Contents

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.

Technical information on the computer on which the analysis is run

We will be using the Python 3 language using the pandas, statsmodels, and numpy library.

```
def print_imported_modules():
    import sys
    for name, val in sorted(sys.modules.items()):
        if(hasattr(val, '__version__')):
            print(val.__name__, val.__version__)
#         else:
#             print(val.__name__, "(unknown version)")
def print_sys_info():
    import sys
    import platform
    print(sys.version)
    print(platform.uname())

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
```

```
print_sys_info()
print_imported_modules()
```

```
3.10.2 (main, Jan 15 2022, 19:56:27) [GCC 11.1.0]
uname_result(system='Linux', node='dellarch', release='5.15.24-1-lts', version='#1 SMP
IPython 8.0.1
IPython.core.release 8.0.1
PIL 9.0.1
PIL.Image 9.0.1
PIL._version 9.0.1
_csv 1.0
_ctypes 1.1.0
decimal 1.70
argparse 1.1
backcall 0.2.0
bottleneck 1.3.2
cffi 1.15.0
colorama 0.4.4
csv 1.0
ctypes 1.1.0
cycler 0.10.0
dateutil 2.8.2
decimal 1.70
decorator 5.1.1
defusedxml 0.7.1
distutils 3.10.2
executing 0.8.2
executing.version 0.8.2
jedi 0.17.2
joblib 1.1.0
joblib.externals.cloudpickle 2.0.0
joblib.externals.loky 3.0.0
json 2.0.9
kiwisolver 1.3.2
logging 0.5.1.2
matplotlib 3.5.1
matplotlib.backends.backend_qt 5.15.6
matplotlib.backends.qt_compat 5.15.6
matplotlib.backends.qt_editor._formlayout 1.0.10
```

```
numexpr 2.7.3
numpy 1.22.2
numpy.core 1.22.2
numpy.core._multiarray_umath 3.1
numpy.lib 1.22.2
numpy.linalg._umath_linalg 0.1.5
numpy.version 1.22.2
packaging 20.9
packaging.__about__ 20.9
pandas 1.4.1
parso 0.7.1
patsy 0.5.2
patsy.version 0.5.2
pexpect 4.8.0
pickleshare 0.7.5
platform 1.0.8
prompt_toolkit 3.0.28
psutil 5.9.0
ptyprocess 0.7.0
pure_eval 0.2.2
pure_eval.version 0.2.2
pygments 2.11.2
pyparsing 2.4.7
pytz 2021.3
re 2.2.1
scipy 1.8.0
scipy._lib._uarray 0.8.2+14.gaf53966.scipy
scipy._lib.decorator 4.0.5
scipy.integrate._dop 1.21.5
scipy.integrate._lsoda 1.21.5
scipy.integrate._ode $Id$
scipy.integrate._odepack  1.9
scipy.integrate._quadpack  1.13
scipy.integrate._vode 1.21.5
scipy.interpolate._fitpack  1.7
scipy.interpolate.dfitpack 1.21.5
scipy.linalg._fblas 1.21.5
scipy.linalg._flapack 1.21.5
scipy.linalg._flinalg 1.21.5
scipy.linalg._interpolative 1.21.5
```

```
scipy.optimize.__nnls 1.21.5
scipy.optimize._cobyla 1.21.5
scipy.optimize._lbfgsb 1.21.5
scipy.optimize._minpack  1.10
scipy.optimize._minpack2 1.21.5
scipy.optimize._slsqp 1.21.5
scipy.signal._spline 0.2
scipy.sparse.linalg._eigen.arpack._arpack 1.21.5
scipy.sparse.linalg._isolve._iterative 1.21.5
scipy.special._specfun 1.21.5
scipy.stats._mvn 1.21.5
scipy.stats._statlib 1.21.5
seaborn 0.11.2
seaborn.external.husl 2.1.0
six 1.16.0
sklearn 1.0.2
sklearn.base 1.0.2
sklearn.utils._joblib 1.1.0
stack_data 0.2.0
stack_data.version 0.2.0
statsmodels 0.13.1
statsmodels.__init__ 0.13.1
statsmodels.api 0.13.1
statsmodels.tools.web 0.13.1
threadpoolctl 2.2.0
traitlets 5.1.1
traitlets._version 5.1.1
urllib.request 3.10
wcwidth 0.2.5
zlib 1.0
```

Loading and inspecting data

Let's start by reading data.

```
data = pd.read_csv("https://app-learninglab.inria.fr/
    moocrr/gitlab/moocrr-session3/moocrr-
    reproducibility-study/raw/master/data/shuttle.csv")
print(data)
```

```
        Date  Count  Temperature  Pressure  Malfunction
```

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

```
import statsmodels.api as sm

data["Success"]=data.Count-data.Malfunction
data["Intercept"]=1

logmodel=sm.GLM(data['Frequency'], data[['Intercept','
    Temperature']],
                family=sm.families.Binomial(sm.
                    families.links.logit())).fit()

print(logmodel.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:              Frequency   No. Observations:                   23
Model:                            GLM   Df Residuals:                       21
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -3.9210
Date:               dim., 27 mars 2022   Deviance:                       3.0144
Time:                         12:46:08   Pearson chi2:                     5.00
No. Iterations:                      6   Pseudo R-squ. (CS):            0.04355
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      5.0850      7.477      0.680      0.496      -9.570      19.740
Temperature   -0.1156      0.115     -1.004      0.316      -0.341       0.110
==============================================================================
```

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:

```
X = np.linspace(start=30, stop=90, num=121)
res =logmodel.get_prediction(sm.add_constant(X))
predictions = res.summary_frame(alpha=0.05)
print(predictions.head())
```

```
       mean   mean_se  mean_ci_lower  mean_ci_upper
0  0.834373  0.229330       0.163069       0.992381
1  0.826230  0.234961       0.161330       0.991563
2  0.817774  0.240453       0.159601       0.990658
3  0.809002  0.245782       0.157884       0.989658
4  0.799911  0.250921       0.156176       0.988552
```

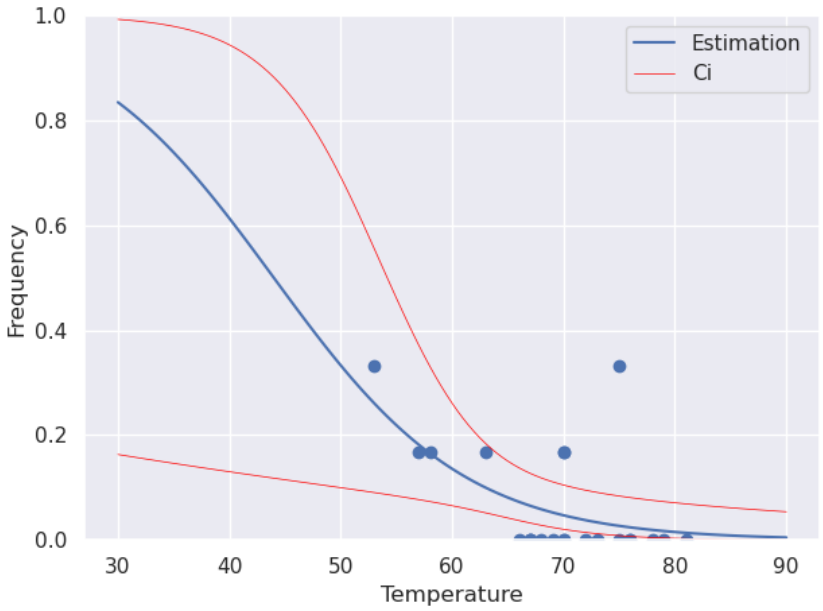
```
import matplotlib.pyplot as plt

plt.scatter(x=data["Temperature"],y=data["Frequency"])
plt.plot(X,predictions['mean'], label='Estimation')
plt.plot(X,predictions['mean_ci_lower'], color="red",
    label='Ci', linewidth=0.5)
```

```
plt.plot(X,predictions['mean_ci_upper'], color="red",
    linewidth=0.5)

plt.legend()
plt.grid(True)

plt.savefig(matplot_lib_filename)
matplot_lib_filename
```



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.**

I think I have managed to correctly compute and plot the uncertainty of my prediction. Contrary to the solution proposed by the mooc, the point 63 is not outside the confidence interval for me, I have a closer match to what is obtained with R. This is likely due to the difference between parametric confidence intervals (stasmodel and R confint) versus non-parametric confidence intervals (seaborn confidence interval). It seems logical that the non-parametric version is a little wider.