```
Main.var"workspace#2".AnalysisModule
```

```julia
# loading necessary packages
begin
    using Revise
    using StatsPlots, UnROOT, StatsBase, Polynomials, ColorSchemes, Suppressor  ,
    HypothesisTests, LaTeXStrings
    using FHist, DataFramesMeta, Distributions, DataFrames , RecipesBase
    using PlutoUI

    AM = include("./src/AnalysisModule.jl")
end

```

```julia
# setting plotting theme
begin
    gr()
    default(fmt = :jpg)
    theme(
        :dao;
        size           = (800, 800),
        legend         = :topleft,
        guidefontsize  = 16,
        tickfontsize   = 12,
        titlefontsize  = 16,
        legendfontsize = 12,
        left_margin    = 4Plots.mm,
        right_margin   = 8Plots.mm,
        top_margin     = 4Plots.mm,
        bottom_margin  = 6Plots.mm,
        dpi            = 200,
        :colorbar_titlefontsize => 20,
        widen = :false,
        :markerstrokewidth => 1,
        :markerstrokecolor => :black,
    );

end
```

Analysis notebook for improved $2\nu\beta\beta$ spectra.

The **goal** of this analysis is to compare 2 (slighty) different angular distributions.

The **methodology** used in the analysis is as follows. First, the angular distribution of $2\nu\beta\beta$ is given by the equation:

$$\frac{d\Gamma}{dcos(\theta)} \sim N(1 + K(\xi_{31}, \xi_{51})cos\theta)$$

,

equation (24) from paper by Ovidiu. In order to properly sample angular distributions, the parameter $K^{2\nu}$ must be provided. This parameter is defined (in eq. 26 from Ovidiu's paper) as:

$$K^{2\nu} = - \frac{H_0 + \xi_{31}H_2 + \frac{5}{9}\xi_{31}^2H_{22}+ (\frac{2}{9}\xi_{31}^2 + \xi_{51})H_{4}}{G_0 + \xi_{31}G_2 + \frac{1}{3}\xi_{31}^2G_{22} + (\frac{1}{3}\xi_{31}^2 + \xi_{51})G_4}$$

The single-electron energy distribution is in turn given by eq. 35 from Fedor's publication:

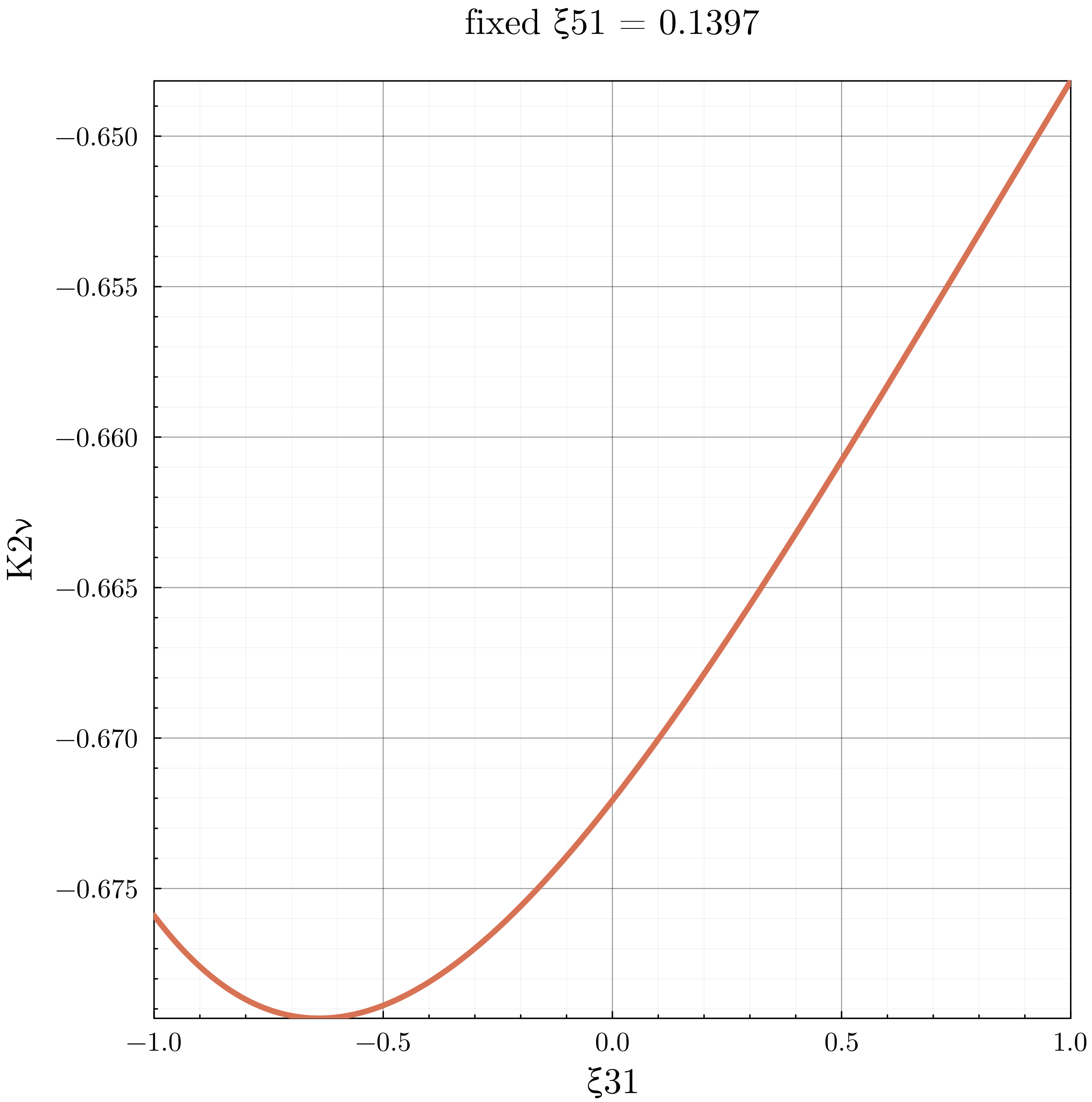
$$\frac{d\Gamma}{dT_{e}} \sim \frac{dG_0}{dT_{e}} + \xi_{31}\frac{dG_2}{dT_{e}} + \frac{1}{3}(\xi_{31})^2\frac{dG_{22}}{dT_{e}} + (\frac{1}{3}(\xi_{31})^2 + \xi_{51})\frac{dG_4}{dT_{e}}$$

.

Figure below shows $K(\xi_{31}, \xi_{51} = 0.1397)$:

```
md"""
Analysis notebook for improved ``2\nu\beta\beta`` spectra.

The **goal** of this analysis is to compare 2 (slighty) different angular distributions.

The **methodology** used in the analysis is as follows. First, the angular distribution of ``2\nu\beta\beta`` is given by the equation:

$\frac{d\Gamma}{dcos(\theta)} \sim N(1 + K (\xi_{31}, \xi_{51})cos\theta)$,

equation (24) from paper by [Ovidiu](https://www.mdpi.com/2218-1997/7/5/147). In order to properly sample angular distributions, the parameter $K^{2\nu}$ must be provided. This parameter is defined (in eq. 26 from Ovidiu's paper) as:

$K^{2\nu} = - \frac{H_0 + \xi_{31}H_2 + \frac{5}{9}\xi_{31}^2H_{22}+ (\frac{2}{9}\xi_{31}^2 + \xi_{51})H_{4}}{G_0 + \xi_{31}G_2 + \frac{1}{3}\xi_{31}^2G_{22} + (\frac{1}{3}\xi_{31}^2 + \xi_{51})G_4}$

The single-electron energy distribution is in turn given by eq. 35 from [Fedor's publication](https://journals.aps.org/prc/abstract/10.1103/PhysRevC.97.034315):

$\frac{d\Gamma}{dT_{e}} \sim \frac{dG_0}{dT_{e}} + \xi_{31}\frac{dG_2}{dT_{e}} + \frac{1}{3}(\xi_{31})^2\frac{dG_{22}}{dT_{e}} + (\frac{1}{3}(\xi_{31})^2 + \xi_{51})\frac{dG_4}{dT_{e}}$.

Figure below shows ``K(\xi_{31}, \xi_{51} = 0.1397)``:
```

```


"""
```


fixed ξ51 = 0.1397
K2ν
−0.650
−0.655
−0.660
−0.665
−0.670
−0.675
−1.0
−0.5
0.0
0.5
1.0
ξ31

In the initial stage of the analysis 2 sets of $2\nu\beta\beta$ spectra were simulated in Falaise, $1e8$ simulated events each. First, a spectrum (*standard spectrum - SM*) with $K = -0.88$ (taken from Decay0, which is currently in Falaise) and second spectrum is the refined one with $K = -0.6639; \xi31 = 0.37$. The simulated data were passed through flreconstruct and the following data cuts were applied:

1. `Two negatively charged tracks reconstructed,`
2. `Two vertices on the source foil, within given distance from each other,`
3. `` Sum of electron energies within the range: ``E_{sum} \in (0, 3500)~keV``, ``
4. `Two individual Optical Module hits,`
5. `Two associated Optical Module hits.`

From this, angular and single-electron energy distributions were obtained. The angle ϕ represents the **escape** angle - ie. the angle which can be measured with SuperNEMO, the angle between the two electrons at the moment they escape the source foil.

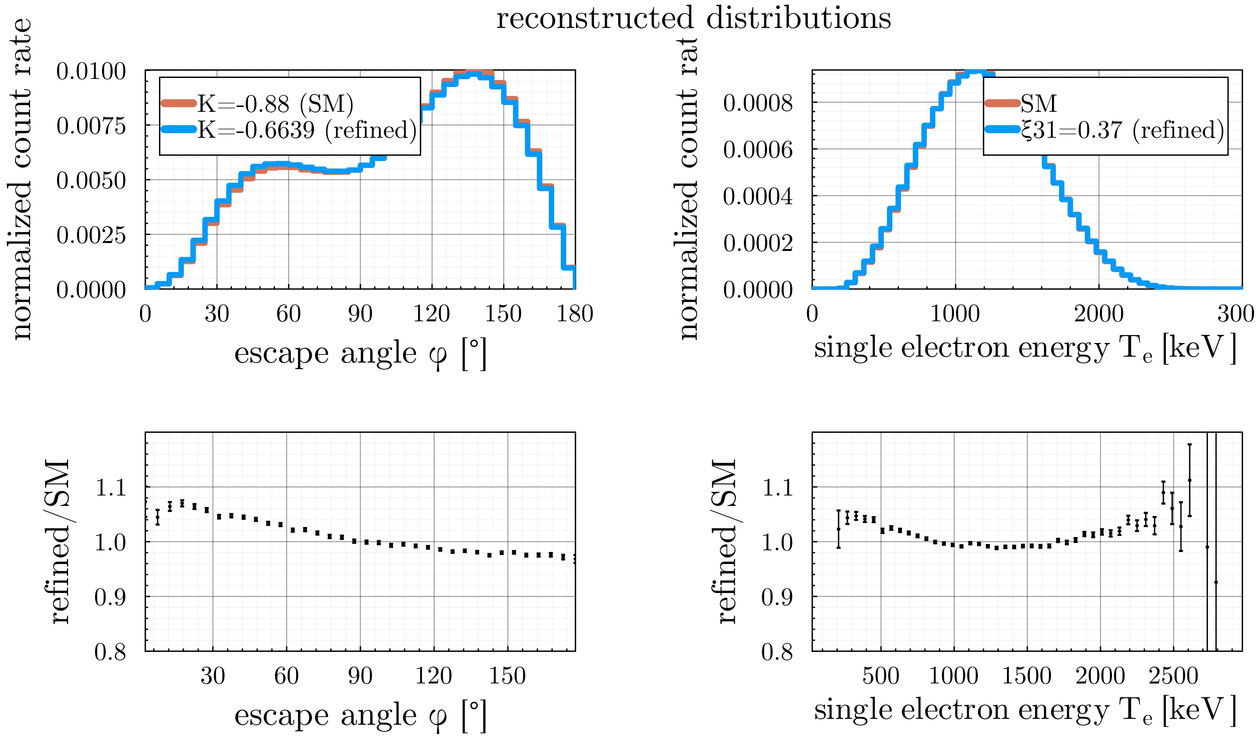
The two spectra are shown below.

```
▶ [152.847, 92.5176, 144.605, 41.9929, 130.177, 147.036, 51.0945, 44.9827, 125.083, 147.38
```

```
begin
    fName1 = "/home/shoram/Work/PhD_Thesis/Job17/Data/2vbb_Se82_Falaise_EneThetaPhi_1e8E.root"                           # root file for reference spectra
    fName2 = "/home/shoram/Work/PhD_Thesis/Job17/Data/G0-G4_xi31_037_kappa_m06639_1e8E.root"    # root file for compared spectra

    file1 = ROOTFile( fName1 )
    file2 = ROOTFile( fName2 )

    singleElectronEnergies1 = 
        AM.fill_from_root_file(file1, "tree","reconstructedEnergy1") .+ 
        AM.fill_from_root_file(file1, "tree", "reconstructedEnergy2") # vector of single-electron energies for reference spectrum
    singleElectronEnergies2 = 
        AM.fill_from_root_file(file2, "tree","reconstructedEnergy1") .+ 
        AM.fill_from_root_file(file2, "tree", "reconstructedEnergy2") # vector of single-electron energies for compared spectrum

    phi1 = AM.fill_from_root_file(file1, "tree","phi") # vector phi angles for reference spectrum
    phi2 = AM.fill_from_root_file(file2, "tree","phi") # vector phi angles for compared spectrum
end
```

```
60
```

```
begin
    Δϕ = 5 # setting bin width
    ΔE = 60
end
```



```
begin
    shPhi = histogram(
        [phi1, phi2],
        normed=:true,
        nbins=0:Δφ:180,
        label=["K=-0.88 (SM)" "K=-0.6639 (refined)"],
        xlabel="escape angle φ [°]" ,
        ylabel="normalized count rate",
        lw=4
    )

    resPhi = scatter(
        midpoints(0:Δφ:180),
        AM.get_residuals( phi1, phi2, 0:Δφ:180, true ),
        yerr=AM.get_residuals_errors( phi1, phi2, 0:Δφ:180, true ),
        mc=:black,
        label="",
        ylabel="refined/SM",
        xlabel="escape angle φ [°]",
        ms=1.5,
        ylims=(0.8, 1.2)
    )

    shEne = histogram(
        [singleElectronEnergies1, singleElectronEnergies2],
        normed=:true,
        nbins=0:ΔE:3000,
        label=["SM" "ξ31=0.37 (refined)"],
        xlabel=L"\textrm{single ~electron ~energy ~T_e ~[keV]}" ,
        ylabel="normalized count rate",
        legend=:best,
        lw=4
    )

    resEne = scatter(
        midpoints(0:ΔE:3000),
        AM.get_residuals(
            singleElectronEnergies1, singleElectronEnergies2, 0:ΔE:3000, true ),
        yerr=AM.get_residuals_errors(
            singleElectronEnergies1, singleElectronEnergies2, 0:ΔE:3000, true ),
        mc=:black,
        label="",
        ylabel="refined/SM",
        xlabel=L"\textrm{single ~electron ~energy ~T_e ~[keV]}",
        ylims=(0.8, 1.2),
        ms=1.5
    )

    plot(
        shPhi,
        shEne,
        resPhi,
        resEne,
        layout = grid(2,2),
        plot_title = "reconstructed distributions",
        size = (1400, 800),
        thickness_scaling = 1.4
    )
```

Looking at the two figures for the residuals, we can notice the following. First, for both distributions ϕ and Energy the biggest difference is at the low/high edges (though, for angles, we must take into account that the detector performance in such regions is lowered compared). Second, while the difference for the φ distribution appears to be more pronounced than that for energy.

In the next section three methods for spectral comparison will be presented:

1. Bin-by-bin (BBB)
2. χ^2 hypothesis test
3. Kolmogorov-Smirnov hypothesis test

The goal is to quantify the difference between the two angular distributions. We wish to provide the following answers:

1.
   ```
   How many events (how large a statistics) must be measured to be able to distinguish two spectral shapes within given ``n_{\sigma}`` confidence?
   ```
2.
   ```
   Is it feasible to obtain such statistics within the 5 year data-taking period of SuperNEMO? If not, what would have to be the parameters?
   ```
3.
   ```
   What is the sensitivity of SuperNEMO toward the given distribution?
   ```

Bin-By-Bin method

The main idea arises from comparing the difference between the two distributions in terms of bin heights (h_i^j; $k \in (1,2)$). First, we introduce the ratio $r_i \equiv \frac{h_i^{j,k}}{h_i^{k,j}}$, where whether i is in the numerator or j is determined dependent on whether $j < k$ or $j > k$, respectively. Furthermore, to be more *fair* in comparisons, we exchange $h_i^{j,k}$ with $\varepsilon_i^{j,k} \equiv \frac{h_i^{j,k}}{total\_number\_of\_events}$, ie. the normalized bin height. Thus, the ratio now becomes: $r_i \equiv \frac{\varepsilon_i^{j,k}}{\varepsilon_i^{k,j}}$

To determine ROI where the ratio r_i is most favourable for our purposes, that is the lowest number, we create *maps* of various ranges of ϕ. The maps are created by taking some range $\phi \in (\phi_{min}, \phi_{max})$, and calculating the respective r_i. The results are shown in the figure below. Each square in the figure represents a certain range, to be read out by the upper-left corner of the square. (That is, the range $\phi \in (0, \Delta\phi)$ is represented by the square in the down-left corner, the very first square.)

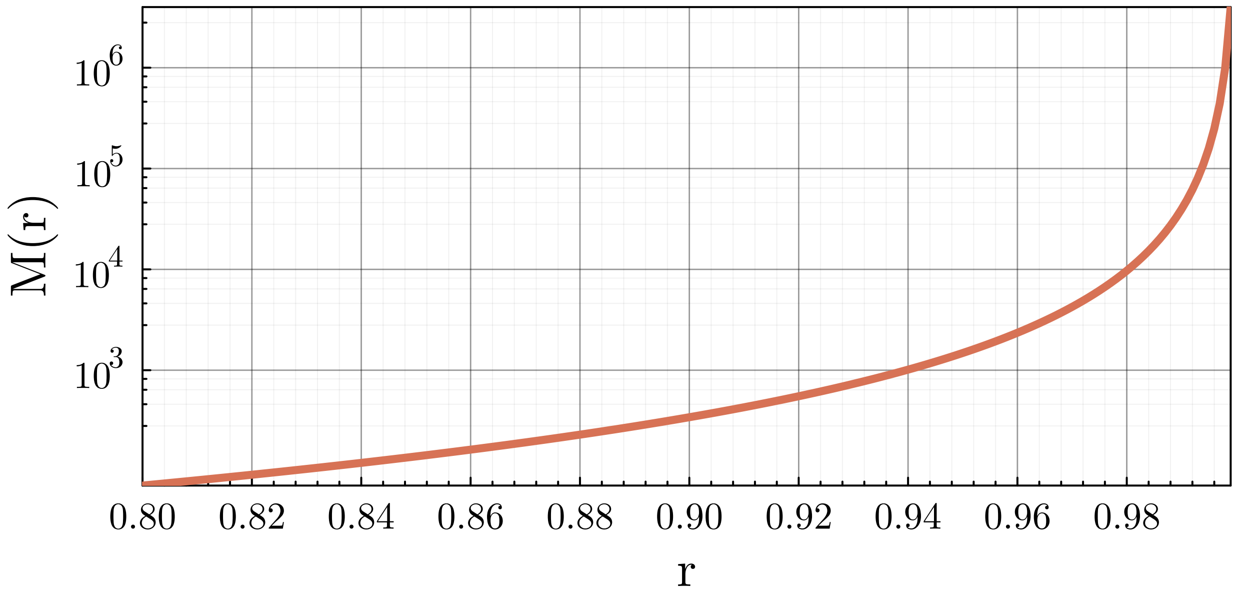
The method is enveloped in the `BBB.jl` file as part of the `AnalysisModule.jl`. The main object for this method is the `BBB` type, which takes in the following inputs:

1. `vector1::{Vector{<:Real}}`: First (reference) vector of values, i.e. measured $\varphi's$ from standard distribution
2. `vector2::{Vector{<:Real}}` Second (compared) vector of values, i.e. measured $\varphi's$ from the refined distribution
3. `xMin::Real` minimum value for the binning, i.e. `0°`
4. `xMax::Real` maximum value for the binning, i.e. `180°`
5. `stepSize::Real` bin step, i.e. `5°`
6. `nSigma::Real` number of sigmas

At construction `BBB` has additional fiels:

1. `minEvents::Real` minimum required number of events
2. `minROI::Tuple{<:Real,<:Real}` best region of interest

We first look at the behaviour of `M(r)`. Notice that for $r \rightarrow 1.0,\ M(r)$ grows rapidly.

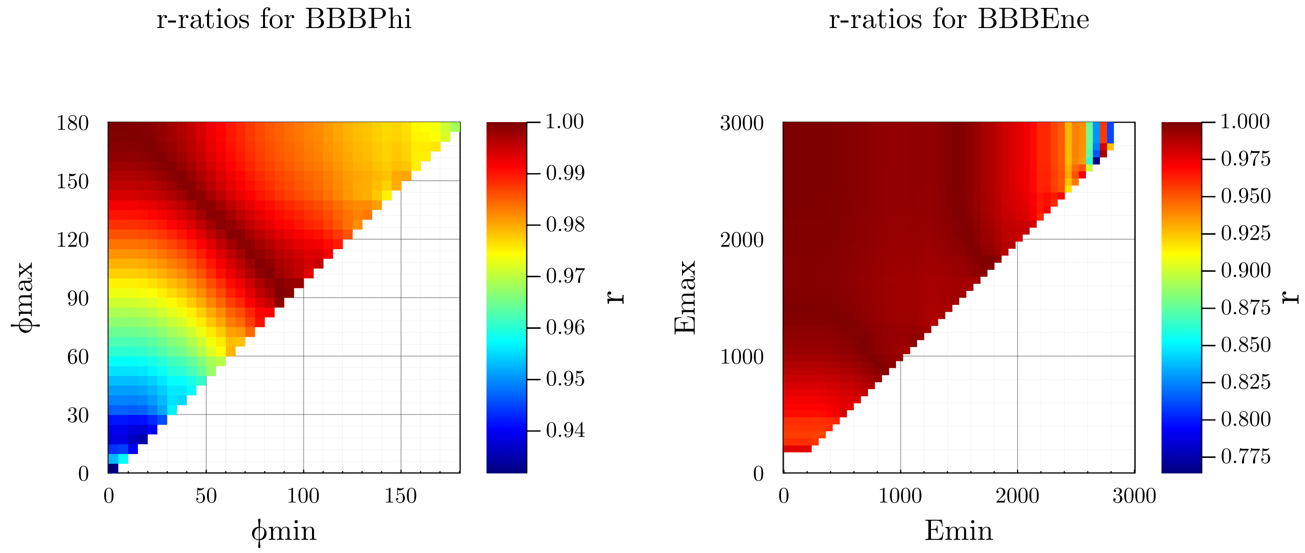


```
plot(0.8:0.001:0.999, AM.Mmin.(0.8:0.001:0.999, 1e5, 1e5),
    xlabel = "r",
    ylabel = "M(r)",
    label = "",
    lw = 4,
    size = (800,400),
    yscale= :log10,
    yticks= [10^3, 10^4, 10^5, 10^6, 10^7, 10^8],
    xticks= 0.8:0.02:1.0,
    thickness_scaling = 1.2
)
```

```
▶BBB([646.987, 865.134, 1072.84, 1199.18, 813.141, 550.047, 838.096, 1170.83, 1480.55, ⋯
```

```
begin
    BBBPhi = AM.BBB(phi1, phi2, 0, 180, Δϕ, 1)
    BBBEne = AM.BBB(singleElectronEnergies1, singleElectronEnergies2, 0, 3000, ΔE,
    1)
end
```

We can look at what the r-ratios look like for each distribution


r-ratios for BBBPhi

r-ratios for BBBEne

This map should be read in the following way: Each square represents a specific ROI defined by (min, max) value to which the square corresponds. (i.e. the very first square - left, bottom corner - of φ distribution corresponds to ROI: φ ∈ (0,5)°. The one above it then corresponds to ROI: φ ∈ (0,10)°, and so on.)

In the figure we can see the calculated r_i for each range of ϕ's and Energies. The most ideal value is the lowest - greatest difference between the two spectra.

This however, does not tell the whole story yet. We must consider the uncertainty of the result as well as the uncertainty of the simulation. Furthermore, since the aim was to answer **how many events are required** to distinguish the two spectra, we still have some steps to take.

We will therefore produce a few more calculations.

First, to calculate the number of events required, we begin with the following:

Assume M represents the number of events in the *smaller* bin i and N represents the number of events in the *larger* bin. Then in order to distinguish the two bins from each other at the level of n_σ, their difference must be at least equal (or greater) than the sum of their respective uncertainties.

$$M - N = n_\sigma(\Delta M + \Delta N); \Delta M = \sqrt{M}, \Delta N = \sqrt{N}$$

Dividing the equation by M, substituting $r = \frac{M}{N}$ and rearranging yields:

$\tilde{M}(r) = n_\sigma^2(\frac{r+\sqrt{r}}{1-r})^2$, where $\tilde{M}$ simply represents the minimum number of events in M required to distinguish M from N by n_σ.

Now, using uncertainty propagation, we can find the uncertainty on $\tilde{M}$ as:

$\Delta\tilde{M} = n_\sigma^2(\frac{r+\sqrt{r}}{1-r})^3 r\sqrt{1/M + 1/N}$.

The uncertainty in $\tilde{M}$ can be improved by obtaining higher statistics.

However, due to the fact that we do not precisely understand the detector angular correlations, we do not know precisely the analytical value for r either. (If we knew perfectly the correlation $\theta \to \phi$, we could obtain r analytically from the input angular distributions). We must, therefore, take into account the uncertainty on r.

$$\Delta r = r\sqrt{1/M + 1/N};$$

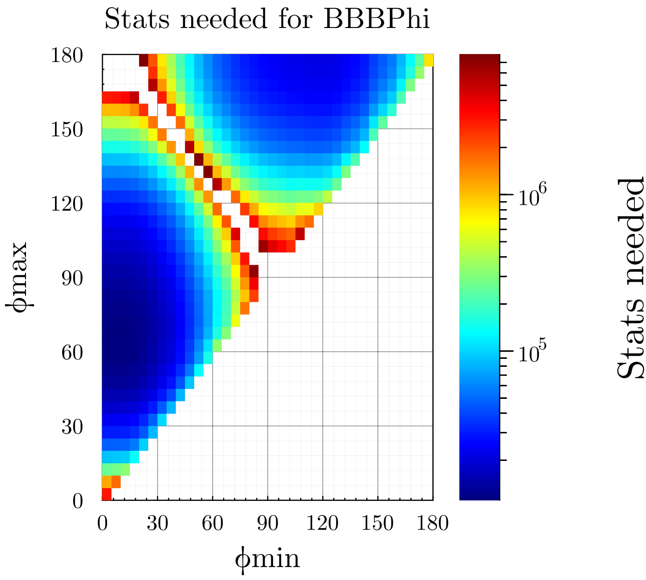
Since the behavior of r is very much non-linear ($\tilde{M}$ rapidly explodes for r close to 1.0) we consider the maximum (and minimum) uncertainties:

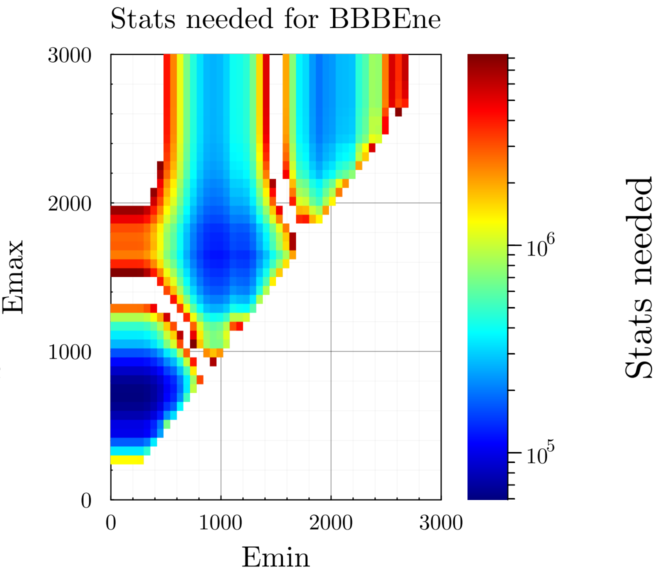
$r_{max} = \frac{M+\Delta M}{N-\Delta N}$ $r_{min} = \frac{M-\Delta M}{N-\Delta N}$.

Here, the worst case scenario (r_{max}) can in some cases exceed 1.0, (ie. $r > 1.0$). Such regions will be removed from the analysis and will show in the maps as empty squares.

To answer the question *how many events SN needs to measure to distinguish* we can convert the $\tilde{M}(r)$ into the number of events needed S as follows: $S = \tilde{M}(r)/\varepsilon$. That is, we scale the number of needed events in the bin by the proportion of the total events that bin represents. S then gives the total statistics needed to obtain desired $\tilde{M}(r)$.

Finally, let us look at all of the mentioned values. In the figure below we show maps for $S(r)$.





First of all, the *excluded* regions are the ones for which the *maximum-error* exceedes physical values (negative under square-root).

We can see in these maps, that the values for *stats-needed* can take a very wide range. The best values, for `nSigma = 1`, are found as:

1. φ distribution: 11099.1 for ROI: (0, 70) degrees
2. Energy distribution: 59348.03 for ROI: (0, 720) keV

The conclusions from this method should be taken with a grain of salt as it is very non-standard. We therefore use more common approaches: χ^2 and KS.

The following text will deal with χ^2 and `KS` methods together

The goal is to find the ideal value of number of events needed to distinguish (`S`). To do so, we construct $H_0 \equiv \textit{tested distributions have the same underlying distribution}$. That is, within null hypothesis we expect the two compared spectra to be the same. Thus our condition is to reject this hypothesis (we want $p - value < \alpha$).

The methodology is as follows:

1. The spectra are split into `100` random subsets of various sizes `M`
2. For each subset a KS and ChiSquare hypothesis test is performed and `p-value` is extracted. We obtain 100 p-values for each `M`.
3. We define efficiency $\varepsilon \equiv N_{rejected\ H_0} / N_{total=100}$
4. For various values of CL (i.e. 90%, 95%) the corresponding `S` is found as the minimum `M` for which the efficiency of rejecting H_0 is **100% for three consequitive times** for increasing `M's`.

We define `sampleSizes` (M) to be sizes from 10000 events to 19000 events (in step 1000) and from 20000 to 150000 (in steps of 10000) and up to 400000 in steps of 50000.

The methods are, again, contained within their own submodules: `Chi2.jl` and `KS.jl`. The input arguments for both are the same:

1. `vector1::Vector{T}`
2. `vector2::Vector{T}`
3. `xMin::Real` # minimum value of the array (i.e. for angles it would be 0 degrees)
4. `xMax::Real` # maximum value of the array (i.e. for angles it would be 180 degrees)
5. `stepSize::Real`
6. `sampleSizes::Vector{<:Real}`
7. `CL::Real`

The constructor then makes three more fieds:

1. `pVals::Vector{Vector{<:Real}}` - container for p-values, for each subset of size M we get 100 p-values
2. `efficiencies::Vector{<:Real}` - vector of efficiencies for each subset size
3. `minEvents::Real` - minimum events required

```
0.95
```

```julia
begin
    sampleSizes = vcat(collect(20_000:10_000:100_000), collect(150_000:50_000:800_000) )
    xticks=(1:length(sampleSizes), sampleSizes)
    CL = 0.95
end
```

Disclaimer: For χ^2 there is an extra condition that must be fillfilled taking into account the methodology. The binning must be chosen so that at least 5 events are in each bin. If this is not fulfilled, it will throw error.

```
▶Chi2([47.2344, 125.405, 134.476, 144.129, 145.962, 117.361, 145.096, 124.481, 139.037, ⋯
```

```julia
begin
    Chi2Ene = AM.Chi2(
        singleElectronEnergies1, singleElectronEnergies2, 300, 2400, 150, sampleSizes,  CL)
    Chi2Phi = AM.Chi2(phi1, phi2, 0, 180, 15, sampleSizes,  CL)
end
```

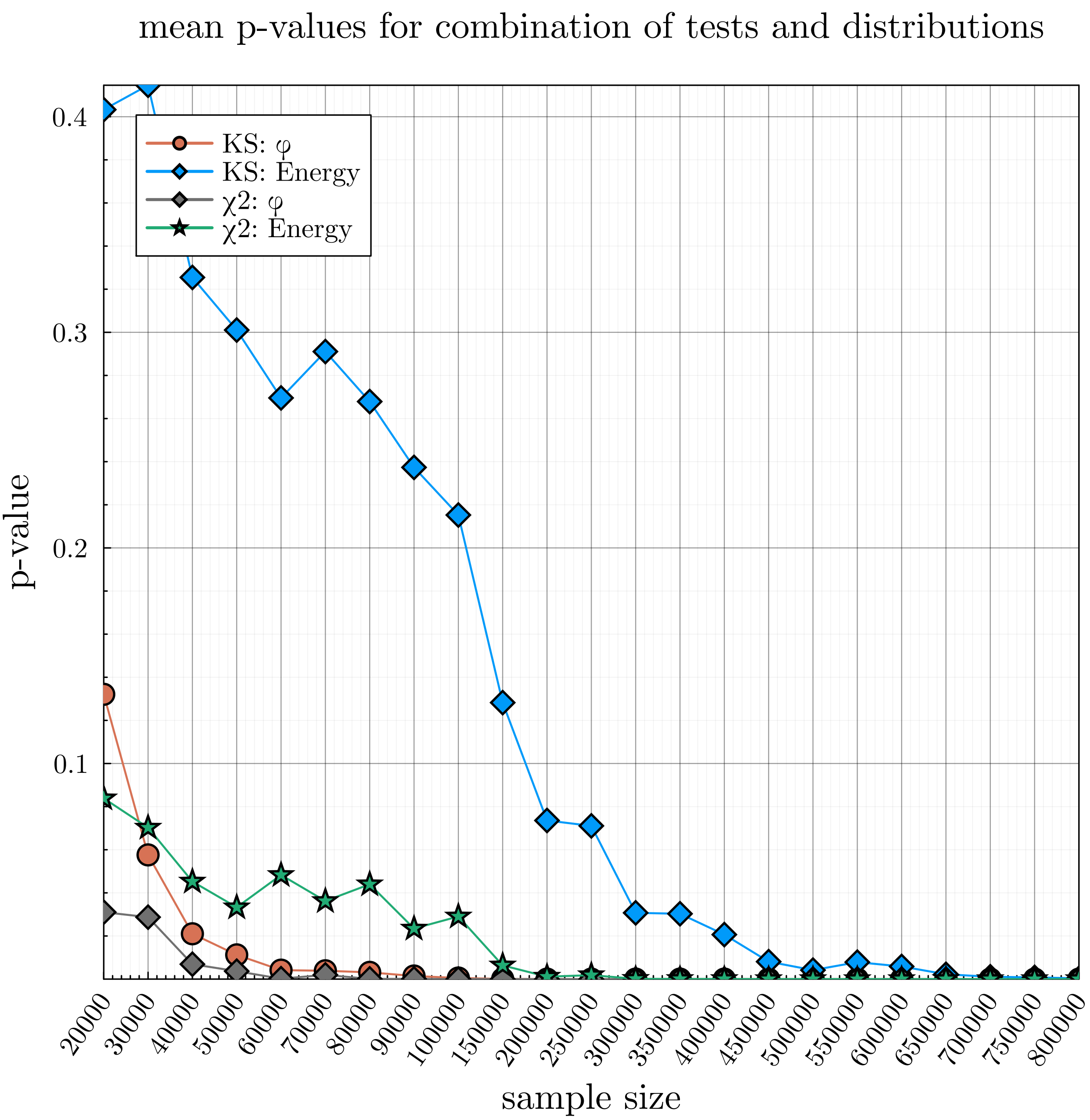
```
▶ KS([47.2344, 125.405, 134.476, 144.129, 145.962, 117.361, 145.096, 124.481, 139.037, ⋯ r
```

```julia
begin
    KSEne = @suppress AM.KS(
        singleElectronEnergies1, singleElectronEnergies2, 0, 3000, ΔE, sampleSizes,
         CL)
    KSPhi = @suppress AM.KS(phi1, phi2, 0, 180, 5, sampleSizes, CL)
end
```

```
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
⚠ This test is inaccurate with ties
```

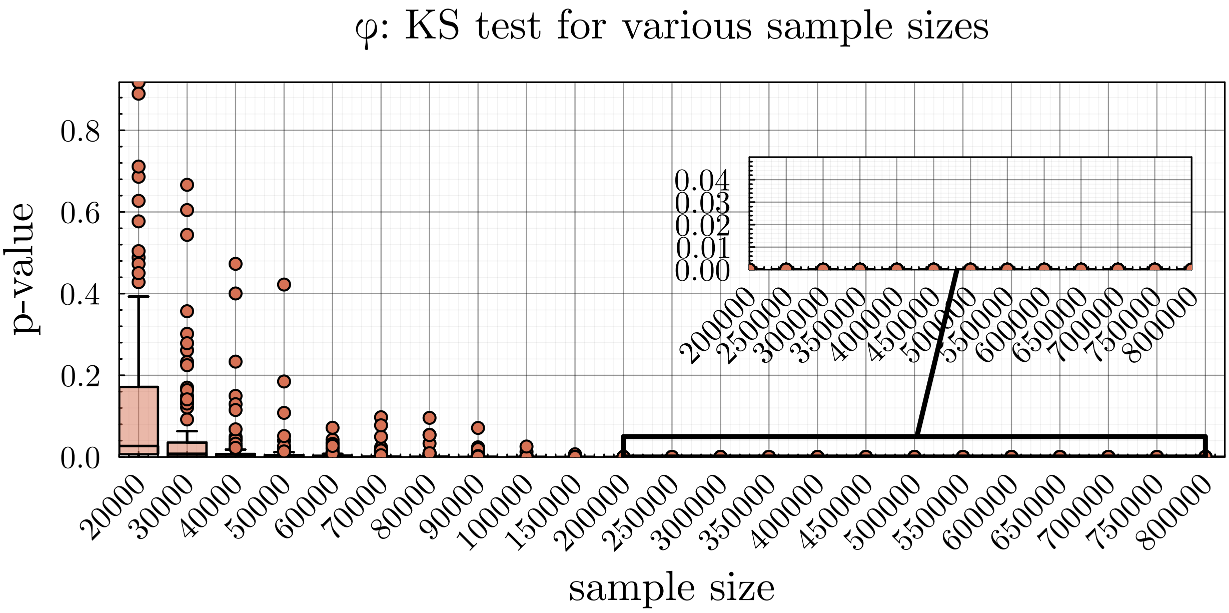
```
▶ [0.0838528, 0.070258, 0.0452556, 0.0332936, 0.0482962, 0.0362167, 0.0439735, 0.0234358,
```

```julia
begin
    meansKSPhi = mean.(KSPhi.pVals)
    meansKSEne = mean.(KSEne.pVals)
    meansChi2Phi = mean.(Chi2Phi.pVals)
    meansChi2Ene = mean.(Chi2Ene.pVals)
end
```



Above is a plot of mean p-values for each combination. We can see that the `energy` distribution needs more statistics to reject H_0, in fact we have not even reach enough to obtain desired statistics. We look a bit closer at the data by looking at a boxplot, which shows the median and 2nd, 3rd quantile and outliers.

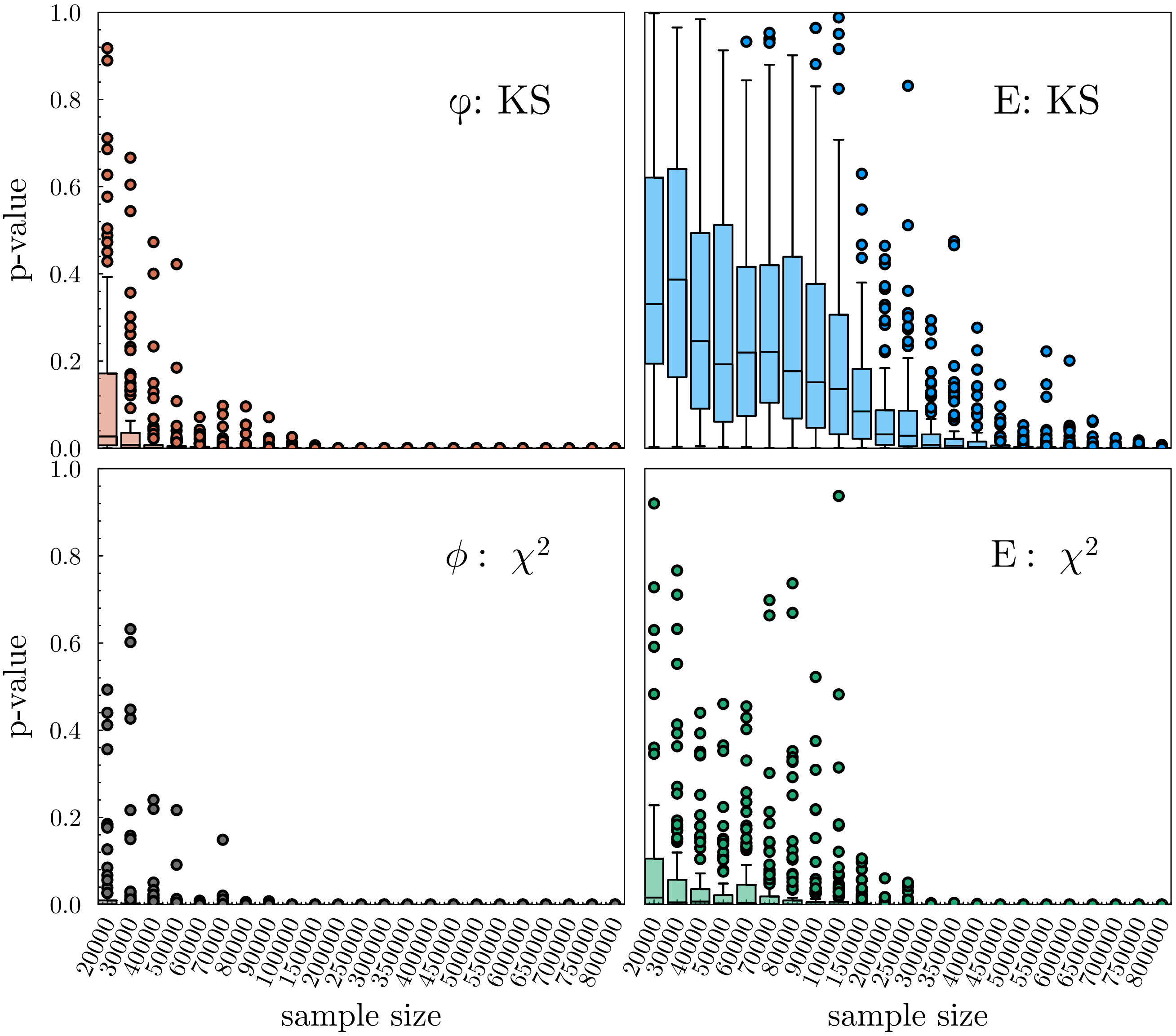
First, we can look at a boxplot of the obtained p-values obtained from KS test for φ for each sample size. We can see that for smaller sample sizes the p-values vary greatly. Whereas for larger sample sizes, i.e. $N > 100k$ events it is very close to zero.



We can also look at the other tests:

1. φ distribution using KS test
2. Single electron spectra (E) using KS test
3. φ distribution using Chi2 test
4. E distribution using Chi2 test

Right away we can see that KS test is more strict when it comes to rejecting H0. Furthermore, it seems that the angular distribution is the more promising candidate for distinguishing the spectra.



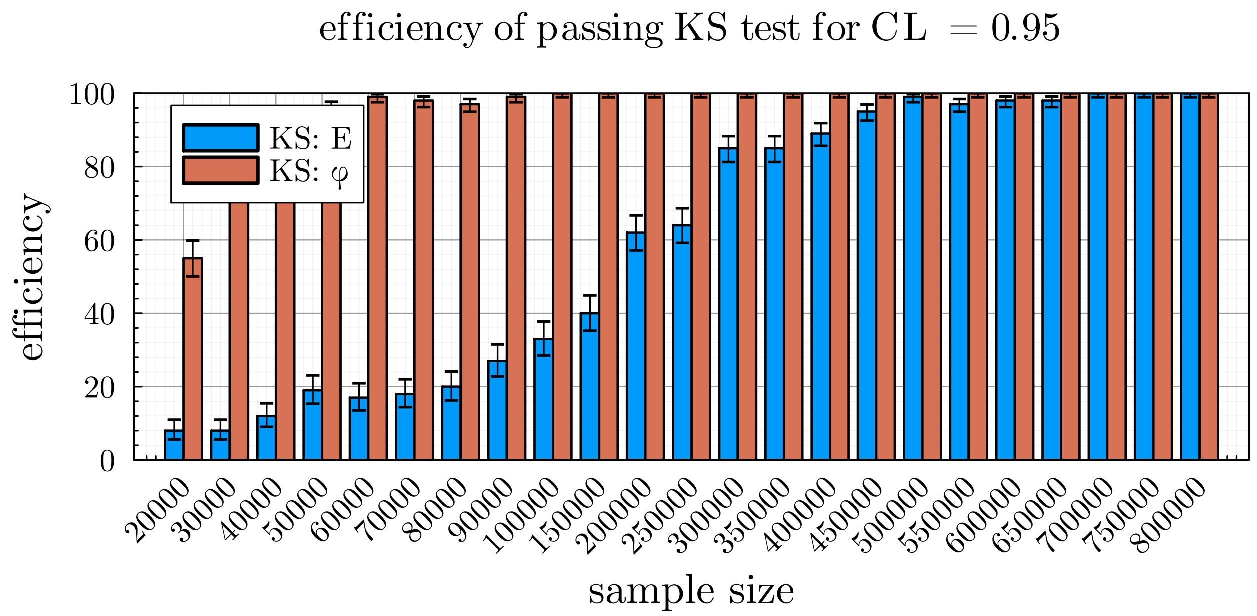
```julia
begin
    bp1 = boxplot(
        KSPhi.pVals,
        label="",
        xticks=:none,
        xrotation=65,
        c=1,
        fa=0.5,
        ylabel="p-value",
        top_margin=0Plots.px,
        bottom_margin=0Plots.px,
        left_margin=0Plots.px,
        right_margin=0Plots.px,
        ylims=(0,1)
    )
    annotate!([(18,0.8,("φ: KS", 20))])

    bp2 = boxplot(
        KSEne.pVals,
        label="",
        xticks=:none, #(1:length(sampleSizes), sampleSizes),
        xrotation=65,
        c=2,
        fa=0.5,
        yticks=:none,
        top_margin=0Plots.px,
        bottom_margin=0Plots.px,
        left_margin=0Plots.px,
        right_margin=0Plots.px,
        ylims=(0,1)
    )
    annotate!([(18,0.8,("E: KS", 20))])

    bp3 = boxplot(
        filter!.(x -> x .>=0, Chi2Phi.pVals),
        label="",
        xticks=(1:length(sampleSizes), sampleSizes),
        xrotation=65,
        c=3,
        fa=0.5,
        xlabel="sample size",
        ylabel ="p-value",
        top_margin=0Plots.px,
        left_margin=0Plots.px,
        right_margin=0Plots.px,
        ylims=(0,1)
    )
    annotate!([(18,0.8,(L"\mathrm{φ:~ \chi^2}", 20))])


    bp4 = boxplot(
        filter!.(x -> x .>=0, Chi2Ene.pVals),
        label="",
        xticks=(1:length(sampleSizes), sampleSizes),
        xrotation=65,
        c=4,
        fa=0.5,
        xlabel="sample size",
        top_margin=0Plots.px,
        left_margin=0Plots.px,
        right_margin=0Plots.px,
        yticks=:false,
        ylims=(0,1)
    )
    annotate!([(18,0.8,(L"\mathrm{E:~ \chi^2}", 20))])


    bxpAll = plot(
        bp1, bp2, bp3, bp4,
        size=(1200, 1100),
        thickness_scaling=1.3,
        grid=:false,
        minorgrid=:false
    )

```

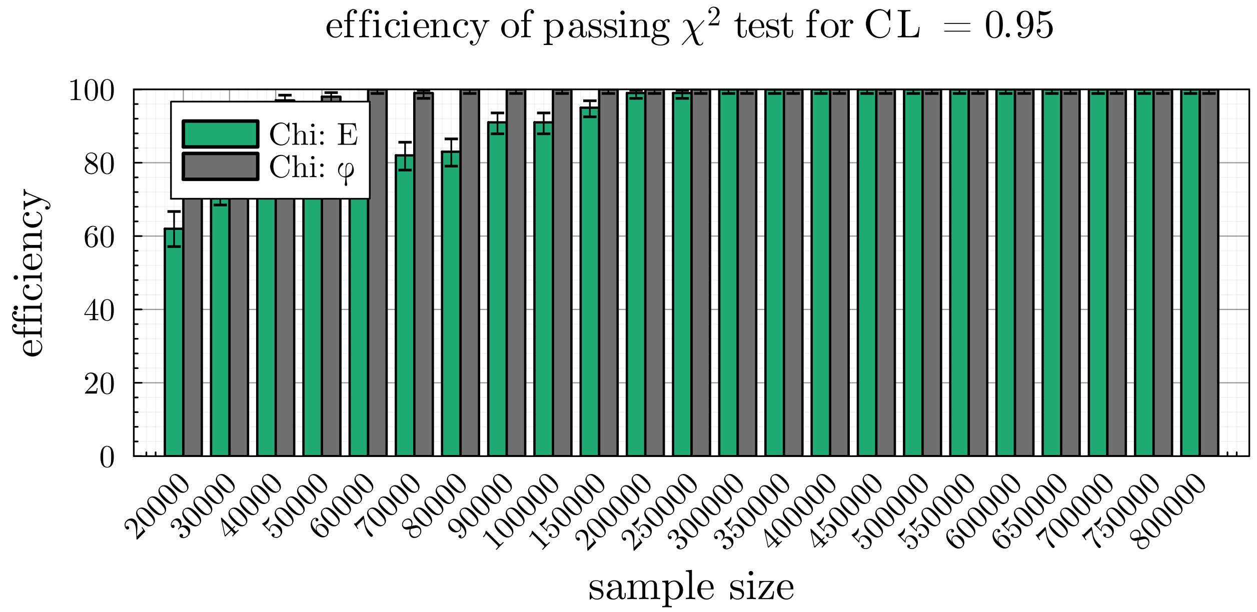
However, since it can be seen that the p-values vary greatly, we can perform an additional analysis by looking at the efficiency of rejecting H0 for various sample sizes. That is, for each subset of size N we compute efficiency as $\varepsilon = \frac{N_{reject}}{N}$.

```
▸ [62, 73, 81, 85, 76, 82, 83, 91, 91, 95, 99, 99, 100, 100, 100, 100, 100, 100, 100, 100, 1
```

```
begin
    alpha = 1-0.95

    effKSPhi95 =
        count.(p -> p<=alpha, KSPhi.pVals)./length.(KSPhi.pVals).*100 .|> round
        .|>Int
    effKSEne95 =
        count.(p -> p<=alpha, KSEne.pVals)./length.(KSEne.pVals).*100 .|> round
        .|>Int

    effChiPhi95 =
        count.(p -> p<=alpha, Chi2Phi.pVals)./length.(Chi2Phi.pVals).*100 .|> round
        .|>Int
    effChiEne95 =
        count.(p -> p<=alpha, Chi2Ene.pVals)./length.(Chi2Ene.pVals).*100 .|> round
        .|>Int

end
```

```
begin
    passedKSPhi = effKSPhi95
    passedKSEne = effKSEne95

    passedChiPhi = effChiPhi95
    passedChiEne = effChiEne95

    errKSPhi = AM.prop_error.(passedKSPhi, 100, true)
    errKSEne = AM.prop_error.(passedKSEne, 100, true)

    errChiPhi = AM.prop_error.(passedChiPhi, 100, true)
    errChiEne = AM.prop_error.(passedChiEne, 100, true)
    nothing
end
```





Finally we can create a table of *S* for the various defined methods and sample sizes.

```
for (cl, nsig) in zip([0.68, 0.90, 0.95], [1, 1.645, 1.96])
    minKSPhi = AM.get_best_sample_size(KSPhi, sampleSizes, cl, 100,3)
    minKSEne = AM.get_best_sample_size(KSEne, sampleSizes, cl, 100,3)
    minChi2Phi = AM.get_best_sample_size(Chi2Phi, sampleSizes, cl, 100,3)
    minChi2Ene = AM.get_best_sample_size(Chi2Ene, sampleSizes, cl, 100,3)
    minBBBPhi = AM.BBB(phi1, phi2, 0, 180, Δφ, nsig).minEvents
    minBBBEne = AM.BBB(singleElectronEnergies1, singleElectronEnergies2, 0, 3000, ΔE, nsig).minEvents
end
```

CL\method	KSphi	KSEne	X^2Phi	X^2Ene	BBBPhi	BBB Ene
68%	70k	450k	60k	350k	11k	60k
90%	150k	750k	60k	350k	30k	160k
95%	200k	750k	200k	500k	45k	230k

```
md"""
CL\method | KSphi |KSEne | X²Phi | X²Ene | BBBPhi | BBB Ene
----------|-------|------|-------|-------|--------|--------
68%       |70k    |450k  |60k    |350k   |11k     |60k
90%       |150k   |750k  |60k    |350k   |30k     |160k
95%       |200k   |750k  |200k   |500k   |45k     |230k

"""
```