Ivo Vladislavov Petrov

Principles of Data Science

Coursework Report

Data-Intensive Science

December 17, 2023

Word count: 2841

Contents

1 Introduction **1**

2 Section A **2**
2.1 Part (a) . . . 2
2.2 Part (b) . . . 2
2.3 Part (c) . . . 3
2.4 Part (d) . . . 4
2.5 Part (e) . . . 4

3 Section B **6**
3.1 Problem Introduction . . . 6
3.2 Hypothesis Testing Theory . . . 6
3.3 Hypothesis Testing Procedure . . . 8
3.3.1 Generation . . . 9
3.3.2 Data Point Threshold Estimation . . . 9
3.4 Part (f) . . . 11
3.5 Part (g) . . . 12

4 Conclusion **15**

Bibliography **15**

Appendix **16**

A Principles of Data Science Assignment - ivp24 **18**
A.1 Table of contents . . . 18
A.2 Requirements . . . 18
A.3 Setup . . . 18
A.4 Running the scripts . . . 19
A.4.1 Part (c) . . . 19
A.4.2 Part (d) . . . 19
A.4.3 Part (e) . . . 19
A.4.4 Parts (f, g) . . . 19
A.5 Features . . . 20
A.5.1 Parameters . . . 20
A.5.2 Initial fit . . . 20
A.5.3 Hypothesis testing . . . 20
A.6 Build status . . . 20
A.7 Credits . . . 20

List of Figures

2.1 Normalization verificiation in part (c) 4
2.2 Distribution and its components 4
2.3 Distribution sample and the MLE fit 5

3.1 Hypothesis test example 6
3.2 Chi squared convergence example 7
3.3 Weighted search example 11
3.4 Chi squared fit part (f) 12
3.5 Threshold finding procedure comparison 13
3.6 Chi squared fit for part (g) 14

Chapter 1

Introduction

Mathematical inference is a powerful statistical tool that has been a key part of scientific discovery for many centuries past. With the development of recent technology and Machine Learning, we have incredibly powerful tools to discern patterns from complex data. In this project, we focus on the development and the mathematical background behind a hypothesis-testing framework that is concerned with detecting weak signals in the presence of a high background noise.

Utilities of such kind are prevalent in most fields of science and technology, being key in detecting the Higgs boson [1], as well as isolating financial market signals [2]. In most such cases, it is crucial to understand the amount of data that is needed to reach a statistically significant conclusion on the existence of said signal. Therefore, our framework aims to formalize this process and give a numerically precise answer in the presence of a known distribution. In developing this framework, robustness was the primary concern, with maximal flexibility being given to the user, while also ensuring a high degree of automation.

Chapter 2

Section A

2.1 Part (a)

We are given that the probability density function (PDF) $p(M; f, \lambda, \mu, \sigma)$ is a linear combination of distributions $b(M; \lambda)$ and $s(M; \mu, \sigma)$. Both distributions are separately normalised. However, we need to mention that the exponential decay distribution is only defined for $M \in [0, +\infty)$, and therefore we assume that:

$$b(M; \lambda) = \begin{cases} 0 & M < 0 \\ \lambda e^{-\lambda * M} & M \geq 0 \end{cases}$$

We can then conclude that:

$$\int_{-\infty}^{+\infty} s(M; \mu, \sigma) dM = 1 \ , \ \int_{0}^{+\infty} b(M; \lambda) dM = 1$$

Then, we can check whether $p(M; f, \lambda, \mu, \sigma)$ is normalized by:

$$\begin{aligned} \int_{-\infty}^{+\infty} p(M; f, \lambda, \mu, \sigma) dM = \int_{-\infty}^{+\infty} (f s(M; \mu, \sigma) + (1-f) b(M; \lambda)) dM = \\ \int_{-\infty}^{+\infty} f s(M; \mu, \sigma) dM + \int_{-\infty}^{+\infty} (1-f) b(M; \lambda) dM = \\ f \int_{-\infty}^{+\infty} s(M; \mu, \sigma) dM + (1-f) \int_{-\infty}^{0} b(M; \lambda) dM + (1-f) \int_{0}^{+\infty} b(M; \lambda) dM = \\ f \times 1 + (1-f) \int_{-\infty}^{0} 0 dM + (1-f) \times 1 = \\ f + (1-f) = \\ 1 \end{aligned}$$

Therefore we conclude that $p(M; f, \lambda, \mu, \sigma)$ is normalized for $M \in [-\infty, +\infty]$.

2.2 Part (b)

We denote the cumulative density functions (CDFs) of the exponential decay distribution and the normal distribution as $B(M; \lambda)$ and $S(M; \mu, \sigma)$ respectively. Similarly to the previous part, we need to define the CDF of the exponential decay distribution below 0, in case either $\alpha < 0$ or $\beta < 0$:

$$F(M; \lambda) = \begin{cases} 0 & M < 0 \\ 1 - e^{-\lambda M} & M \geq 0 \end{cases}$$

Let us express the bounded distribution p' as:

$$p'(M; f, \lambda, \mu, \sigma, \alpha, \beta) = fN_s(\alpha, \beta)s(M; \mu, \sigma) + (1 - f)N_b(\alpha, \beta)b(M; \lambda))$$

. We have the condition that the total signal-to-background ratio must be $f/(1-f)$, or:

$$\begin{aligned}
\frac{\int_\alpha^\beta fN_s(\alpha, \beta)s(M; \mu, \sigma)}{\int_\alpha^\beta (1 - f)N_b(\alpha, \beta)b(M; \lambda)} = \frac{f}{1 - f} &\iff \\
\frac{\int_\alpha^\beta N_s(\alpha, \beta)s(M; \mu, \sigma)}{\int_\alpha^\beta N_b(\alpha, \beta)b(M; \lambda)} = 1 &\iff \\
\int_\alpha^\beta N_s(\alpha, \beta)s(M; \mu, \sigma) = \int_\alpha^\beta N_b(\alpha, \beta)b(M; \lambda)
\end{aligned}$$

Let us label the value of the 2 integrals as A. Then, because p' has to be normalized, it follows that:

$$\begin{aligned}
\int_\alpha^\beta p'(M; f, \lambda, \mu, \sigma, \alpha, \beta) = 1 &\iff \\
\int_\alpha^\beta fN_s(\alpha, \beta)s(M; \mu, \sigma) + (1 - f)N_b(\alpha, \beta)b(M; \lambda)) = 1 &\iff \\
f \int_\alpha^\beta N_s(\alpha, \beta)s(M; \mu, \sigma) + (1 - f) \int_\alpha^\beta N_b(\alpha, \beta)b(M; \lambda)) = 1 &\iff \\
f \times A + (1 - f) \times A = 1 &\iff \\
A = 1
\end{aligned}$$

Therefore, the signal-to-background ration will be $f/(1-f)$ if each of the signal and background components are individually normalized. Then, we can find:

$$\begin{aligned}
N_s^{-1} = \int_\alpha^\beta s(M; \mu, \sigma) = \\
\int_{-\infty}^\beta s(M; \mu, \sigma) - \int_{-\infty}^\alpha s(M; \mu, \sigma) = \\
S(\beta; \mu, \sigma) - S(\alpha; \mu, \sigma) = \frac{1}{2}\left[\text{erf}\left(\frac{\beta - \mu}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\alpha - \mu}{\sigma\sqrt{2}}\right)\right]
\end{aligned}$$

and analogically:

$$N_b^{-1} = B(\beta; \lambda) - B(\alpha; \lambda) = \begin{cases} 1 - e^{-\lambda\beta} & \alpha \le 0 < \beta \\ e^{-\lambda\alpha} - e^{-\lambda\beta} & 0 < \alpha \end{cases}$$

Here we assume that $\beta > 0$, as there is no background otherwise, which will violate the $f/(1-f)$ signal-to-background ration. Finally, we can write p' as:

$$p'(M; f, \lambda, \mu, \sigma, \alpha, \beta) = \begin{cases} f \times \dfrac{2s(M; \mu, \sigma)}{\text{erf}(\frac{\beta-\mu}{\sigma\sqrt{2}}) - \text{erf}(\frac{\alpha-\mu}{\sigma\sqrt{2}})} + (1 - f) \times \dfrac{b(M; \lambda)}{1 - e^{-\lambda\beta}} & \alpha \le 0 < \beta \\ f \times \dfrac{2s(M; \mu, \sigma)}{\text{erf}(\frac{\beta-\mu}{\sigma\sqrt{2}}) - \text{erf}(\frac{\alpha-\mu}{\sigma\sqrt{2}})} + (1 - f) \times \dfrac{b(M; \lambda)}{e^{-\lambda\alpha} - e^{-\lambda\beta}} & 0 < \alpha \end{cases}$$

2.3 Part (c)

In order to determine whether the normalization computed is correct, we substitute the formulas from part (b), implemented in the purest possible way. We use the `scipy.stats` library to

compute the PDF of the normal and exponential distributions due to their higher numerical precision, but I do not use them for computing the normalization constants to verify the correctness of the above formulae. To finally verify the resulting PDF, We compute the integral between the boundaries (α, β) using the `scipy.integrate.quad` method, and verify whether the area under the curve is close to 1 (with some room for numerical errors). A sample run of the script (`part_c_check.py`) can be seen in Figure 2.1.

```
(base) root@1befc5189e95:/ivp24# python -m src.runnable_scripts.part_c_check
Calculated area for parameter set [0.1   0.5   5.28  0.018 5.    5.6  ]: 1.00000
Calculated area for parameter set [ 0.3  0.4  3.   3.5  2.  10. ]: 1.00000
Calculated area for parameter set [ 0.1  1.   7.   2.  12.  12.1]: 1.00000
Calculated area for parameter set [ 0.5  10.    2.32  0.3   1.    2.  ]: 1.00000
Calculated area for parameter set [ 0.99  0.33  7.77 10.    0.    1.23]: 1.00000
```

Figure 2.1: When running the validation script, we use different sets of parameters with reasonable values to determine the computation robustness. As expected, all selected values converge to an area of 1, showing the distribution is properly normalized.

2.4 Part (d)

As we determined in parts (b) and (c), we can construct the PDF of the combined distribution by simply weighing the PDFs of the signal and background distributions through f, as seen in Figure 2.2.

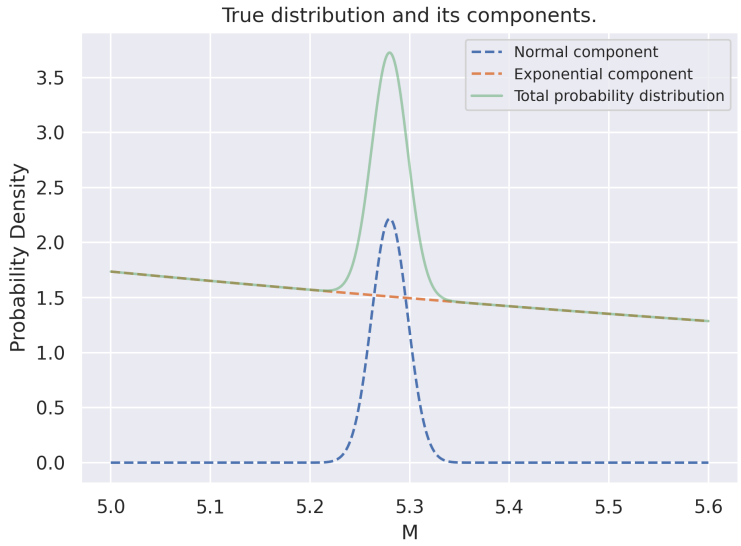

Figure 2.2: Parameters used for the given distribution are $f = 0.1, \lambda = 0.5, \mu = 5.28, \sigma = 0.018, \alpha = 5, \beta = 5.6$. The components seem to visually match the combined distribution. Furthermore, the background does dominate the signal, confirming visually that f is correctly implemented as the signal-to-noise ratio.

2.5 Part (e)

For generating a sample, I utilise the Accept-Reject method, which can be described as follows:

Definition 2.5.1 Accept-Reject method Given a range of values (α, β), and a probability density function f, we iteratively sample for N values as so:

- Find the maximal value of f - f_{max} by scanning the range (α, β) using a bruteforce algorithm.
- Then, iteratively sample x from a uniform distribution in the range (α, β).
- Also, sample y from a uniform distribution in the range $(0, f_{max})$.
- If $y \leq f(x)$, we accept x as a sample, otherwise we reject it.

In this case, since it is only a single estimation task, We use `Iminuit`'s[3] implementation of **Unbinned Maximum Likelihood Estimation**. In order to ensure the fit will make physical sense, we set limits on the parameter f, so that it lies in the range $(0, 1)$. For the purpose of automating the script, the initial values are set as the true parameters for the distribution. This ensures the fit will be numerically correct. We then obtain a precise estimation of the parameters, as seen on Figure 2.3. We can then conclude that the limits set are not detrimental to a precise fit, at least in the case of a big dataset.

```
(base) root@1befc5189e95:/ivp24# python -m src.runnable_scripts.part_e_sampling
Estimated value for f: 0.103 ± 0.002
Estimated value for lam: 0.516 ± 0.019
Estimated value for mu: 5.279 ± 0.000
Estimated value for sigma: 0.019 ± 0.000
Estimated value for alpha: 5.000 ± 0.050
Estimated value for beta: 5.600 ± 0.056
```

Distributions of the sample and estimated fit.

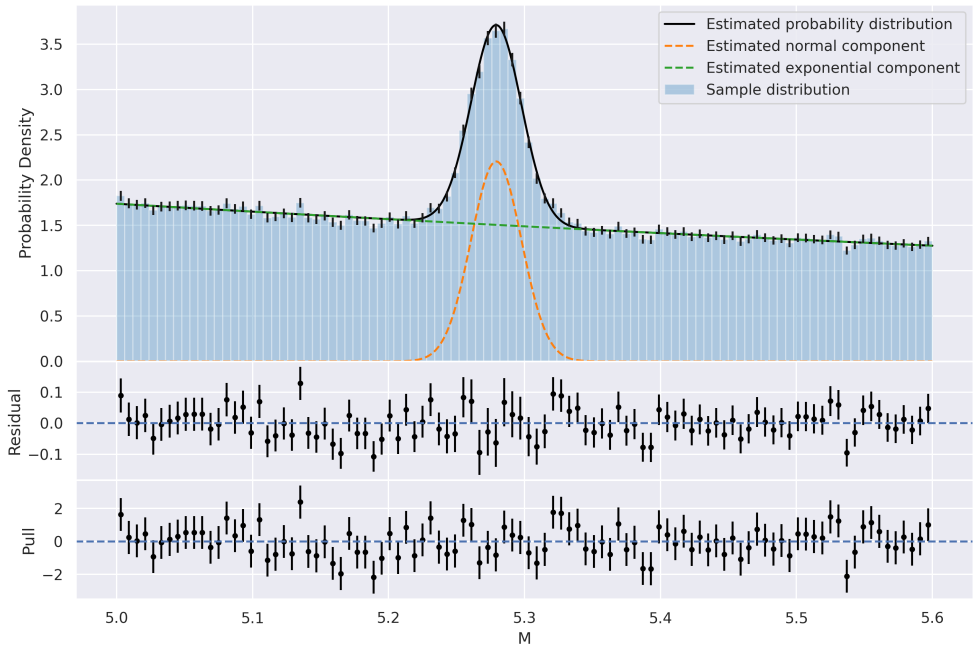

Figure 2.3: The resulting estimates cover the expected values, which are the same ones used in part (d), as seen on the top. Furthermore, the fit matches the samples presented well visually (bottom). Below the fit we observe the residual and pull plots. As can be seen, the pull plot looks normally distributed, meaning that no heteroskedasticity is present.

Chapter 3

Section B

3.1 Problem Introduction

For both part (f) and (g), we would like to discover a signal, appearing in contrast to an underlying distribution. This means we would like to set up a hypothesis-testing framework to differentiate between the two given distributions.

By knowing the true distribution we will be able to generate as many samples as we wish, in order to construct a test statistic distribution under the null and alternate hypotheses (hereafter referenced as H_0 and H_1 respectively), an example of which is shown in Figure 3.1. Another point

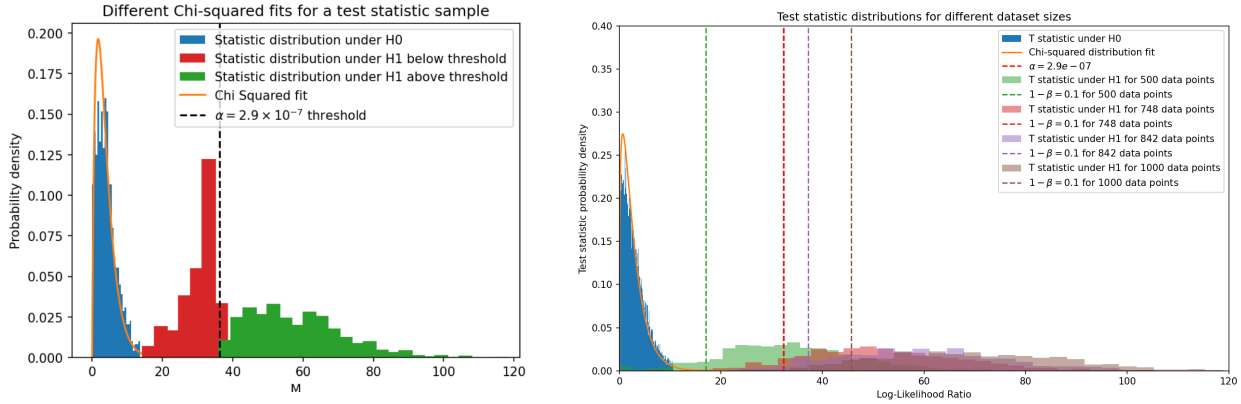

Figure 3.1: An example of a hypothesis test can be seen on the left-hand side. The respective type 2 error can be calculated as the fraction of the red area over the total area. The plot on the right further shows that, as expected, an increase in the dataset size shifts the distribution, as well as the 10-th% percentile mark to the right. With that in mind, we expect that the proposed framework will be guaranteed to converge for some dataset size.

we would like to make is that we change the implementations of computing any distribution property to use more efficient methods from the `numba-stats` [4] module, instead of the naive methods we designed for the previous parts. That said, no parallelisation has been utilised due to the value-sensitive nature of the functions. We will discuss below precisely how to reach the threshold within some level of accuracy.

3.2 Hypothesis Testing Theory

For separating the two hypotheses, we utilise the Log-Likelihood Ratio (LLR) test statistic. Not only is it the most powerful statistic, as shown by the Neyman-Pearson lemma, but it also

holds the following result through Wilks' theorem:

Definition 3.2.1 Wilks' Theorem: For parameter space Θ and null hypothesis $\Theta_0 \subset \Theta$, the LLR $\lambda_{LR} = -2\ln\left[\frac{\max_{\theta\epsilon\Theta_0}\mathcal{L}(\theta;X)}{\max_{\theta\epsilon\Theta}\mathcal{L}(\theta;X)}\right]$ is χ^2_k-distributed with degrees of freedom $k = df_{alt} - df_{null}$, where df_{alt}, df_{null} are the degrees of freedom of the alternate and null hypothesis respectively.

Remark: As shown by **(author?)** [5], one of the crucial assumptions of Wilks' theorem is that the true values of the unknown parameters in the null hypothesis lie in the interior[6] of the defined parameter space. That should also be the case for any nuisance parameters. However, in our case this assumption breaks for f, which under the null hypothesis has a true value of $f = 0$, lying on the boundary of the space. Furthermore, α and β often converge to the minimum and maximum of the sample respectively. As described by **(author?)** [7], this makes the underlying distribution more complicated than a χ^2_k. While there are methods to mitigate the effects, for the sake of simplicity we will continue with Wilks' theorem's assumption, but with a floating value for k.

In the case of part (f), the null hypothesis contains 1 free parameter λ (or 3 if we allow α, β to float), while the alternate contains 4 - λ, f, μ, σ (or 6 in the latter case). In any case, we expect the LLR to be χ^2_k-distributed with $k = 3$. However, the parameters can be correlated, which might reduce the actual value of the χ^2 d.o.f. fit.

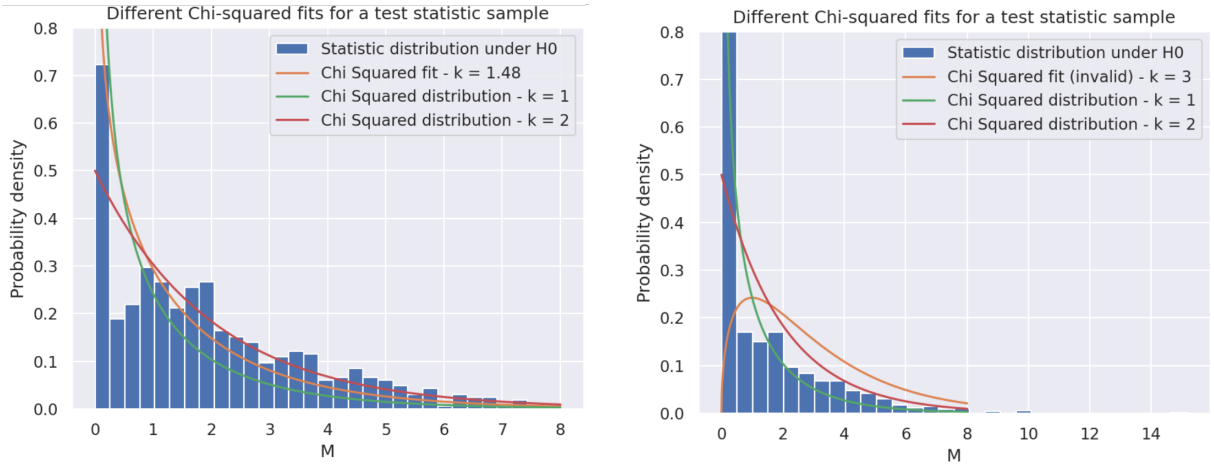

Figure 3.2: We can see on the left that a reasonable fit is achievable for the χ^2_k distribution. However, if we do not exclude invalid fits, resulting in a poorly behaved LLR, we are not able to obtain convergence of the fit, as shown on the right hand side.

Furthermore, given the numerical nature of the optimization algorithm, we can expect that due to numerical imprecision, any inference for the d.o.f. might be heavily influenced by incorrect LLR computation, as seen on Figure 3.2. As such, we use the following simple fact:

Lemma 3.2.2 For any sample X, parameter space Θ and null hypothesis $\Theta_0 \subset \Theta$, the LLR $\lambda_{LR} = -2\ln\left[\frac{\max_{\theta\epsilon\Theta_0}\mathcal{L}(\theta;X)}{\max_{\theta\epsilon\Theta}\mathcal{L}(\theta;X)}\right]$ is always non-negative.

Proof 3.2.3 Let $\theta_0 = \text{argmax}_{\theta \in \Theta_0} \mathcal{L}(\theta; X)$. From $\Theta_0 \subset \Theta$ it follows that $\theta_0 \in \Theta$ and therefore $\max_{\theta \in \Theta} \mathcal{L}(\theta; X) \geq \mathcal{L}(\theta_0; X) = \max_{\theta \in \Theta_0} \mathcal{L}(\theta; X)$. Therefore:

$$\lambda_{LR} = -2 \ln \left[\frac{\max_{\theta \in \Theta_0} \mathcal{L}(\theta; X)}{\max_{\theta \in \Theta} \mathcal{L}(\theta; X)} \right] = -2 \ln \left[\frac{\mathcal{L}(\theta_0; X)}{\max_{\theta \in \Theta} \mathcal{L}(\theta; X)} \right] \geq -2 \ln \left[\frac{\mathcal{L}(\theta_0; X)}{\mathcal{L}(\theta_0; X)} \right] = -2 \ln 1 = 0$$

Given Lemma 3.2.2, we know that the LLR is always non-negative. However due to the `Migrad`[8] algorithm being a numerical approximation, this is not always guaranteed. Most of the error is caused by the algorithm finding only local minima, which can be improved through better initialization. However, it is difficult to create a perfect initialization with no prior knowledge for every single sample. Therefore, we ignore any sample that produces a negative LLR value, as that is caused by an invalid fitting procedure, i.e. the `Iminuit` gets stuck in a local minimum when fitting the alternate hypothesis model. We are aware this might cause a bias in the χ^2_k distribution fit, however including or capping the values also produces significant errors.

3.3 Hypothesis Testing Procedure

With the necessary background, we can now determine how to ascertain a discovery within some confidence level. Let us assume we want to determine whether a discovery is made at a rate of β, with a discovery threshold of α. For the given tasks, this would correspond to $\alpha = 2.9 \times 10^{-7}, \beta = 90\%$. If we are given the PDFs of the test statistic under the null hypothesis $p(T|H_0)$ and under the alternative hypothesis $p(T|H_1)$, we can infer the corresponding p-values by setting $\alpha = 2.9 \times 10^{-7}$ as constant.

1. Generate a sample from a given parameter set of the alternate hypothesis ($s(M; \theta)$).
2. Fit a model from the null hypothesis for the given sample and use the parameters as our null hypothesis generative distribution ($b(M; \theta)$).
3. For a given number of iterations (the default being 1000), execute steps 4-7.
4. Generate a sample from H0 and fit both models using MLE.
5. Measure the test statistic under the above assumption, and store it.
6. Generate a sample from H1 and fit both models using MLE.
7. Measure the test statistic under the above assumption, and store it.
8. After generating the probability, fit a χ^2_k distribution using `Iminuit`'s MLE on the samples generated from the null hypothesis to find the distribution $p(T|H_0)$.
9. Find the value of T such that $\int_T^\infty p(T|H_0) = \alpha$
10. Measure the Type-2 Error by finding the number of H_1-generated sample below the T threshold. If the measured error is smaller than $1 - \beta$, we conclude that the number of data points we used are sufficient.

3.3.1 Generation

Generating the data can be done using the accept-reject method, described in Part (e). However, this tends to be quite slow with a naive Pythonic implementation. Therefore, we opted to use a different approach to generating a sample from a mixture of distributions:
Given a distribution with a PDF of the form $p(M;\theta) = \sum_{i=1}^{N} f_i p_i(M;\theta_i)$, where $\sum_{i=1}^{N} f_i = 1$, p_i is a normalized PDF and $\theta_i \subset \theta$, the following procedure produces a valid sample of size M:

1. Generate M samples $S_1, S_2, S_3, ..., S_N$ from each distribution using an efficient implementation. This can be done using the `rvs` function of a distribution module.
2. Generate M samples $u_1, u_2, ...$ from a uniform distribution in the range (0, 1).
3. For each distribution, let $S_i' = \{S_i[j] | \sum_{k=1}^{i-1} f_k < u_j \leq \sum_{k=1}^{i} f_k\}$
4. We obtain our final sample as $S = \bigcup_{i=1}^{N} S_i'$. This procedure produces a correct sample as each item i is selected with probability $P(s) = P(s \in (\sum_{k=1}^{i-1} f_k, \sum_{k=1}^{i} f_k)) = U(\sum_{k=1}^{i} f_k) - U(\sum_{k=1}^{i-1} f_k) = \sum_{k=1}^{i} f_k - \sum_{k=1}^{i-1} f_k = f_i$, with U being the CDF of the uniform distribution.

3.3.2 Data Point Threshold Estimation

The described procedure is defined for a given number of data points N and allows us to measure $\hat{\beta}(N)$. We would like to find N, such that $\hat{\beta}(N) \geq 90\%$ and $\forall N' < N. \hat{\beta}(N') < 90\%$. Because more samples lead to a greater certainty of the signal, we can assume $\hat{\beta}$ is increasing at every N. Therefore we can employ the following binary search procedure:

1. Find a range $[a, b]$, such that $\hat{\beta}(a) < 0.9 < \hat{\beta}(b)$.
2. If $b - a = 1$, we have found $N = b$.
3. Sample $\hat{\beta}(\frac{a+b}{2}) = c$.
4. If $c < 0.9$, repeat from step 2 in range $[c, b]$.
5. Otherwise, repeat from step 2 in range $[a, c]$.

This algorithm is guaranteed to converge, as the range size is halved every step. Moreover, it is quite efficient, such that for the true value of the threshold N_{true}, it is guaranteed to converge in $\mathcal{O}(\log(N_{true}))$ steps. However, this method is not guaranteed to converge to the same number of data points every time. Due to the randomness in sampling, each observation of $\hat{\beta}$ has an associated error. Therefore, we propose a different method for finding a **range** for the N threshold.
Instead of measuring $\hat{\beta}$, we measure the Type-2 Error we denote as $\beta^{\complement}$, with an estimator defined analogically as $\hat{\beta^{\complement}}(N)$. Then, we can see the following:

Statement 3.3.1 The error associated with the estimator $\hat{\beta^{\complement}}(N)$ is given by $\frac{\hat{\beta^{\complement}}(N)}{\sqrt{N_{iter}}}$, where N_{iter} is the number of algorithm iterations, described in Section 2.3

Proof 3.3.2 We assume that the number of bins measured below the relevant T-statistic T_0 is Poisson-distributed with a rate of λ. We denote this number as $N_{T<T_0}$.

$$E[N_{T<T_0}] = \frac{N_{iter} \times \hat{\beta^{\mathbb{C}}}(N)}{N_{iter}} = \lambda \text{ (Expectation of Poisson distribution)}$$

$$\sigma^2 = V[N_{T<T_0}] = \lambda = \frac{\hat{\beta^{\mathbb{C}}}(N)}{N_{iter}} \text{ (Variance of Poisson distribution)}$$

$$\Rightarrow \sigma = \sqrt{\frac{\hat{\beta^{\mathbb{C}}}(N)}{N_{iter}}}$$

Let us denote the CDF of the required Poisson distribution as $P(X; \lambda_p)$, where $\lambda_p = 0.1$. Then we can define the 'distance' between two measurements as so:

Definition 3.3.3 The 'Poisson distance' d_{poi} of two values of N: $N_a < N_b$ is given by:

$$d_{poi}(N_a, N_b) = P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_a); \lambda_p) - P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_b); \lambda_p)$$

An important property is that for any $N_c \in (N_a, N_b)$:

$$d_{poi}(N_a, N_b) = P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_a); \lambda_p) - P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_b); \lambda_p) = (P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_a); \lambda_p) -$$

$$P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_c); \lambda_p)) + (P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_c); \lambda_p) - P(N_{iter}\hat{\beta^{\mathbb{C}}}(N_b); \lambda_p) = d_{poi}(N_a, N_c) + d_{poi}(N_c, N_b)$$

.

We then define the following procedure, which we will call "Weighted Search", for finding a sensible range of values for N_{true}:

1. Find an initial range $[a, b] = [a_0, b_0]$, such that $\hat{\beta^{\mathbb{C}}}(a_0) > 0.1 > \hat{\beta^{\mathbb{C}}}(b_0)$.
2. Until $\hat{\beta^{\mathbb{C}}}(a) + \sqrt{\frac{\hat{\beta^{\mathbb{C}}}(a)}{N_{iter}}} < 0.1 < \hat{\beta^{\mathbb{C}}}(b) - \sqrt{\frac{\hat{\beta^{\mathbb{C}}}(b)}{N_{iter}}}$, i.e. 0.1 is within the error range, repeat steps 3 - 5.
3. Sample $\hat{\beta^{\mathbb{C}}}(\frac{a+b}{2}) = c$.
4. Set $a' = c - \frac{(b-a)}{2} \times \frac{d_{poi}(a,c)}{d_{poi}(a,b)}$ and $b' = c + \frac{(b-a)}{2} \times \frac{d_{poi}(c,b)}{d_{poi}(a,b)}$.
5. Measure $\hat{\beta^{\mathbb{C}}}(a')$ and $\hat{\beta^{\mathbb{C}}}(b')$.
6. After termination, find the widest range (a, b) among the attempted values, such that the condition in step 2 is satisfied.
7. Finally, quote the estimate $\hat{N}_{true} = \frac{(a+b)}{2}$ with an error of $\frac{(b-a)}{2}$

Remark: Note that after the transformation $b' - a' = \frac{(b-a)}{2} \times [\frac{d_{poi}(a,c)+d_{poi}(c,b)}{d_{poi}(a,b)}] = \frac{(b-a)}{2}$, i.e. we have the same asymptotic complexity as the binary search algorithm.

This method is visually similar to the Feldman-Cousins method, as can be seen on Figure 3.3. However, while Feldman-Cousins presents a continuous confidence belt, this one can have significant stochastic errors. The solutions presented above were our way of avoiding said stochastic errors. Furthermore, instead of having to scan the entire possible range, this algorithm, similar to the binary search one, attempts only $\mathcal{O}(\log(N_{true}))$ steps.

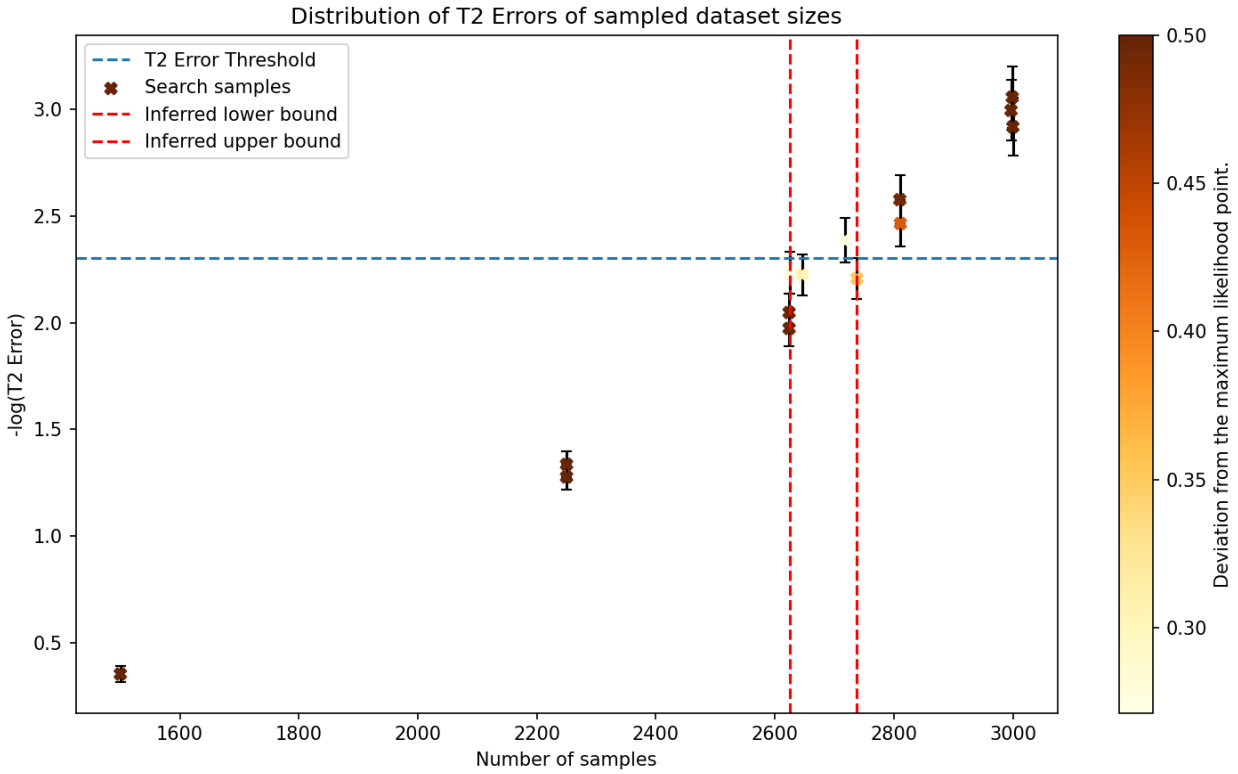

Figure 3.3: A sample run of the part (g) hypothesis test setup. Notice how, similar to the Feldman-Cousins method, all measurements are associated with a probable range. We take the largest range of dataset sizes, such that the threshold line passes through the error bars. However, this method is very likely to undercover, as precise evaluation will require a thorough space search.

> **Remark:** Note that there can be instances where visibly the error bars include the threshold line, but are not included in the final limit. For visualization purposes, we used a logarithmic scale on the y-axis, with an error $\Delta log(y) = \frac{\Delta y}{y}$. This is a first-order approximation, meaning that there exists a possibility that the error of the log covers the threshold, but the error of the original value does not.

Instructions on how to run these procedures can be found in the appendix, as detailed in the `README.md` file.

3.4 Part (f)

From Wilks' theorem, we define the parameter space as:

$$\Theta = \{(f, \lambda, \mu, \sigma, \alpha, \beta) | f \in (0,1) \wedge \lambda \in \mathcal{R} \wedge \mu \in \mathcal{R} \wedge \sigma \in \mathcal{R}^+ \wedge \alpha \in \mathcal{R} \wedge \beta \in \mathcal{R} \wedge \alpha < \beta\}$$

The subset, corresponding to the null hypothesis can be given as:

$$\Theta_0 = \{(f, \lambda, \alpha, \beta) | f = 0 \wedge \exists \mu, \sigma . (f, \lambda, \mu, \sigma, \alpha, \beta) \in \Theta\}$$

However, the parameters α, β and f lie on the parameter boundaries, which breaks the Wilks' theorem assumptions. Nevertheless, the results are still good enough to fit a χ^2_k distribution to obtain a value for T_0. In that case, we have to fit for the best degrees of freedom, as they are likely to be lower than 3, as shown in Figure 3.4.

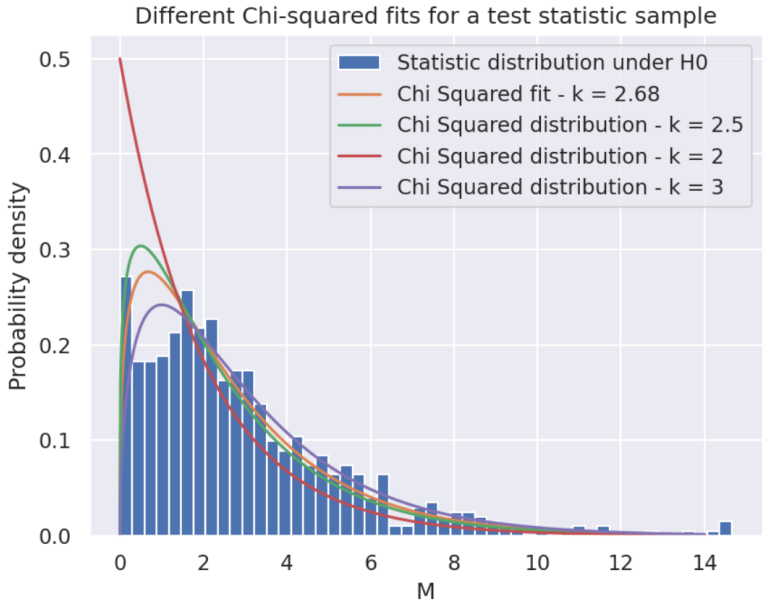

Figure 3.4: When fitting the test statistic distribution under H_0, we do not obtain a "degrees of freedom" value close to an integer. Instead, we observe the best fit lies somewhere between 2 and 3 degrees of freedom. It can be further noticed that the sample distribution does not resemble a χ^2_k one, due to the parameter space breaking the Wilks' theorem assumptions.

We use a binned fit with $\sqrt{N}$ bins for N datapoints. This makes it likely that most bins will have sufficient data inside, and there will also be sufficient points for the MLE fit. Furthermore, we include limits on the parameters to guarantee correct convergence. If we allow all parameters to float, the α and β parameters tend to cause the highest issues in convergence. When using a binned fit on a sample S, it can result in values for $\alpha > min(S)$ or $\beta < max(S)$, because the estimator takes only the bin values into account. Therefore, we set limits that $\alpha \in (-\infty, min(S))$ and $\beta \in (max(S), \infty)$. This also ensures that $\alpha \leq min(S) < max(s) \leq \beta$, meaning the distribution will be well defined. Furthermore, due to the physical definition of f being the signal-to-noise ratio, we need to establish that $f \in [0, 1]$. The lower bound can be empirically observed to cause instability, as no signal causes great errors in fitting the alternate hypothesis model. Therefore, we enforce that there should be at least 1% of signal, i.e. $f \in [0.01, 1]$. We leave all other parameters to float with no restrictions.

We then obtain a dataset size threshold of 750 ± 25. Furthermore, we compare the methods from other alternatives, in particular using the binary search algorithm, or a binned fit instead of the unbinned one, as seen in Figure 3.5. We can then conclude our methodology is consistent across all findings, meaning that we can continue using the recommended setup, in this case - a binned fit using a weighted search.

3.5 Part (g)

We use an identical setup as for part (f), with a different parameter space. The full parameter space can be specified as:

$$\Theta = \{(f_1, f_2, \lambda, \mu_1, \mu_2, \sigma, \alpha, \beta) | (f_1 \in (0,1)) \wedge (f_2 \in (0,1)) \wedge (f1 + f2 < 1) \wedge (\lambda \in \mathcal{R}) \wedge (\mu_1 \in \mathcal{R}) \wedge (\mu_2 \in \mathcal{R}) \wedge (\sigma \in \mathcal{R}^+) \wedge (\alpha \in \mathcal{R}) \wedge (\beta \in \mathcal{R}))\}$$

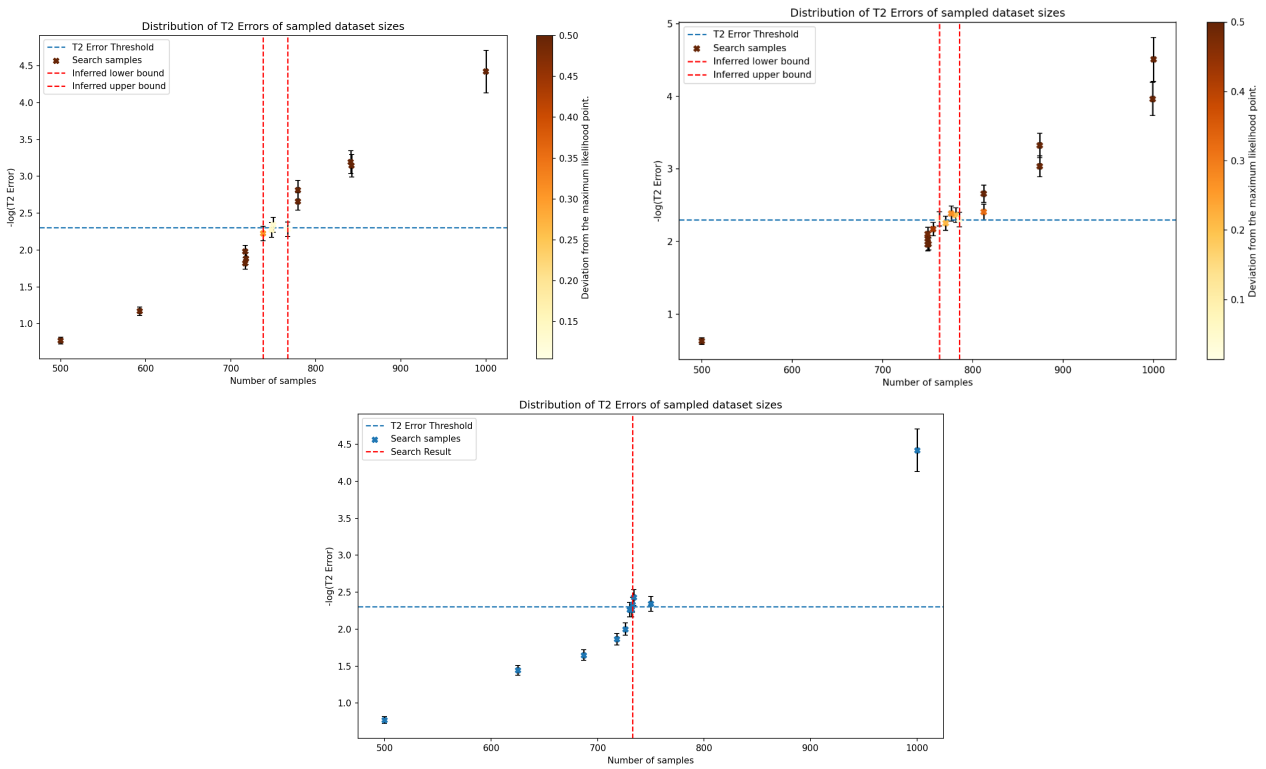

Figure 3.5: We can see all methods reach a similar and overlapping conclusion. Our binned procedure(top left) does not significantly differ from the unbinned one, with a mean difference of less than 20. The binary search method (bottom) reaches a smaller value than both, but it is still within a reasonable range. Note that it does not scan the entire probable space around the found value, meaning the result might be caused by random chance in the sampling stage.

The subset, corresponding to the null hypothesis can be given as:

$$\Theta_0 = \{(f_1, f_2, \lambda, \mu_1, \mu_2, \sigma, \alpha, \beta) | (f_2 = 0) \wedge \exists \mu_2 . (f_1, f_2, \lambda, \mu_1, \mu_2, \sigma, \alpha, \beta) \in \Theta\}$$

Similarly to before, we attempt to fit a χ^2_k distribution, with the expectation of 2 degrees of freedom (again, the fit is likely to result in a value less than 2, as seen in Figure 3.6).

We use a similar setup as before, with setting limits for the following parameters: $\alpha \in (-\infty, min(S))$, $\beta \in (max(S), \infty)$, $f_1 \in [0.01, 1]$, $f_2 \in [0.01, 1]$.

When executing the `part_g_ht.py` script, we can expect results in anywhere between 15 and 20 minutes, depending on the random seed we selected to run. We can then observe a threshold value of 2680 ± 60 datapoints required to detect the second signal. This is well over 3 times the amount we derived for the previous part, even though the weaker second signal is only twice as weak. This likely means it becomes much more difficult to detect new sources of signal, as the number of signals increases.

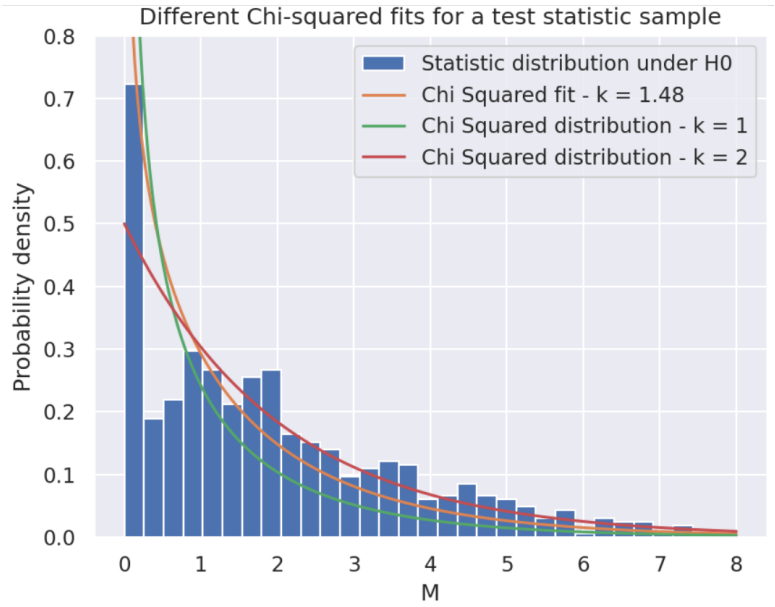

Figure 3.6: Similarly to part (f) when fitting the test statistic distribution under H_0, we do not obtain a "degrees of freedom" value close to an integer. Instead, we observe the best fit lies somewhere between 1 and 2 degrees of freedom.

Chapter 4

Conclusion

In this report we described a robust framework for estimating the number of points required for differentiating two different models. The specific context was detecting an additional signal in the presence of background noise, or an extra signal alongside the noise. As the problem was defined, it was highly dependent on random chance, as well as the robustness of the results of Wilks' theorem. However, the choices of parameter limits, and search algorithms, combined with the power of Maximum Likelihood Estimation, mitigated the problems associated with the imprecise assumptions. In particular, even though our parameter space violates one of the most important assumptions of Wilks' theorem, a slight relaxation of its results allowed us to produce valid predictions.

Flexible frameworks are provided in the repository linked to this report, which allow for different parameter configurations to be passed for evaluation. That said, very precise handling with the limits and parameters must be performed, as small changes might cause numerical instabilities in the models. Graphical representations support the findings in each script, in order to present the result in an understandable visual format. Many of the plots the scripts provide were used as figures in the report.

In conclusion, this project served as a good example of both the power of statistical methods, as well as their drawbacks. However, each problem was approachable by focusing on the practical aspect of the task, with which we were able to present accurate results.

Bibliography

[1] G. Aad and et al., "Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc," *Physics Letters B*, vol. 716, p. 1–29, Sept. 2012.

[2] K. Varaku, "Stock price forecasting and hypothesis testing using neural networks," 2019.

[3] H. Dembinski and P. O. et al., "scikit-hep/iminuit," Dec 2020.

[4] H. Dembinski, "Python package index - numba-stats."

[5] S. Algeri, J. Aalbers, K. D. Morå, and J. Conrad, "Searching for new phenomena with profile likelihood ratio tests," *Nature Reviews Physics*, vol. 2, p. 245–252, Apr. 2020.

[6] Wikipedia, "Interior (topology) — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Interior%20(topology)&oldid=1186135417`, 2023. [Online; accessed 08-December-2023].

[7] S. G. Self and K.-Y. Liang, "Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 605–610, 1987.

[8] F. James, "MINUIT Function Minimization and Error Analysis: Reference Manual Version 94.1," 1994.

Appendix

README.md

Appendix A

Principles of Data Science Assignment - ivp24

This repository contains a solution for the PDS coursework assignment.

A.1 Table of contents

1. Requirements[1]
2. Setup[2]
3. Running the scripts[3]
4. Features[4]
5. Frameworks[5]
6. Build status[6]
7. Credits[7]

A.2 Requirements

The user should have a version of Docker installed in order to ensure correct setup of environments. If that is not possible, please ensure you have all the specified packages in the `environment.yml` file correct.

A.3 Setup

To correctly set up the environment, we utilise a Docker image. To build the image before creating the container, you can run.

```
docker build -t ivp24_pds .
```

The setup image will also add the necessary pre-commit checks to your git repository, ensuring the commits work correctly.

[1] `#requirements`
[2] `#setup`
[3] `#running-the-scripts`
[4] `#features`
[5] `#frameworks`
[6] `#build-status`
[7] `#credits`

Afterwards, any time you want to use the code, you can launch a Docker container using:

`docker run --rm -ti ivp24_pds`

If you want to make changes to the repository, you would likely need to use your Git credentials. A safe way to load your SSH keys was to use the following command:

`docker run --rm -v <ssh folder on local machine>:/root/.ssh --name pds -ti ivp24_pds`

This copies your keys to the created container and you should be able to run all required git commands.

A.4 Running the scripts

A.4.1 Part (c)

In order to run the script for the relevant part, execute the following command while still in the main project directory:

`python -m src.runnable_scripts.part_c_check <config_file>`

A sample configuration file can be found in the `configs` folder. If a configuration is not specified, the default one will be used. If executed correctly, the program should return a list of parameters settings and the evaluated area for the pdf. If correctly implemented, all should return a value close to 1, showing a normalized probability function.

A.4.2 Part (d)

In order to run the script for the relevant part, execute the following command while still in the main project directory:

`python -m src.runnable_scripts.part_d_plot <config_file>`

A sample configuration file can be found in the `configs` folder. If a configuration is not specified, the default one will be used. If executed correctly, the program should produce a plot of the distribution with the given parameters, as well as the background and signal components.

A.4.3 Part (e)

In order to run the script for the relevant part, execute the following command while still in the main project directory:

`python -m src.runnable_scripts.part_e_sampling <config_file>`

A sample configuration file can be found in the `configs` folder. If a configuration is not specified, the default one will be used. If executed correctly, the program should produce a plot of the distribution. This includes a histogram of the produced samples, the estimated distribution, as well as the (unnormalized) components of the estimate.

A.4.4 Parts (f, g)

In order to run the script for the relevant parts, execute the following command while still in the main project directory:

`python -m src.runnable_scripts.part_f_ht <config_file>` for Part (f) or `python -m src.runnable_scripts.part_g_ht <config_file>` for Part (g)

A sample configuration file can be found in the `configs` folder. If a configuration is not specified, the default one will be used. The features of the configuration are displayed below. If

correctly executed, the program will give either a possible range for the number of data points to differentiate the hypotheses, or a conrete value (which might have a high uncertainty). It will then produce plots for the fit procedure and show examples of different test statistic distributions for a range of dataset sizes.

A.5 Features

While the implementations of parts (c), (d) and (e) are relatively straightforward, the configurations of (f) and (g) need to be explored further:

A.5.1 Parameters

This section details the true parameters of the alternate hypothesis. In the case of part (g), the signal with higher f value is taken as the generative one for the null hypothesis. For each part that includes:

f: $f, \lambda, \mu, \sigma, \alpha, \beta$, given respectively using the tags `f`, `lam`, `mu`, `sigma`, `alpha`, `beta`. It is necessary that $\alpha ¡ \beta$.

g: $f*1, f*2, \lambda, \mu*1, *mu*2, \sigma, \alpha, \beta$, given respectively using the tags `f1`, `f2`, `lam`, `mu1`, `mu2`, `sigma`, `alpha`, `beta`. It is necessary that $\alpha ¡ \beta$.

A.5.2 Initial fit

This section should contain identical members to the parameters one. The values are used to make an initial guess for the distribution when fitting the alternate hypothesis when the sample is generated from the null. Varying these numbers might ensure the fit is more stable.

A.5.3 Hypothesis testing

This section describes the parameters, with which the threshold search is executed. 1. `t1_error_rate` - the type 1 error rate, i.e. with what probability do we want to incorrectly accept the null hypothesis. 2. `t2_error_rate` - the type 2 error_rate, i.e. with what probability we want to incorrectly reject the alternate hypothesis. 3. `binned` - whether we want to use a binned or unbinned fit for the distribution estimation. 4. `n_bootstraps` - the number of samples produced for a single dataset size. 5. `algorithm` - whether to use the "binary search" or "weighted search" algorithm. The former will produce a single value, while the latter will quote range. It is notable that binary search is more efficient, but might also be inaccurate, as the decisions the algorithm takes are affected by random chance. This is less so the case for the latter, but is also an effect not to be neglected.

A.6 Build status

The build is still in the development phase and might be unstable.

A.7 Credits

The `.pre-commit-config.yaml` configuration file content has been adapted from the Research Computing lecture notes.