Challenger - R - Emacs - Windows 7 64 bits

December 10, 2018

Contents

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

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.

2 Technical information on the computer on which the analysis is run

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

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

```
R version 3.5.1 (2018-07-02)
Platform: i386-w64-mingw32/i386 (32-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_3.1.0

loaded via a namespace (and not attached):
 [1] bindrcpp_0.2.2    digest_0.6.18     R6_2.3.0          scales_1.0.0
 [5] assertthat_0.2.0  rprojroot_1.3-2   grid_3.5.1        fs_1.2.6
 [9] devtools_2.0.1    cli_1.0.1         tidyselect_0.2.5  desc_1.2.0
[13] testthat_2.0.1    munsell_0.5.0     pillar_1.3.0      compiler_3.5.1
[17] tibble_1.4.2      pkgconfig_2.0.2   labeling_0.3      purrr_0.2.5
[21] plyr_1.8.4        pkgload_1.0.2     sessioninfo_1.1.1 glue_1.3.0
[25] base64enc_0.1-3   ps_1.2.1          remotes_2.0.2     pkgbuild_1.0.2
[29] processx_3.2.0    magrittr_1.5      gtable_0.2.0      rlang_0.3.0.1
[33] prettyunits_1.0.2 colorspace_1.3-2  tools_3.5.1       bindr_0.1.1
[37] callr_3.0.0       dplyr_0.7.8       withr_2.1.2       usethis_1.4.0
[41] lazyeval_0.2.1    crayon_1.3.4      backports_1.1.2   memoise_1.1.0
[45] Rcpp_1.0.0
```

Here are the available libraries

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

```
- Session info -------------------------------------------------------------
 setting  value
 version  R version 3.5.1 (2018-07-02)
 os       Windows 7 x64 SP 1
 system   i386, mingw32
 ui       RTerm
 language (EN)
 collate  French_France.1252
 ctype    French_France.1252
 tz       Europe/Paris
 date     2018-12-10

- Packages -----------------------------------------------------------------
 package     * version date       lib source
 assertthat    0.2.0   2017-04-11 [1] CRAN (R 3.5.1)
 backports     1.1.2   2017-12-13 [1] CRAN (R 3.5.0)
 base64enc     0.1-3   2015-07-28 [1] CRAN (R 3.5.0)
 bindr         0.1.1   2018-03-13 [1] CRAN (R 3.5.1)
 bindrcpp      0.2.2   2018-03-29 [1] CRAN (R 3.5.1)
 callr         3.0.0   2018-08-24 [1] CRAN (R 3.5.1)
 cli           1.0.1   2018-09-25 [1] CRAN (R 3.5.1)
 colorspace    1.3-2   2016-12-14 [1] CRAN (R 3.5.1)
 crayon        1.3.4   2017-09-16 [1] CRAN (R 3.5.1)
 desc          1.2.0   2018-05-01 [1] CRAN (R 3.5.1)
 devtools      2.0.1   2018-10-26 [1] CRAN (R 3.5.1)
 digest        0.6.18  2018-10-10 [1] CRAN (R 3.5.1)
 dplyr         0.7.8   2018-11-10 [1] CRAN (R 3.5.1)
```

```
 fs            1.2.6   2018-08-23 [1] CRAN (R 3.5.1)
 ggplot2     * 3.1.0   2018-10-25 [1] CRAN (R 3.5.1)
 glue          1.3.0   2018-07-17 [1] CRAN (R 3.5.1)
 gtable        0.2.0   2016-02-26 [1] CRAN (R 3.5.1)
 labeling      0.3     2014-08-23 [1] CRAN (R 3.5.0)
 lazyeval      0.2.1   2017-10-29 [1] CRAN (R 3.5.1)
 magrittr      1.5     2014-11-22 [1] CRAN (R 3.5.1)
 memoise       1.1.0   2017-04-21 [1] CRAN (R 3.5.1)
 munsell       0.5.0   2018-06-12 [1] CRAN (R 3.5.1)
 pillar        1.3.0   2018-07-14 [1] CRAN (R 3.5.1)
 pkgbuild      1.0.2   2018-10-16 [1] CRAN (R 3.5.1)
 pkgconfig     2.0.2   2018-08-16 [1] CRAN (R 3.5.1)
 pkgload       1.0.2   2018-10-29 [1] CRAN (R 3.5.1)
 plyr          1.8.4   2016-06-08 [1] CRAN (R 3.5.1)
 prettyunits   1.0.2   2015-07-13 [1] CRAN (R 3.5.1)
 processx      3.2.0   2018-08-16 [1] CRAN (R 3.5.1)
 ps            1.2.1   2018-11-06 [1] CRAN (R 3.5.1)
 purrr         0.2.5   2018-05-29 [1] CRAN (R 3.5.1)
 R6            2.3.0   2018-10-04 [1] CRAN (R 3.5.1)
 Rcpp          1.0.0   2018-11-07 [1] CRAN (R 3.5.1)
 remotes       2.0.2   2018-10-30 [1] CRAN (R 3.5.1)
 rlang         0.3.0.1 2018-10-25 [1] CRAN (R 3.5.1)
 rprojroot     1.3-2   2018-01-03 [1] CRAN (R 3.5.1)
 scales        1.0.0   2018-08-09 [1] CRAN (R 3.5.1)
 sessioninfo   1.1.1   2018-11-05 [1] CRAN (R 3.5.1)
 testthat      2.0.1   2018-10-13 [1] CRAN (R 3.5.1)
 tibble        1.4.2   2018-01-22 [1] CRAN (R 3.5.1)
 tidyselect    0.2.5   2018-10-11 [1] CRAN (R 3.5.1)
 usethis       1.4.0   2018-08-14 [1] CRAN (R 3.5.1)
 withr         2.1.2   2018-03-15 [1] CRAN (R 3.5.1)

[1] c:/Program Files/R/R-3.5.1/library
```

3 Loading and inspecting data

Let's start by reading data:

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

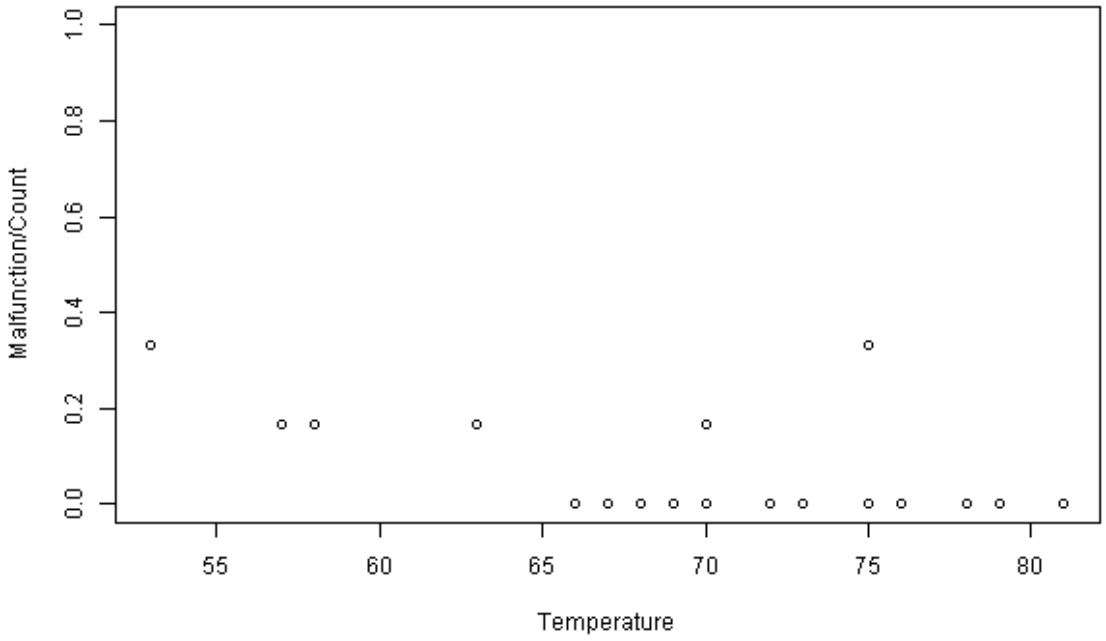
```
         Date Count Temperature Pressure Malfunction
1    4/12/81     6          66       50           0
2   11/12/81     6          70       50           1
3    3/22/82     6          69       50           0
4   11/11/82     6          68       50           0
5    4/04/83     6          67       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))
```



4 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)
```

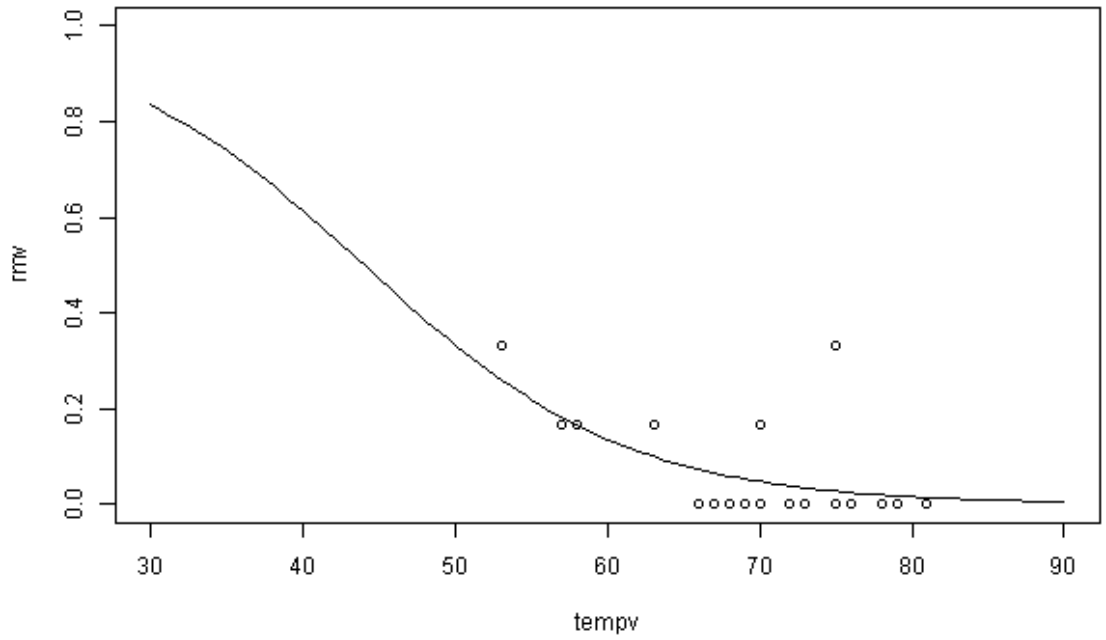
```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.95227  -0.78299  -0.54117  -0.04379   2.65152

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 estimatator of the intercept and of Temperature are thus $\hat{\alpha} =$ **5.0850** 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**.

5 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)
```

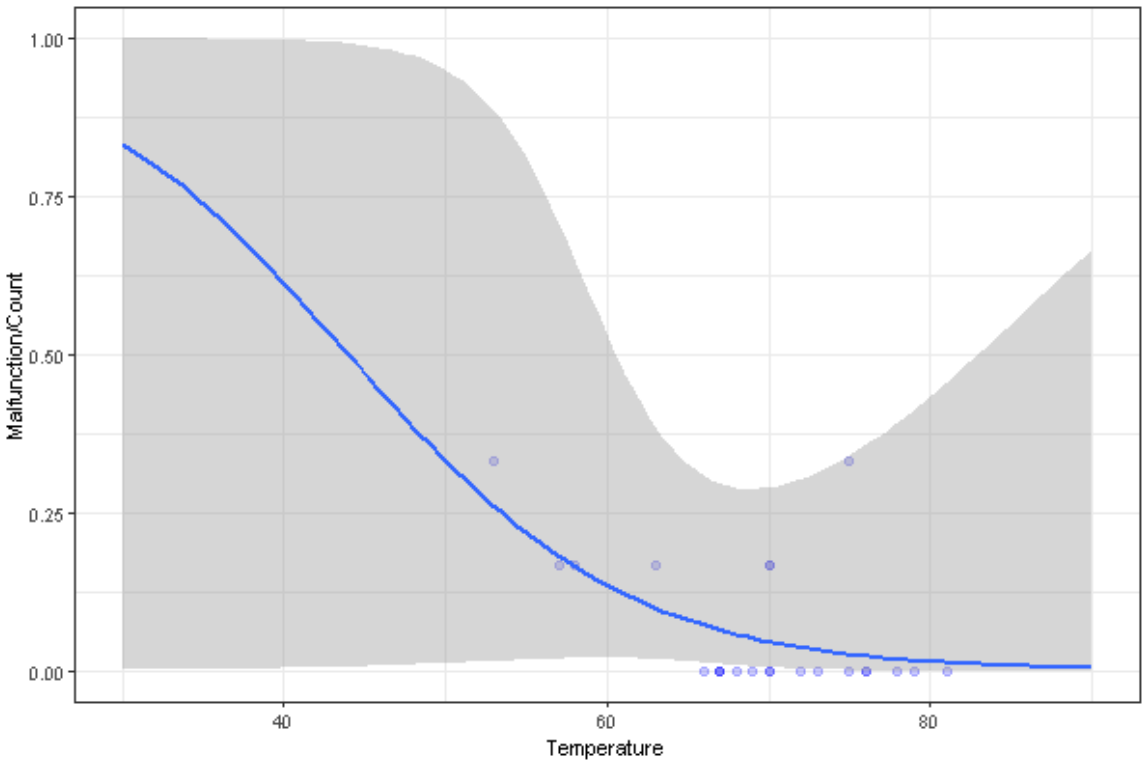


This figure is very similar to the Figure 4 of Dalal *et al.* **I have managed to replicate the Figure 4 of the Dalal *et al.* article.**

6 Confidence on the prediction

Let's try to plot confidence intervals with `ggplot2`.

```
ggplot(data, aes(y=Malfunction/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()
```



I don't get any warning from `ggplot2` indicating *"non-integer #successes in a binomial glm!"* but 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 ration Malfunction/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"];
  malfunction = data[i,"Malfunction"];
  d = data.frame(Temperature=temperature,Malfunction=rep(0,times = data[i,"Count"]))
  if(malfunction>0) {
    d[1:malfunction, "Malfunction"]=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 ...
```

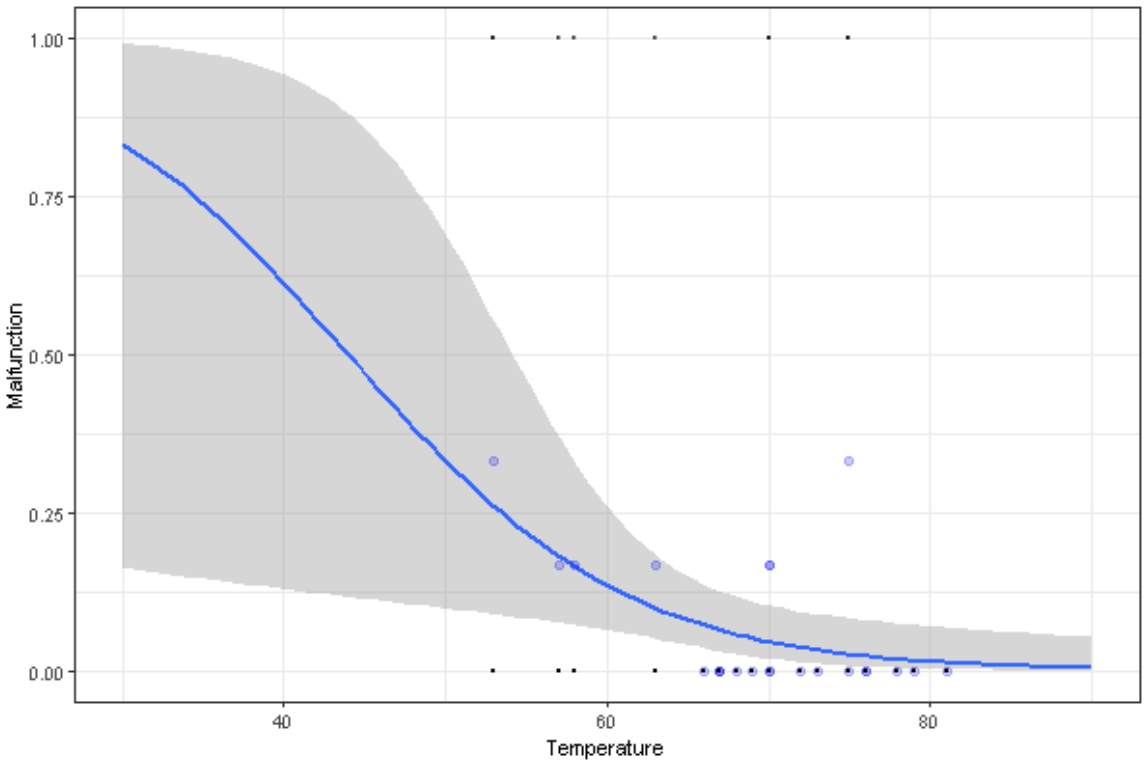
Let's check whether I obtain the same regression or not:

```
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 for him 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 rations. 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 ± 2 *se.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
```

```
        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 uncertainty of my prediction. Let's be honnest, it took me a wile. 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 becuase this has been somehow validated by other colleagues but it will be interesting that you collect other kind of plot 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.