Challenger - Python - Emacs - Windows 7 64 bits

November 27, 2018

Contents

1 Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure **1**
1.1 Technical information on the computer on which the analysis is run 1
1.2 Loading and inspecting data 3
1.3 Logistic regression 5
1.4 Predicting failure probability 6
1.5 Computing and plotting uncertainty 8

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.

1.1 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()
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)]
uname_result(system='Windows', node='MGDONDON', release='7', version='6.1.7601', machine='AMD64'
IPython 6.5.0
IPython.core.release 6.5.0
_csv 1.0
_ctypes 1.1.0
decimal 1.70
argparse 1.1
backcall 0.1.0
colorama 0.3.9
csv 1.0
ctypes 1.1.0
cycler 0.10.0
dateutil 2.7.3
decimal 1.70
decorator 4.3.0
distutils 3.7.0
ipykernel 4.8.2
ipykernel._version 4.8.2
ipython_genutils 0.2.0
ipython_genutils._version 0.2.0
ipywidgets 7.4.0
ipywidgets._version 7.4.0
jedi 0.12.1
json 2.0.9
jupyter_client 5.2.3
jupyter_client._version 5.2.3
jupyter_core 4.4.0
jupyter_core.version 4.4.0
kiwisolver 1.0.1
logging 0.5.1.2
matplotlib 3.0.2
matplotlib.backends.backend_agg 3.0.2
numpy 1.15.4
numpy.core 1.15.4
numpy.core.multiarray 3.1
numpy.lib 1.15.4
numpy.linalg._umath_linalg b'0.1.5'
numpy.matlib 1.15.4
pandas 0.23.4
_libjson 1.33
parso 0.3.1
patsy 0.5.0
patsy.version 0.5.0
pickleshare 0.7.4
platform 1.0.8
prompt_toolkit 1.0.15
```

```
pygments 2.2.0
pyparsing 2.2.0
pytz 2018.5
re 2.2.1
scipy 1.1.0
scipy._lib.decorator 4.0.5
scipy._lib.six 1.2.0
scipy.fftpack._fftpack b'$Revision: $'
scipy.fftpack.convolve b'$Revision: $'
scipy.integrate._dop b'$Revision: $'
scipy.integrate._ode $Id$
scipy.integrate._odepack  1.9
scipy.integrate._quadpack  1.13
scipy.integrate.lsoda b'$Revision: $'
scipy.integrate.vode b'$Revision: $'
scipy.interpolate._fitpack  1.7
scipy.interpolate.dfitpack b'$Revision: $'
scipy.linalg 0.4.9
scipy.linalg._fblas b'$Revision: $'
scipy.linalg._flapack b'$Revision: $'
scipy.linalg._flinalg b'$Revision: $'
scipy.ndimage 2.0
scipy.optimize._cobyla b'$Revision: $'
scipy.optimize._lbfgsb b'$Revision: $'
scipy.optimize._minpack  1.10
scipy.optimize._nnls b'$Revision: $'
scipy.optimize._slsqp b'$Revision: $'
scipy.optimize.minpack2 b'$Revision: $'
scipy.signal.spline 0.2
scipy.sparse.linalg.eigen.arpack._arpack b'$Revision: $'
scipy.sparse.linalg.isolve._iterative b'$Revision: $'
scipy.special.specfun b'$Revision: $'
scipy.stats.mvn b'$Revision: $'
scipy.stats.statlib b'$Revision: $'
seaborn 0.9.0
seaborn.external.husl 2.1.0
seaborn.external.six 1.10.0
six 1.11.0
statsmodels 0.9.0
statsmodels.__init__ 0.9.0
traitlets 4.3.2
traitlets._version 4.3.2
urllib.request 3.7
zlib 1.0
zmq 17.1.2
zmq.sugar 17.1.2
zmq.sugar.version 17.1.2
```

1.2 Loading and inspecting data

Let's start by reading data.

```
data = pd.read_csv("https://app-learninglab.inria.fr/gitlab/moocrr-session1/moocrr-reproducibili
print(data)
```

```
Traceback (most recent call last):
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 1317, in do_open
    encode_chunked=req.has_header('Transfer-encoding'))
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 1229, in request
    self._send_request(method, url, body, headers, encode_chunked)
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 1275, in _send_request
    self.endheaders(body, encode_chunked=encode_chunked)
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 1224, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 1016, in _send_output
    self.send(msg)
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 956, in send
    self.connect()
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 1384, in connect
    super().connect()
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 932, in connect
    self._tunnel()
  File "c:\Program Files\Python\Python37\lib\http\client.py", line 911, in _tunnel
    message.strip()))
OSError: Tunnel connection failed: 407 Proxy Authorization Required

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "c:/Users/dondon/AppData/Local/Temp/babel-C4mAt5/python-fKMIca", line 1, in <module>
    data = pd.read_csv("https://app-learninglab.inria.fr/gitlab/moocrr-session1/moocrr-reproduci
  File "c:\Program Files\Python\Python37\lib\site-packages\pandas\io\parsers.py", line 678, in p
    return _read(filepath_or_buffer, kwds)
  File "c:\Program Files\Python\Python37\lib\site-packages\pandas\io\parsers.py", line 424, in _
    filepath_or_buffer, encoding, compression)
  File "c:\Program Files\Python\Python37\lib\site-packages\pandas\io\common.py", line 195, in ge
    req = _urlopen(filepath_or_buffer)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 222, in urlopen
    return opener.open(url, data, timeout)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 525, in open
    response = self._open(req, data)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 543, in _open
    '_open', req)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 503, in _call_chain
    result = func(*args)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 1360, in https_open
    context=self._context, check_hostname=self._check_hostname)
  File "c:\Program Files\Python\Python37\lib\urllib\request.py", line 1319, in do_open
    raise URLError(err)
urllib.error.URLError: <urlopen error Tunnel connection failed: 407 Proxy Authorization Required
```

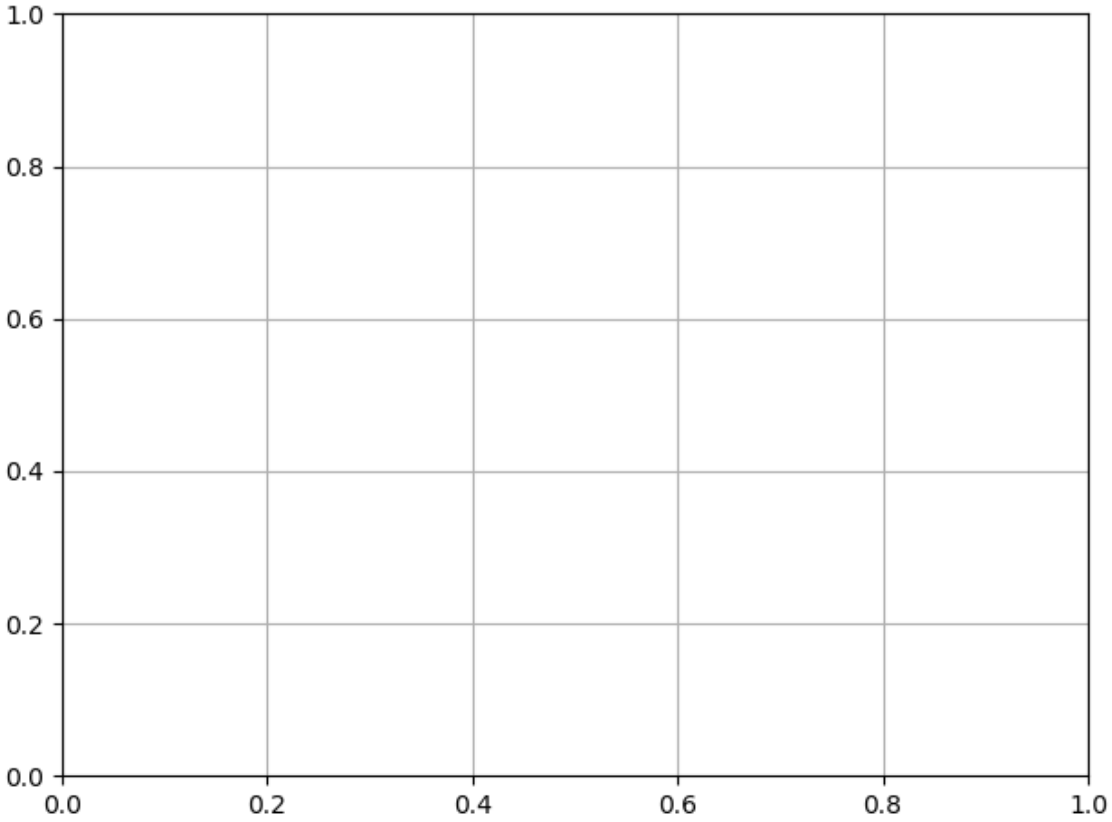
We know from our previous experience on this data set that filtering data is a really bad idea. We will therefore process it as such.

```
%matplotlib inline
pd.set_option('mode.chained_assignment',None) # this removes a useless warning from pandas

data["Frequency"]=data.Malfunction/data.Count
```

```
data.plot(x="Temperature",y="Frequency",kind="scatter",ylim=[0,1])
plt.grid(True)
plt.tight_layout()

plt.savefig(matplot_lib_filename)
matplot_lib_filename
```



1.3 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())
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "c:/Users/dondon/AppData/Local/Temp/babel-C4mAt5/python-QaCEOv", line 3, in <module>
    data["Success"]=data.Count-data.Malfunction
NameError: name 'data' is not defined
```

The maximum likelyhood estimator of the intercept and of Temperature are thus $\hat{\alpha} =$ **5.0850** and $\hat{\beta} =$ **-0.1156**. This **corresponds** to the values from the article of Dalal *et al.* The standard errors are

$s_{\hat{\alpha}} = 7.477$ and $s_{\hat{\beta}} = 0.115$, which is **different** from the **3.052** and **0.04702** reported by Dallal *et al.* The deviance is *3.01444* with **21** degrees of freedom. I cannot find any value similar to the Goodness of fit ($G^2 = \mathbf{18.086}$) reported by Dalal *et al.* There seems to be something wrong. Oh I know, I haven't indicated that my observations are actually the result of 6 observations for each rocket launch. Let's indicate these weights (since the weights are always the same throughout all experiments, it does not change the estimates of the fit but it does influence the variance estimates).

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

print(logmodel.summary())
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "c:/Users/dondon/AppData/Local/Temp/babel-C4mAt5/python-ODKOcF", line 1, in <module>
    logmodel=sm.GLM(data['Frequency'], data[['Intercept','Temperature']],
NameError: name 'data' is not defined
```

Good, now I have recovered the asymptotic standard errors $s_{\hat{\alpha}} = \mathbf{3.052}$ and $s_{\hat{\beta}} = \mathbf{0.047}$. The Goodness of fit (Deviance) indicated for this model is $G^2 = \mathbf{18.086}$ with **21** degrees of freedom (Df Residuals).

I have therefore managed to fully replicate the results of the Dalal *et al.* article.

1.4 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:

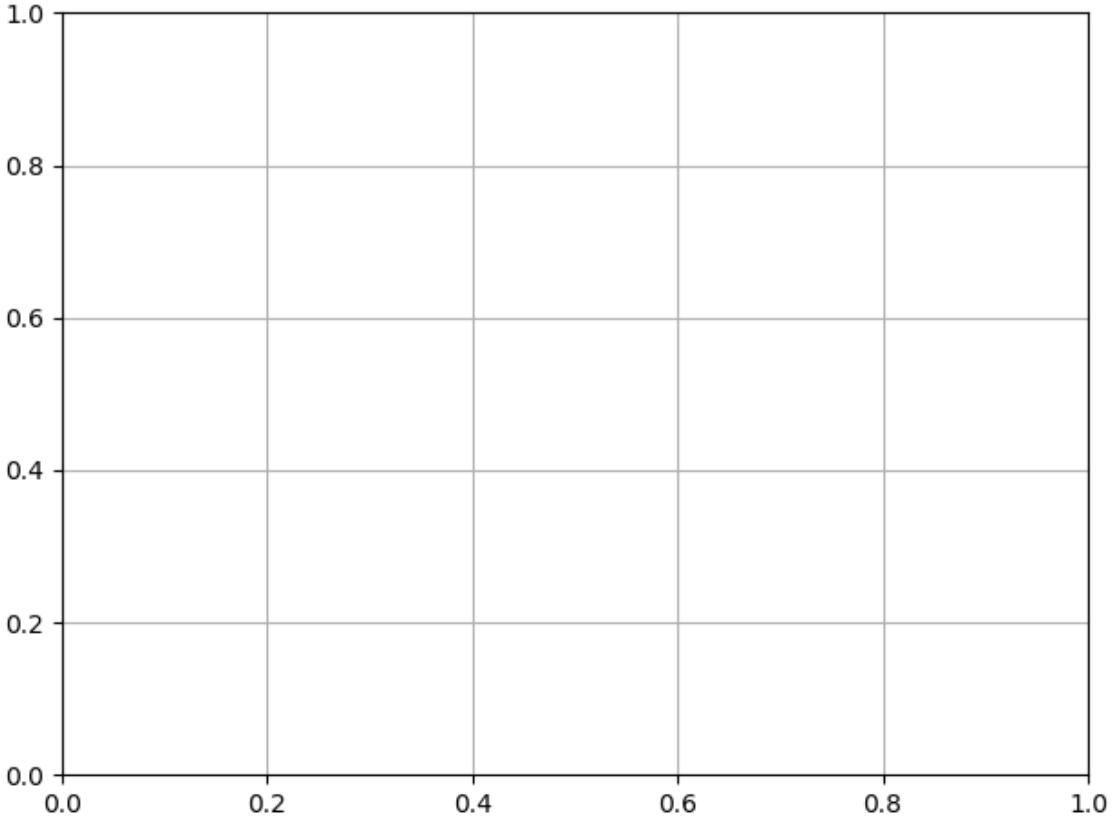
```
data_pred = pd.DataFrame({'Temperature': np.linspace(start=30, stop=90, num=121),
                          'Intercept': 1})
data_pred['Frequency'] = logmodel.predict(data_pred)
print(data_pred.head())
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "c:/Users/dondon/AppData/Local/Temp/babel-C4mAt5/python-eRVTd2", line 3, in <module>
    data_pred['Frequency'] = logmodel.predict(data_pred)
NameError: name 'logmodel' is not defined
```

```
%matplotlib inline


data_pred.plot(x="Temperature",y="Frequency",kind="line",ylim=[0,1])
plt.scatter(x=data["Temperature"],y=data["Frequency"])
plt.grid(True)

plt.savefig(matplot_lib_filename)
matplot_lib_filename
```



La fonction `logmodel.predict(data_pred)` ne fonctionne pas avec les dernières versions de pandas (Frequency = 1 pour toutes les températures).

On peut alors utiliser le code suivant pour calculer les prédictions et tracer la courbe :

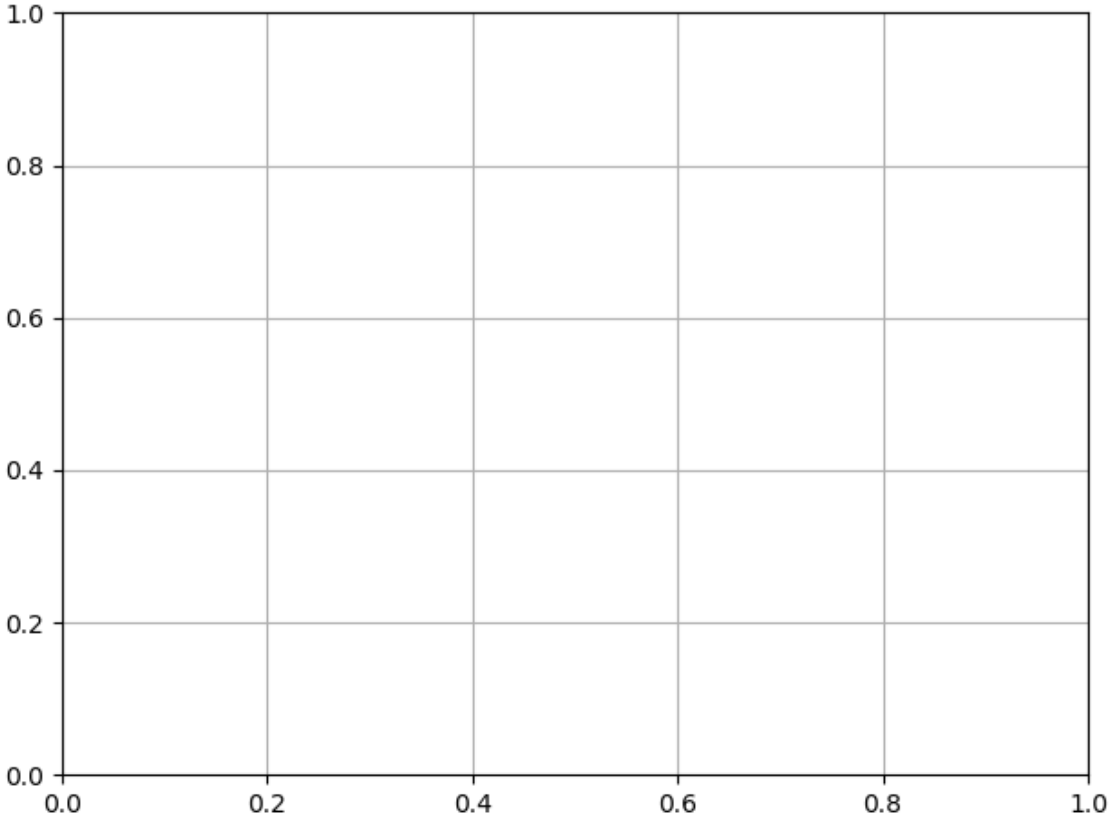
```
# Inspiring from http://blog.yhat.com/posts/logistic-regression-and-python.html
def logit_inv(x):
    return(np.exp(x)/(np.exp(x)+1))

data_pred['Prob']=logit_inv(data_pred['Temperature'] * logmodel.params['Temperature'] +
                            logmodel.params['Intercept'])
print(data_pred.head())

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "c:/Users/dondon/AppData/Local/Temp/babel-C4mAt5/python-u7RGBJ", line 5, in <module>
    data_pred['Prob']=logit_inv(data_pred['Temperature'] * logmodel.params['Temperature'] +
NameError: name 'logmodel' is not defined

%matplotlib inline
data_pred.plot(x="Temperature",y="Prob",kind="line",ylim=[0,1])
plt.scatter(x=data["Temperature"],y=data["Frequency"])
plt.grid(True)

plt.savefig(matplot_lib_filename)
matplot_lib_filename
```



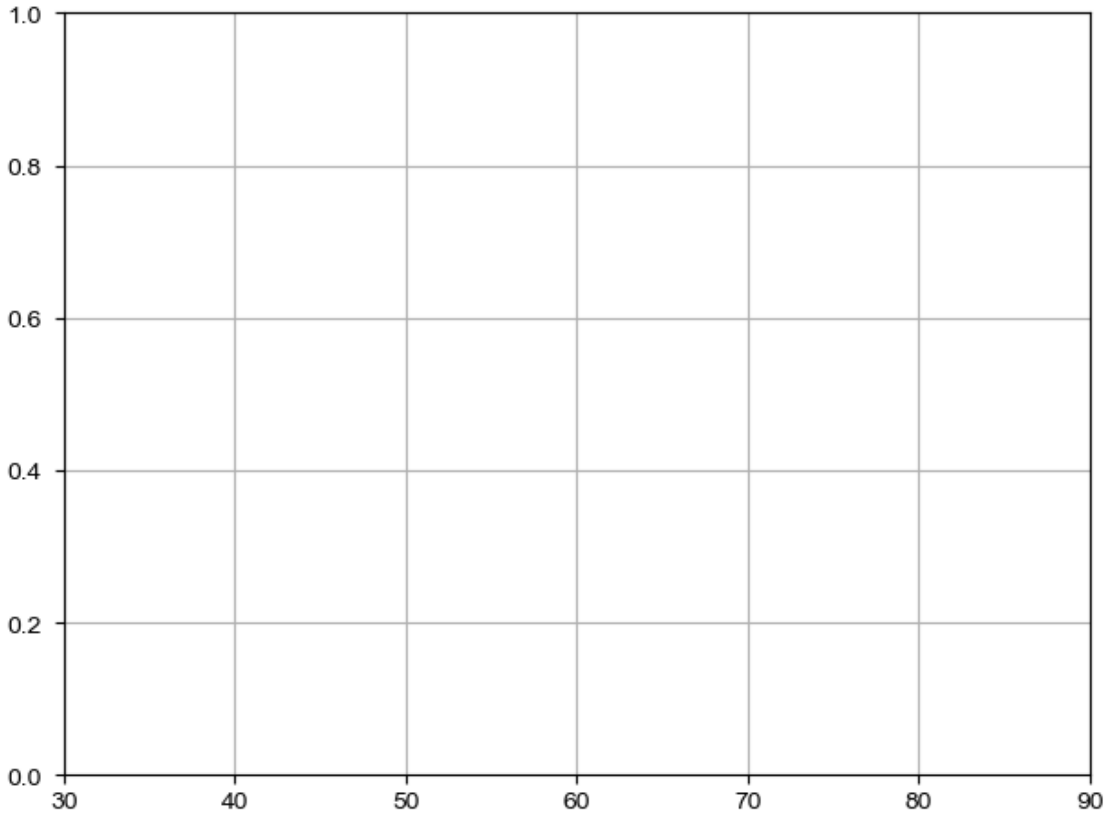
This figure is very similar to the Figure 4 of Dalal *et al.* **I have managed to replicate the Figure 4 of the Dalal *et al.* article.**

1.5 Computing and plotting uncertainty

Following the documentation of [Seaborn](`https://seaborn.pydata.org/generated/seaborn.regplot.html`), I use regplot.

```
sns.set(color_codes=True)
plt.xlim(30,90)
plt.ylim(0,1)
sns.regplot(x='Temperature', y='Frequency', data=data, logistic=True)
plt.show()

plt.savefig(matplot_lib_filename)
matplot_lib_filename
```



I think I have managed to correctly compute and plot the uncertainty of my prediction. Although the shaded area seems very similar to [the one obtained by with R](`https://app-learninglab.inria.fr/gitlab/moocrr-session1/moocrr-reproducibility-study/raw/5c9dbef11b4d76`
`challenger.pdf`), I can spot a few differences (e.g., the blue point for temperature 63 is outside)... Could this be a numerical error ? Or a difference in the statistical method ? It is not clear which one is "right".