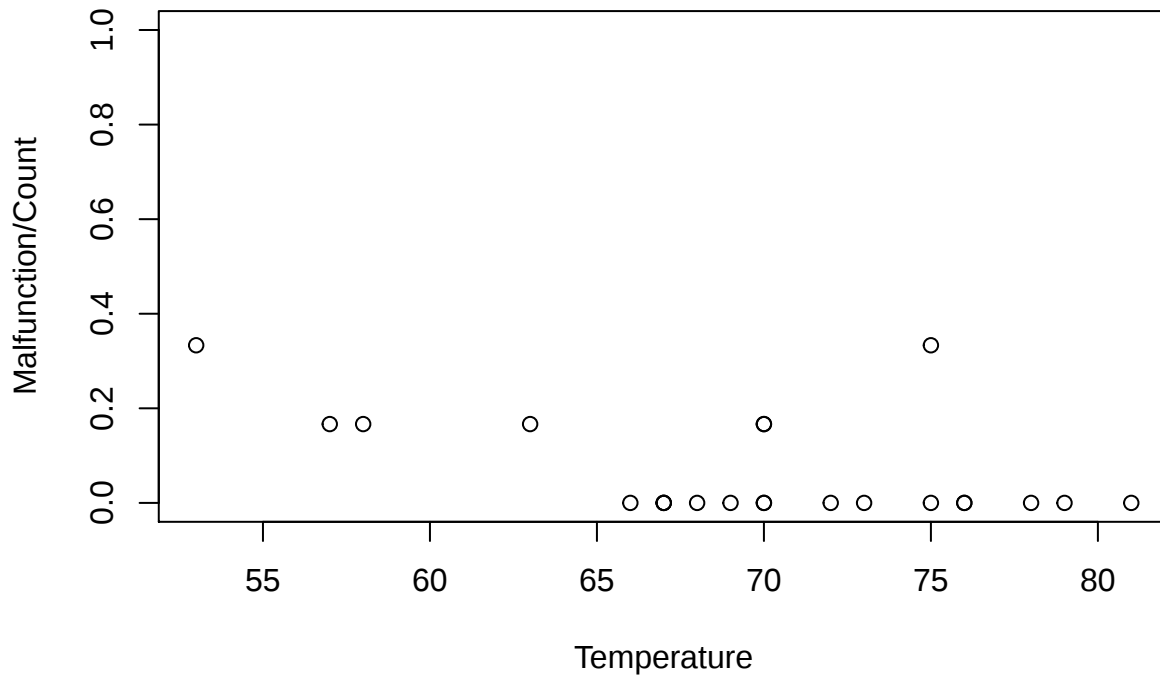
```
##      Date Count Temperature Pressure Malfunction
## 1  4/12/81     6          66        50           0
## 2 11/12/81     6          70        50           1
## 3  3/22/82     6          69        50           0
## 4 11/11/82     6          68        50           0
## 5  4/04/83     6          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:
```

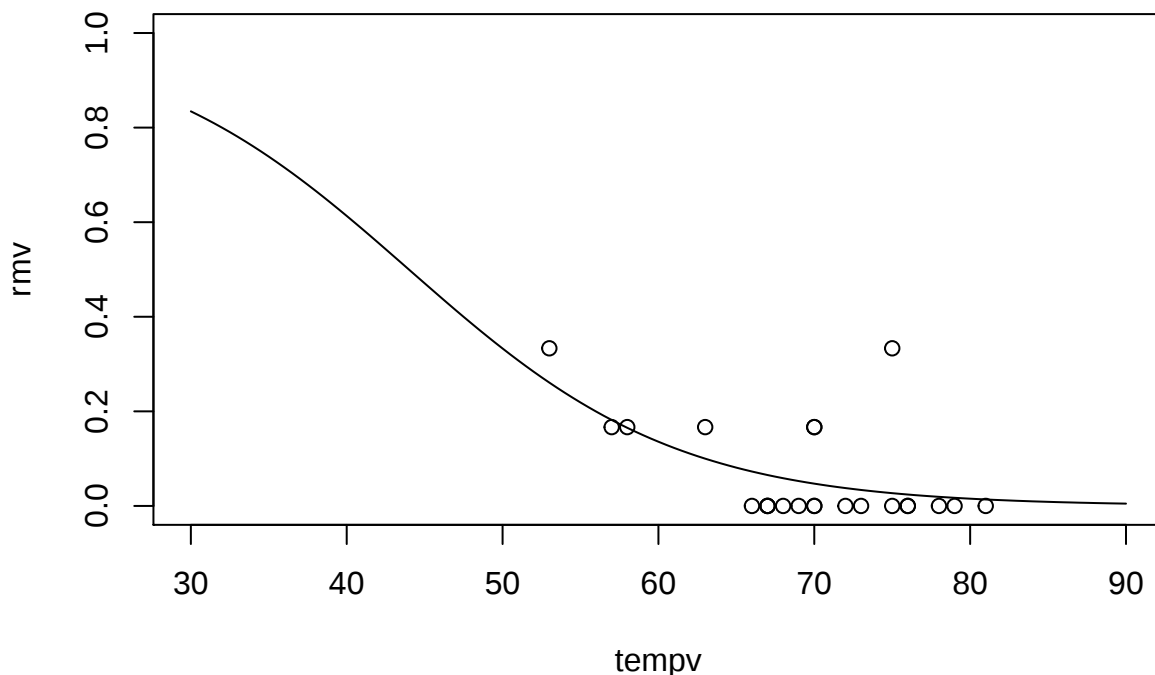
```
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.08498    3.05247   1.666  0.0957 .
## Temperature -0.11560    0.04702  -2.458  0.0140 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## 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 likelihood estimator of the intercept and of Temperature are thus  $\hat{\alpha} = 5.0849$  and  $\hat{\beta} = -0.1156$  and their standard errors are  $s_{\hat{\alpha}} = 3.052$  and  $s_{\hat{\beta}} = 0.04702$ . The Residual deviance corresponds to the Goodness of fit  $G^2 = 18.086$  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)
```



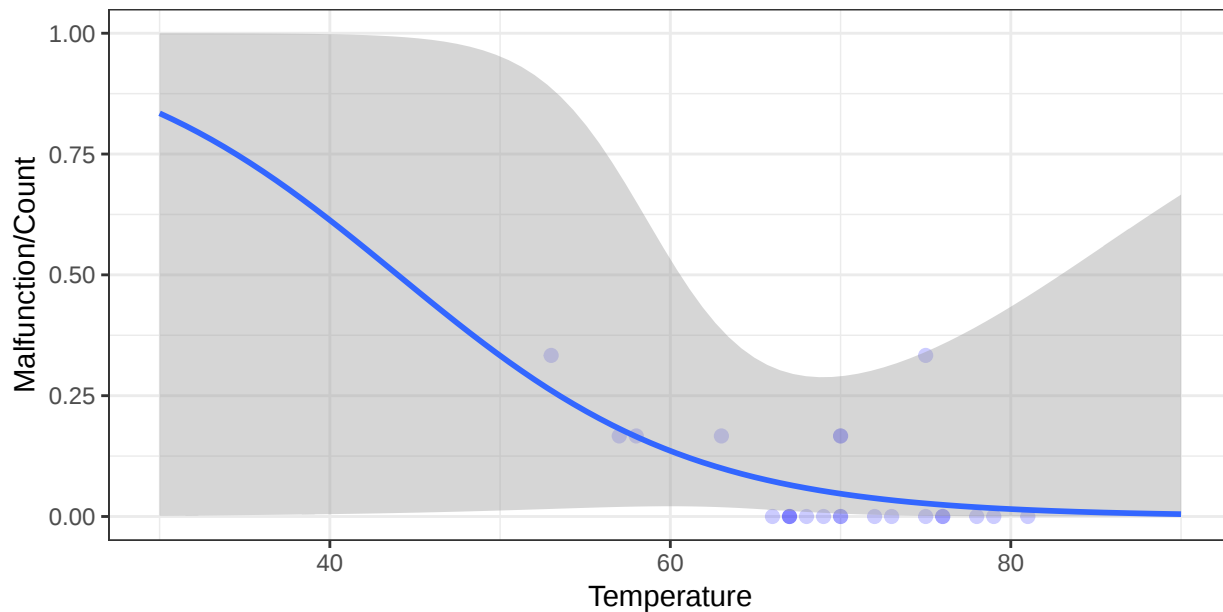
This figure is very similar to the Figure 4 of Dalal et al. **I have managed to replicate the Figure 4 of the Dalal *et al.* article.**

## Confidence on the prediction

Let's try to plot confidence intervals with ggplot2.

```
ggplot(data, aes(y=Malfuction/Count, x=Temperature)) + geom_point(alpha=.2, size = 2, color="blue") +  
  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!
```



Mmmh, I have a warning from ggplot2 indicating “*non-integer #successes in a binomial glm!*”. This seems fishy. Furthermore, this confidence region seems huge... It seems strange to me that the uncertainty grows so large for higher temperatures. And compared to my previous call to glm, I haven't indicated the weight which accounts for the fact that each ratio Malfuction/Count corresponds to Count observations (if someone knows how to do this...). There must be something wrong.

So let's provide the “raw” data to ggplot2.

```
data_flat=data.frame()  
for(i in 1:nrow(data)) {  
  temperature = data[i,"Temperature"];  
  malfuction = data[i,"Malfuction"];  
  d = data.frame(Temperature=temperature,Malfuction=rep(0,times = data[i,"Count"]))  
  if(malfuction>0) {  
    d[1:malfuction, "Malfuction"]=1;  
  }  
  data_flat=rbind(data_flat,d)  
}  
dim(data_flat)
```

```
## [1] 138 2
```

```
str(data_flat)
```

```
## 'data.frame': 138 obs. of 2 variables:  
## $ Temperature: int 66 66 66 66 66 66 70 70 70 70 ...
```

```
## $ Malfunction: num 0 0 0 0 0 0 1 0 0 0 ...
```

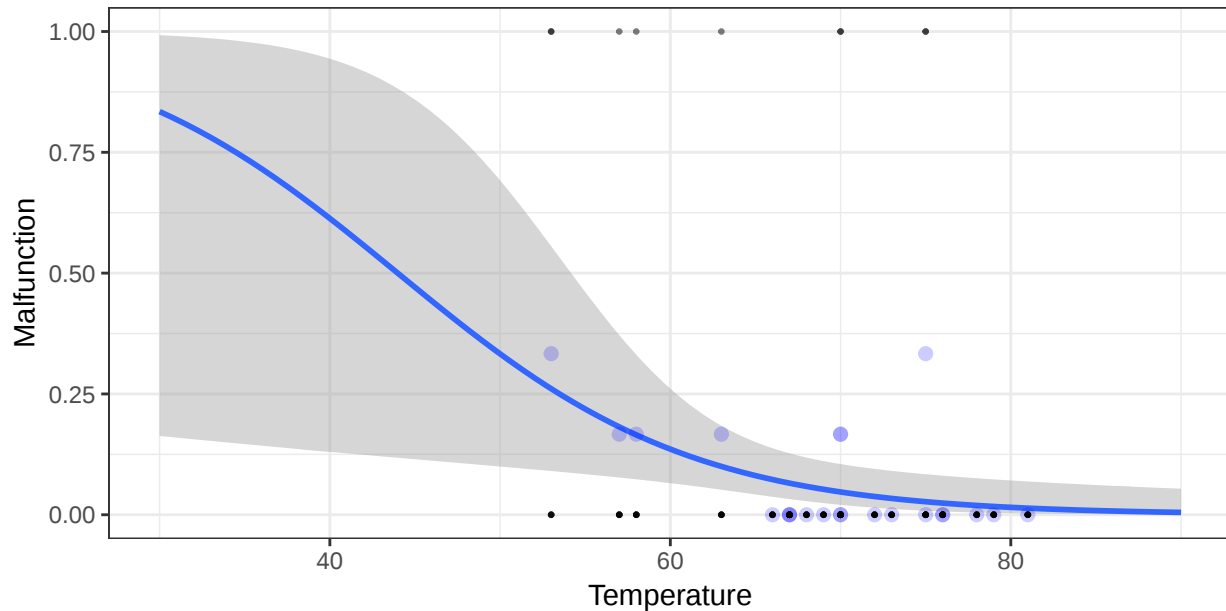
Let's check whether I obtain the same regression or not:

```
logistic_reg_flat = glm(data=data_flat, Malfunction ~ Temperature, 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
```

Perfect. The estimates and the standard errors are the same although the Residual deviance is difference since the distance is now measured with respect to each 0/1 measurement and not to ratios. Let's use plot the regression for *data\_flat* along with the ratios (*data*).

```
ggplot(data=data_flat, aes(y=Malfunction, x=Temperature)) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"), fullrange=T) +
  geom_point(data=data, aes(y=Malfunction/Count, x=Temperature), alpha=.2, size = 2, color="blue") +
  geom_point(alpha=.5, size = .5) +
  xlim(30,90) + ylim(0,1) + theme_bw()
```



This confidence interval seems much more reasonable (in accordance with the data) than the previous one. Let's check whether it corresponds to the prediction obtained when calling directly predict. Obtaining the prediction can be done directly or through the link function.

Here is the “direct” (response) version I used in my very first plot:

```
pred = predict(logistic_reg_flat,list(Temperature=30),type="response",se.fit = T)
pred
```

```
## $fit
##      1
## 0.834373
##
## $se.fit
##      1
## 0.2293304
##
## $residual.scale
## [1] 1
```

The estimated Failure probability for 30° is thus 0.834. However, the *se.fit* value seems pretty hard to use as I can obviously not simply add  $\pm 2se.fit$  to *fit* to compute a confidence interval.

Here is the “link” version:

```
pred_link = predict(logistic_reg_flat,list(Temperature=30),type="link",se.fit = T)
pred_link
```

```
## $fit
##      1
## 1.616942
##
## $se.fit
## [1] 1.659473
##
## $residual.scale
## [1] 1
```

```
logistic_reg$family$linkinv(pred_link$fit)
```

```
##          1  
## 0.834373
```

I recover 0.834 for the estimated Failure probability at 30°. But now, going through the *linkinv* function, we can use *se.fit*:

```
critval = 1.96  
logistic_reg$family$linkinv(c(pred_link$fit-critval*pred_link$se.fit,  
                               pred_link$fit+critval*pred_link$se.fit))
```

```
##          1          1  
## 0.1630612 0.9923814
```

The 95% confidence interval for our estimation is thus [0.163,0.992]. This is what ggplot2 just plotted me. This seems coherent.

**I am now rather confident that I have managed to correctly compute and plot the uncertainty of my prediction.** Let's be honest, it took me a while. My first attempts were plainly wrong (I didn't know how to do this so I trusted ggplot2, which I was misusing) and did not use the correct statistical method. I also feel confident now because this has been somehow validated by other colleagues but it will be interesting that you collect other kind of plots values that you obtained, that differ and that you would probably have kept if you didn't have a reference to compare to. Please provide us with as many versions as you can.