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

26 février 2021

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.

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

```
## R version 4.0.3 (2020-10-10)
## 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_3.3.2
##
## loaded via a namespace (and not attached):
##  [1] knitr_1.30       magrittr_1.5     tidyselect_1.1.0 munsell_0.5.0
##  [5] colorspace_2.0-0 R6_2.5.0         rlang_0.4.8      stringr_1.4.0
##  [9] dplyr_1.0.2      tools_4.0.3      grid_4.0.3       gtable_0.3.0
## [13] xfun_0.19        withr_2.3.0      htmltools_0.5.0  ellipsis_0.3.1
## [17] yaml_2.2.1       digest_0.6.27    tibble_3.0.4     lifecycle_0.2.0
```

```
## [21] crayon_1.3.4     purrr_0.3.4      vctrs_0.3.4      glue_1.4.2
## [25] evaluate_0.14    rmarkdown_2.5    stringi_1.5.3    compiler_4.0.3
## [29] pillar_1.4.6     generics_0.1.0   scales_1.1.1     pkgconfig_2.0.3
```

Here are the available libraries

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

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.3 (2020-10-10)
##  os       Windows 7 x64 SP 1
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  French_France.1252
##  ctype    French_France.1252
##  tz       Europe/Paris
##  date     2021-02-26
##
## - Packages -------------------------------------------------------------------
##  package     * version date       lib source
##  assertthat    0.2.1   2019-03-21 [1] CRAN (R 4.0.3)
##  callr         3.5.1   2020-10-13 [1] CRAN (R 4.0.3)
##  cli           2.1.0   2020-10-12 [1] CRAN (R 4.0.3)
##  colorspace    2.0-0   2020-11-11 [1] CRAN (R 4.0.3)
##  crayon        1.3.4   2017-09-16 [1] CRAN (R 4.0.3)
##  desc          1.2.0   2018-05-01 [1] CRAN (R 4.0.3)
##  devtools      2.3.2   2020-09-18 [1] CRAN (R 4.0.3)
##  digest        0.6.27  2020-10-24 [1] CRAN (R 4.0.3)
##  dplyr         1.0.2   2020-08-18 [1] CRAN (R 4.0.3)
##  ellipsis      0.3.1   2020-05-15 [1] CRAN (R 4.0.3)
##  evaluate      0.14    2019-05-28 [1] CRAN (R 4.0.3)
##  fansi         0.4.1   2020-01-08 [1] CRAN (R 4.0.3)
##  fs            1.5.0   2020-07-31 [1] CRAN (R 4.0.3)
##  generics      0.1.0   2020-10-31 [1] CRAN (R 4.0.3)
##  ggplot2     * 3.3.2   2020-06-19 [1] CRAN (R 4.0.3)
##  glue          1.4.2   2020-08-27 [1] CRAN (R 4.0.3)
##  gtable        0.3.0   2019-03-25 [1] CRAN (R 4.0.3)
##  htmltools     0.5.0   2020-06-16 [1] CRAN (R 4.0.3)
##  knitr         1.30    2020-09-22 [1] CRAN (R 4.0.3)
##  lifecycle     0.2.0   2020-03-06 [1] CRAN (R 4.0.3)
##  magrittr      1.5     2014-11-22 [1] CRAN (R 4.0.3)
##  memoise       1.1.0   2017-04-21 [1] CRAN (R 4.0.3)
##  munsell       0.5.0   2018-06-12 [1] CRAN (R 4.0.3)
##  pillar        1.4.6   2020-07-10 [1] CRAN (R 4.0.3)
##  pkgbuild      1.1.0   2020-07-13 [1] CRAN (R 4.0.3)
##  pkgconfig     2.0.3   2019-09-22 [1] CRAN (R 4.0.3)
##  pkgload       1.1.0   2020-05-29 [1] CRAN (R 4.0.3)
##  prettyunits   1.1.1   2020-01-24 [1] CRAN (R 4.0.3)
##  processx      3.4.4   2020-09-03 [1] CRAN (R 4.0.3)
##  ps            1.4.0   2020-10-07 [1] CRAN (R 4.0.3)
##  purrr         0.3.4   2020-04-17 [1] CRAN (R 4.0.3)
```

```
##  R6            2.5.0   2020-10-28 [1] CRAN (R 4.0.3)
##  remotes       2.2.0   2020-07-21 [1] CRAN (R 4.0.3)
##  rlang         0.4.8   2020-10-08 [1] CRAN (R 4.0.3)
##  rmarkdown     2.5     2020-10-21 [1] CRAN (R 4.0.3)
##  rprojroot     2.0.2   2020-11-15 [1] CRAN (R 4.0.3)
##  scales        1.1.1   2020-05-11 [1] CRAN (R 4.0.3)
##  sessioninfo   1.1.1   2018-11-05 [1] CRAN (R 4.0.3)
##  stringi       1.5.3   2020-09-09 [1] CRAN (R 4.0.3)
##  stringr       1.4.0   2019-02-10 [1] CRAN (R 4.0.3)
##  testthat      3.0.0   2020-10-31 [1] CRAN (R 4.0.3)
##  tibble        3.0.4   2020-10-12 [1] CRAN (R 4.0.3)
##  tidyselect    1.1.0   2020-05-11 [1] CRAN (R 4.0.3)
##  usethis       1.6.3   2020-09-17 [1] CRAN (R 4.0.3)
##  vctrs         0.3.4   2020-08-29 [1] CRAN (R 4.0.3)
##  withr         2.3.0   2020-09-22 [1] CRAN (R 4.0.3)
##  xfun          0.19    2020-10-30 [1] CRAN (R 4.0.3)
##  yaml          2.2.1   2020-02-01 [1] CRAN (R 4.0.3)
##
## [1] C:/R/R-4.0.3/library
```

Loading and inspecting data

Let's start by reading data:

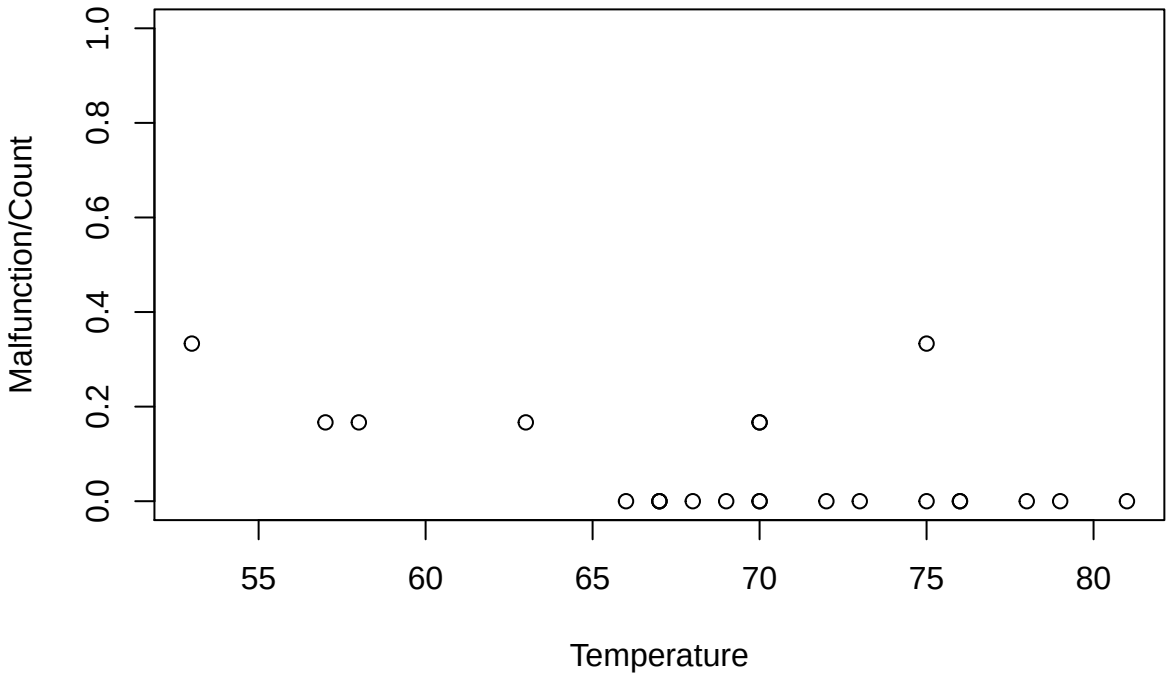
```
data_url <- "https://app-learninglab.inria.fr/moocrr/gitlab/moocrr-session3/moocrr-reproducibility-study
data = read.csv(data_url, h=T)
head(data)
```

```
##       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
```

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:

```r
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.

```r
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 .
```

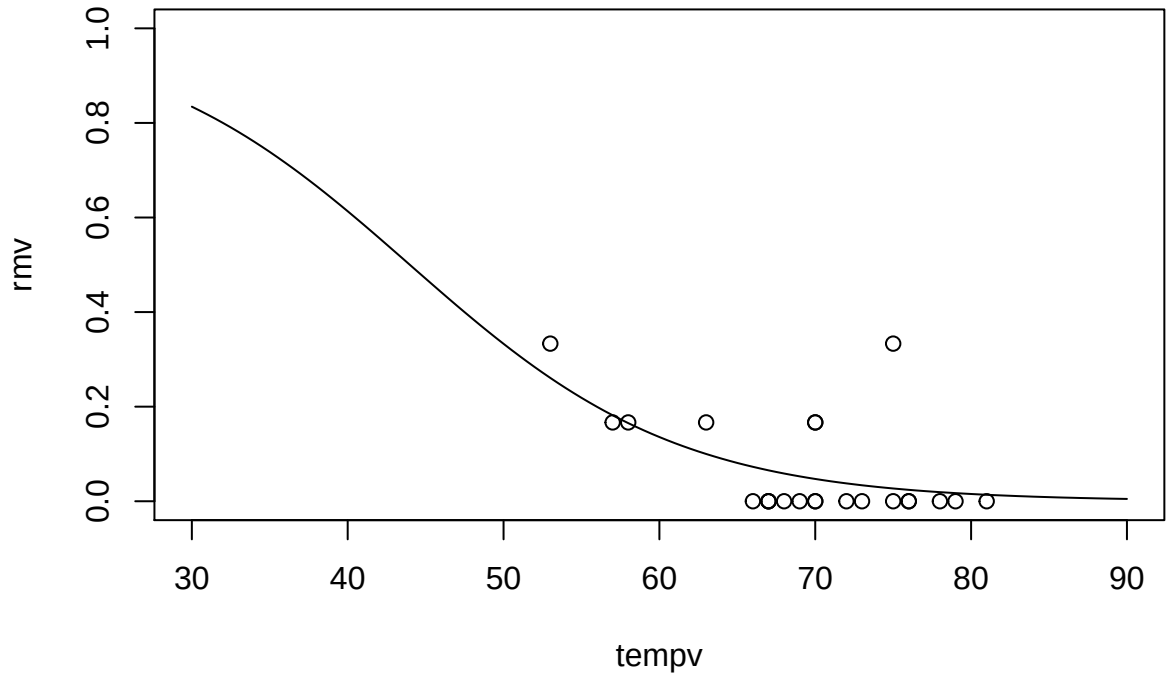
```
## Temperature -0.11560    0.04702  -2.458   0.0140 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.230  on 22  degrees of freedom
## Residual deviance: 18.086  on 21  degrees of freedom
## AIC: 35.647
##
## Number of Fisher Scoring iterations: 5
```

The maximum likelyhood estimator of the intercept and of Temperature are thus $\hat{\alpha} = 5.0849$ 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.**

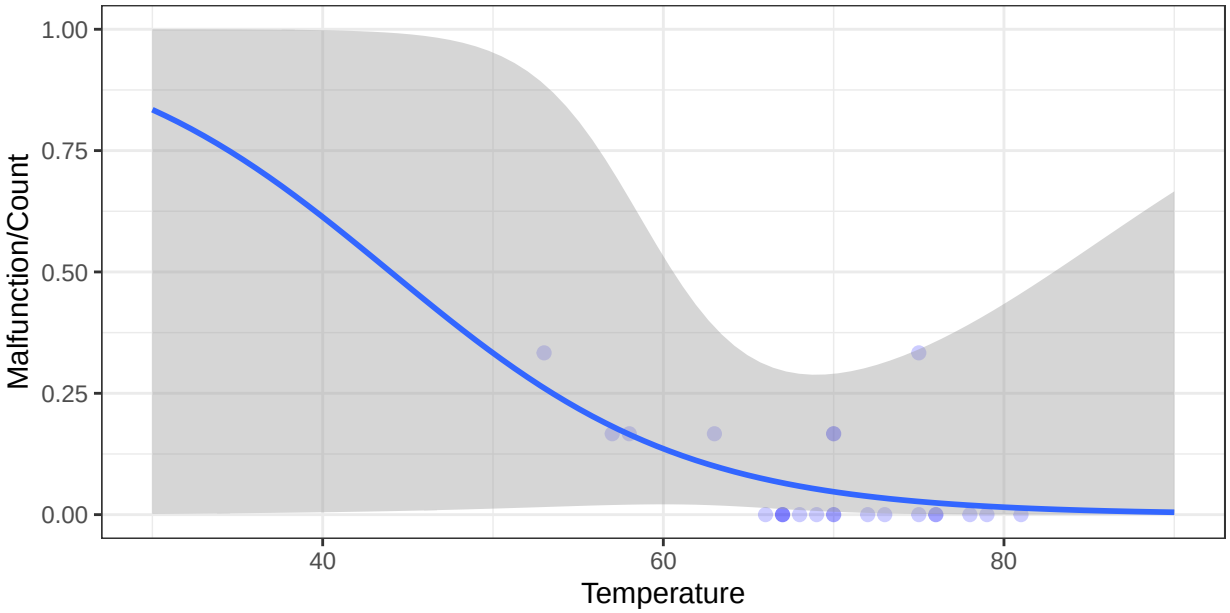
Confidence on the prediction

Let's try to plot confidence intervals with ggplot2.

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

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## 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 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:

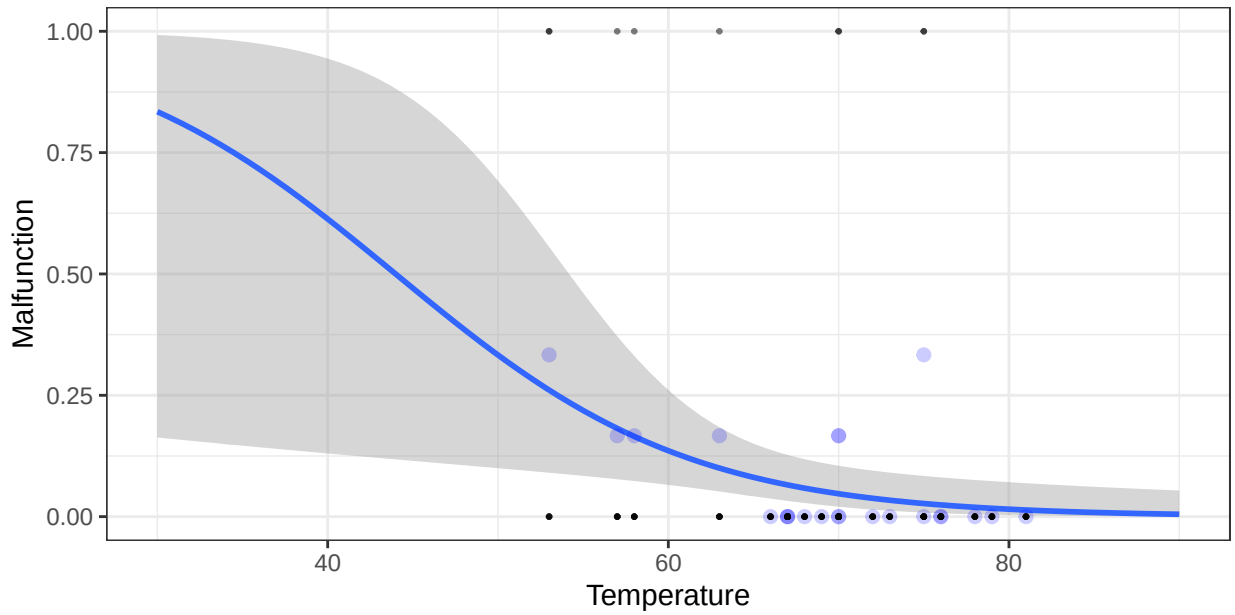
```
logistic_reg_flat = glm(data=data_flat, Malfunction ~ Temperature, family=binomial(link='logit'))
summary(logistic_reg)
```

```
##
## 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 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()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



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 $\pm 2 se.fit$ 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$fit+critval*pred_link$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 honnest, 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.