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

Matthieu HAAS

05/02/2021

Dans ce travail, l'objectif est de répliquer certaines analyses de l'article *Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure*, disponible ici. A la quatrième page de l'article, ils présentent les meilleures estimations de la régression logistique utilisant uniquement la température comme étant : $\widehat{\alpha} = 5.085$ et $\widehat{\beta} = -0.1156$ et leur erreur standard asymtpotique respective est de : $s_{\widehat{\alpha}} = 3.052$ et $s_{\widehat{\beta}} = 0.047$.La qualité d'ajustement de ce modèle est de : $G^2 = 18.086$ avec 21 degrés de liberté. Notre objectif est de reproduire les étapes derrière ce calcul et la figure 4 de l'article, en l'améliorant si possible.

Information technique sur l'ordinateur réalisant l'analyse

Nous utiliserons ici un langage R et la librairie ggplot2.

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

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.3
##
## loaded via a namespace (and not attached):
##  [1] knitr_1.31       magrittr_2.0.1   munsell_0.5.0    colorspace_2.0-0
##  [5] R6_2.5.0         rlang_0.4.10     stringr_1.4.0    tools_4.0.3
##  [9] grid_4.0.3       gtable_0.3.0     xfun_0.20        withr_2.4.1
## [13] htmltools_0.5.1  ellipsis_0.3.1   yaml_2.2.1       digest_0.6.27
```

```
## [17] tibble_3.0.6     lifecycle_0.2.0  crayon_1.4.0     vctrs_0.3.6
## [21] glue_1.4.2       evaluate_0.14    rmarkdown_2.6    stringi_1.5.3
## [25] compiler_4.0.3   pillar_1.4.7     scales_1.1.1     pkgconfig_2.0.3
```

Voici les librairies disponibles.

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

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.3 (2020-10-10)
##  os       Windows 10 x64
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United Kingdom.1252
##  ctype    English_United Kingdom.1252
##  tz       Europe/Paris
##  date     2021-02-09
##
## - Packages -------------------------------------------------------------------
##  package     * version date       lib source
##  assertthat    0.2.1   2019-03-21 [1] CRAN (R 4.0.3)
##  cachem        1.0.1   2021-01-21 [1] CRAN (R 4.0.3)
##  callr         3.5.1   2020-10-13 [1] CRAN (R 4.0.3)
##  cli           2.3.0   2021-01-31 [1] CRAN (R 4.0.3)
##  colorspace    2.0-0   2020-11-11 [1] CRAN (R 4.0.3)
##  crayon        1.4.0   2021-01-30 [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)
##  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)
##  fastmap       1.1.0   2021-01-25 [1] CRAN (R 4.0.3)
##  fs            1.5.0   2020-07-31 [1] CRAN (R 4.0.3)
##  ggplot2     * 3.3.3   2020-12-30 [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     0.5.1   2021-01-12 [1] CRAN (R 4.0.3)
##  knitr         1.31    2021-01-27 [1] CRAN (R 4.0.3)
##  lifecycle     0.2.0   2020-03-06 [1] CRAN (R 4.0.3)
##  magrittr      2.0.1   2020-11-17 [1] CRAN (R 4.0.3)
##  memoise       2.0.0   2021-01-26 [1] CRAN (R 4.0.3)
##  munsell       0.5.0   2018-06-12 [1] CRAN (R 4.0.3)
##  pillar        1.4.7   2020-11-20 [1] CRAN (R 4.0.3)
##  pkgbuild      1.2.0   2020-12-15 [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      3.4.5   2020-11-30 [1] CRAN (R 4.0.3)
##  ps            1.5.0   2020-12-05 [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         0.4.10  2020-12-30 [1] CRAN (R 4.0.3)
##  rmarkdown     2.6     2020-12-14 [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      3.0.1   2020-12-17 [1] CRAN (R 4.0.3)
##  tibble        3.0.6   2021-01-29 [1] CRAN (R 4.0.3)
##  usethis       2.0.0   2020-12-10 [1] CRAN (R 4.0.3)
##  vctrs         0.3.6   2020-12-17 [1] CRAN (R 4.0.3)
##  withr         2.4.1   2021-01-26 [1] CRAN (R 4.0.3)
##  xfun          0.20    2021-01-06 [1] CRAN (R 4.0.3)
##  yaml          2.2.1   2020-02-01 [1] CRAN (R 4.0.3)
##
## [1] C:/Users/mhaas/Documents/R/win-library/4.0
## [2] C:/Program Files/R/R-4.0.3/library
```

Chargement et vérification des données

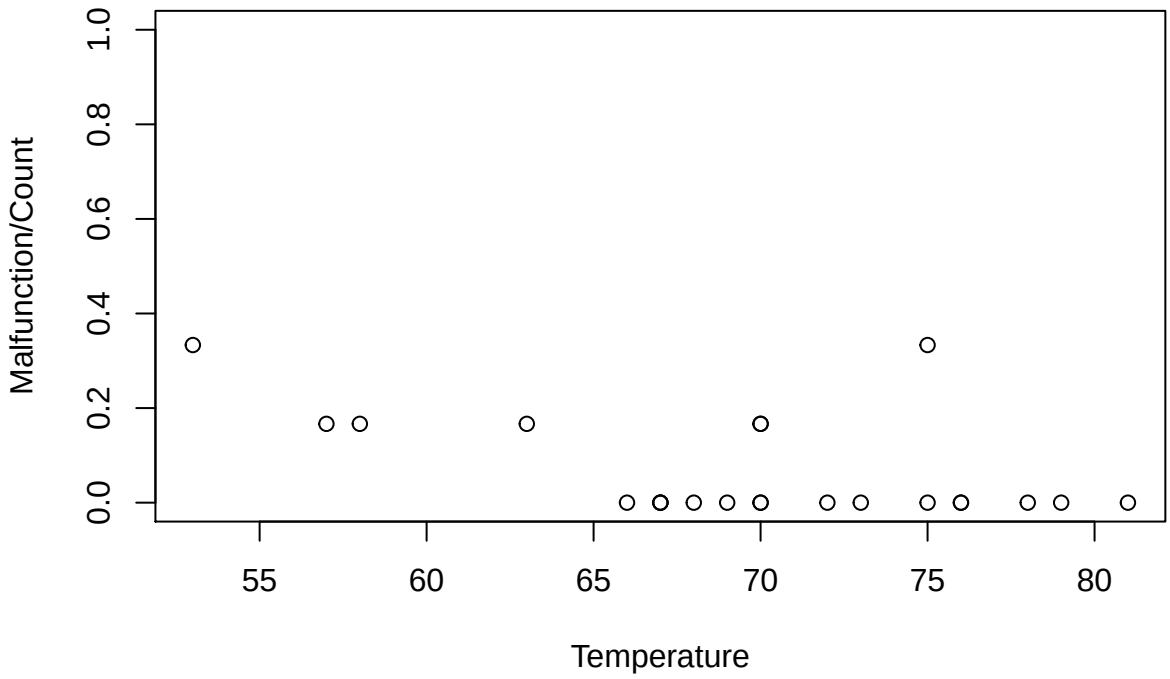
La première étape est de charger les données.

```
data = read.csv("https://app-learninglab.inria.fr/moocrr/gitlab/moocrr-session3/moocrr-reproducibility-s
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
```

On peut représenter l'impact de la température sur les échecs.

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



Régression logistique

Ainsi, une régression logistique doit nous permettre d'estimer l'influence de la température.

```
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
```

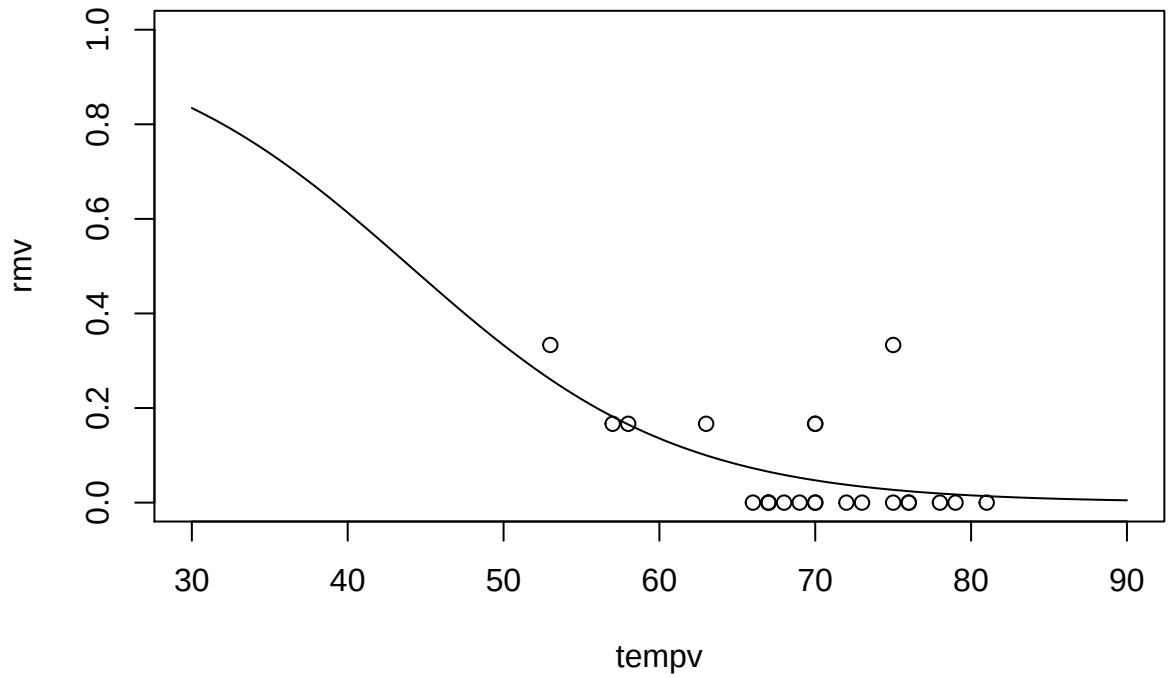
```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.230  on 22  degrees of freedom
## Residual deviance: 18.086  on 21  degrees of freedom
## AIC: 35.647
##
## Number of Fisher Scoring iterations: 5
```

Les meilleures estimations de la régression logistique utilisant uniquement la température sont : $\hat{\alpha} = 5.08498$ et $\hat{\beta} = -0.11560$ et leur erreur standard asymtpotique respective est de : $s_{\hat{\alpha}} = 3.05247$ et $s_{\hat{\beta}} = 0.04702$.La qualité d'ajustement de ce modèle est de : $G^2 = 18.086$ avec 21 degrés de liberté. De fait, je réplique les résultats de l'article de Dalal *et al.*

Prédiction du risque d'échec

Lors du lancement, la température lors du lancement était de 31°F. Estimons la probabilité d'un échec dans ces conditions en utilisant notre modèle:

```
# 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)
```



Cette figure est proche de la figure 4 de l'article de Dalal *et al.* Ainsi, on considère qu'elle est bien répliquée.

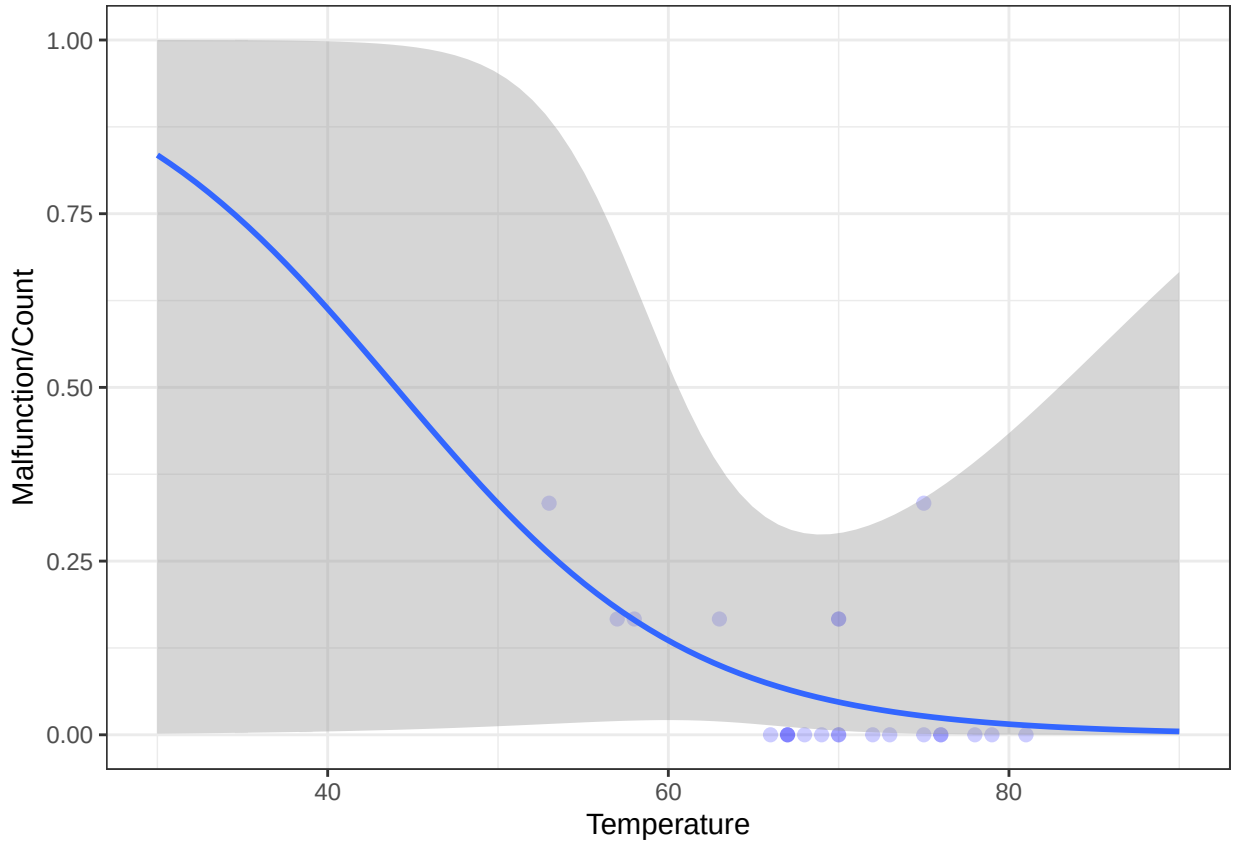
Confiance de la prédiction

A l'aide de ggplot, représentons l'intervalle de confiance.

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

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

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



On a un message d'avertissement de ggplot indiquant "non-integer #successes in a binomial glm!". C'est étrange. De plus, cet intervalle de confiance est énorme.

```
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)
```

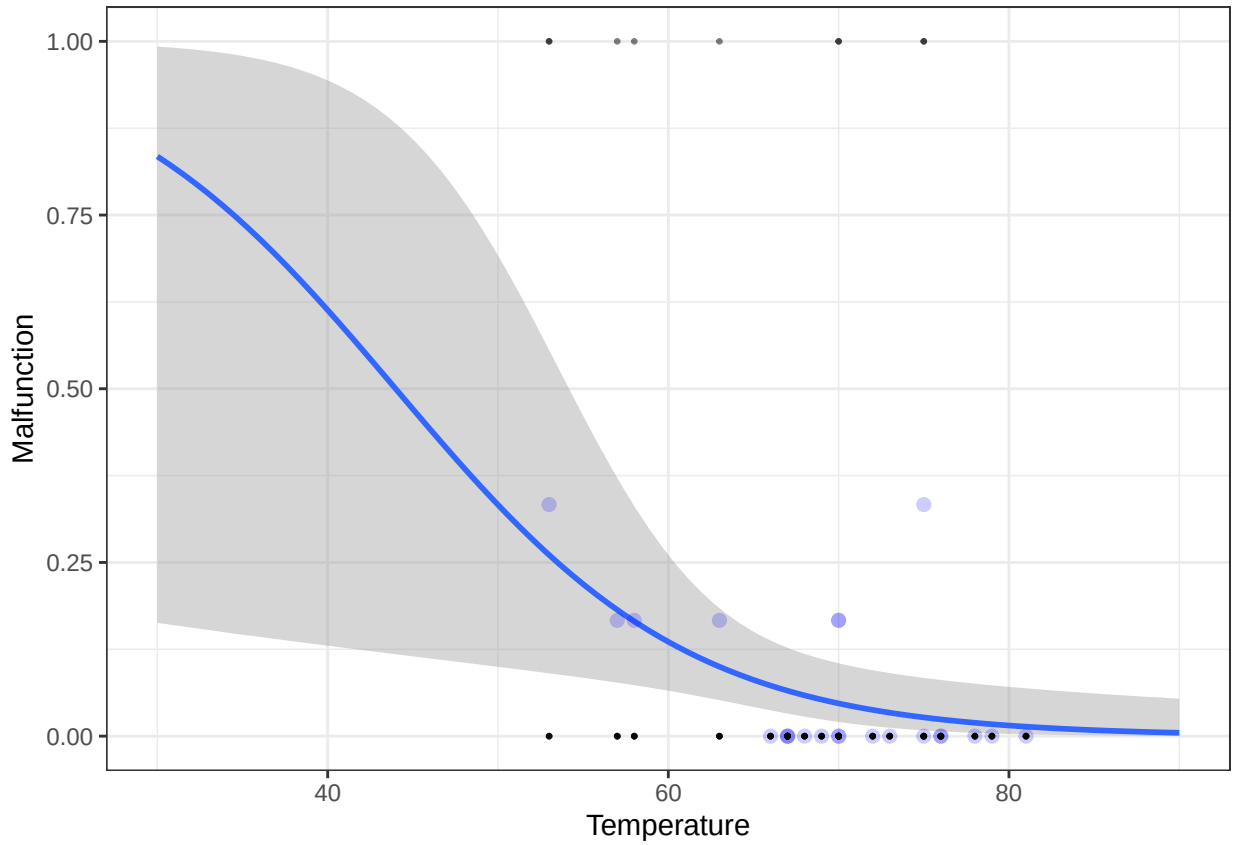
```
## 'data.frame':    138 obs. of  2 variables:
##  $ Temperature: int  66 66 66 66 66 66 70 70 70 70 ...
##  $ Malfunction: num  0 0 0 0 0 0 1 0 0 0 ...
```

```
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
```

```
ggplot(data=data_flat,aes(y=Malfunction, x=Temperature))+geom_smooth(method = "glm", method.args =list(
```

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




```
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
```

Pour une température de 30°F, le risque d'échec est de 0.834.

```
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
```

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

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

L'intervalle de confiance à 95% pour notre estimation est [0.163,0.992]. C'est ce que ggplot a représenté.