In [1]:
```
using Revise
using StatsPlots, UnROOT, StatsBase, CategoricalArrays, Polynomials, LinearAlgebra
using FHist, LaTeXStrings, MPThemes, DataFrames, DataFramesMeta, Distributions
```

In [2]:
```
ENV["COLUMNS"] = 2000
ENV["LINES"] = 20
```

Out[2]:
```
20
```

Custom functions are placed in the MiscFuncs.jl file.

In [3]:
```
include("MiscFuncs.jl")
using .MiscFuncs
```

In [4]:
```
gr()
my_vibrant(;
    size           = (800, 600),
    legend         = :outertopleft,
    guidefontsize  = 12,
    tickfontsize   = 12,
    titlefontsize  = 12,
    legendfontsize = 10,
    left_margin    = 4Plots.mm,
    right_margin   = 4Plots.mm,
    top_margin     = 4Plots.mm,
    bottom_margin  = 4Plots.mm,
    dpi            = 200,
);
```

In [5]:
```
baseDir = "/home/shoram/Work/PhD_Thesis/Job15/AngularCorrelations/"
```

Out[5]:
```
"/home/shoram/Work/PhD_Thesis/Job15/AngularCorrelations/"
```

In [6]:
```
f = ROOTFile(
    baseDir*"AngularCorrelationAllEnergies96MilEvents.root",
```

```
);
tree = DataFrame(LazyTree(f, "tree", keys(f["tree"])));
f= nothing; # free file from memory with gc
```

`@transform` adds a column `:ESum` to the `tree` which contains the sum of the electron energies

In [7]:
```
@transform! tree :ESum = :reconstructedEnergy2 + :reconstructedEnergy1;
```

Initializing constants.

In [8]:
```
dEmitted = 1 # dθdif in degrees
nBins    = Int(180 / dEmitted)
minAngle = 0
maxAngle = 180
binWidth = maxAngle / nBins

minEnergy = 500
maxEnergy = 3500
dEnergy   = 500

xPts = minAngle:dEmitted:maxAngle-dEmitted

dϕ = dEmitted               # step in ϕ, if same as bin width
sign = "p"                  # sign in get_cut_edges function
maxSteps = Int(180 / dϕ)    # max number of steps (slices)
```

Out[8]:
```
180
```

In [9]:
```
colors = [palette(:seaborn_bright)[i] for i in 1:length(palette(:seaborn_bright))];
```

The 2d Histogram of `ϕ` vs `θ` is defined to be `f(ϕ, θ)`. For each combination of `ϕ` and `θ`, the bin number is obtained as functional value of `f(ϕ, θ)`.

In [10]:
```
rho(_cosdTheta) = 0.5 - 0.5 * _cosdTheta
```

```
# @transform! tree :weights = Weights(rho.(cosd.(:thetaEmitted))) # weights by pdf
@transform! tree :weights = 1                                    # unweighted
```

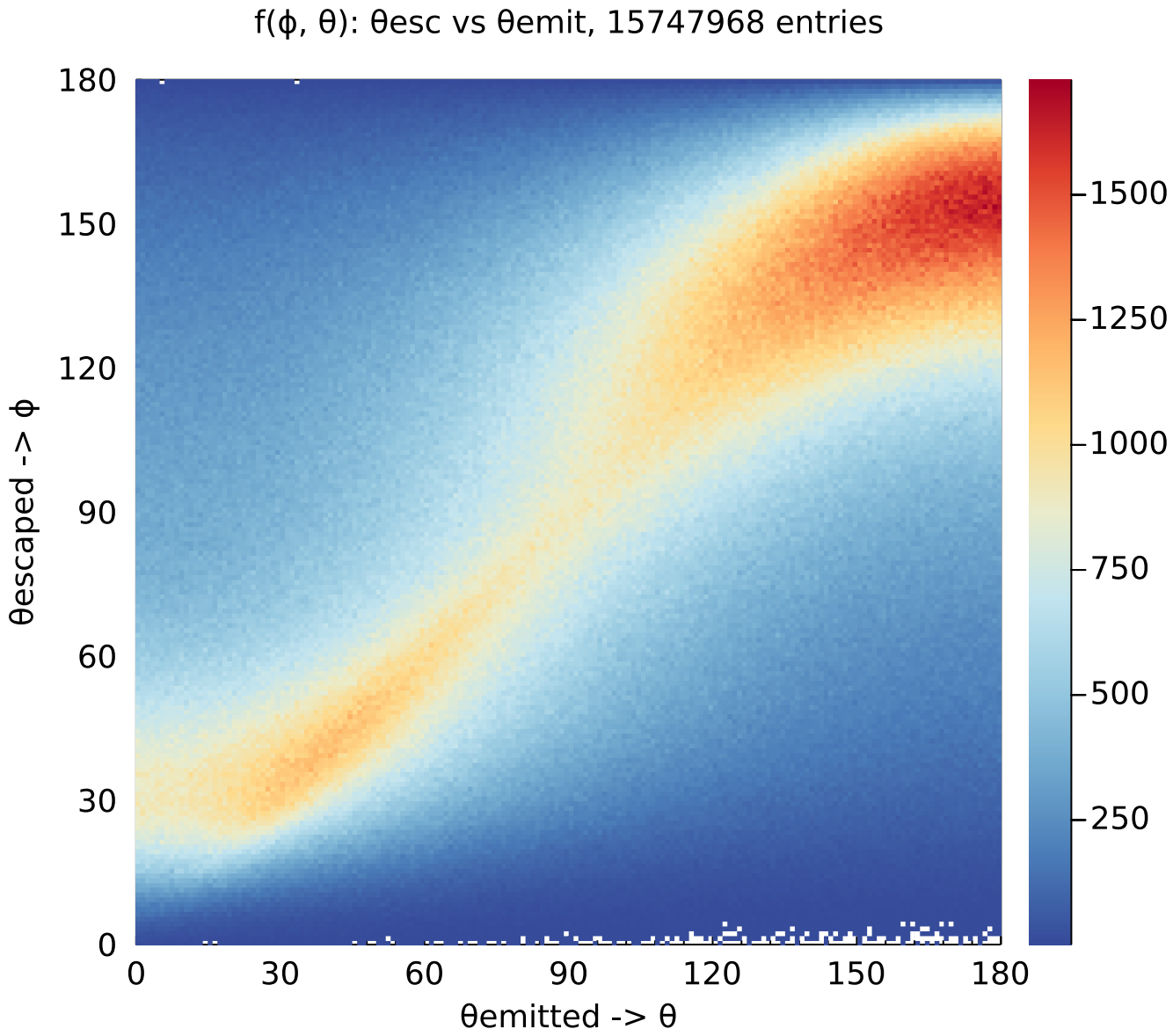
Out[10]: 15,747,968 rows × 18 columns

| | momentumEmitted2x | reconstructedEnergy2 | momentumEscaped1z | momentumEmitted1x | thetaEmitted | momentumEscaped1y | momentumEmitted1z |
|---|---|---|---|---|---|---|---|
| | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 |
| 1 | 0.723365 | 604.269 | -0.471071 | -1.41249 | 89.6031 | 2.26532 | -0.946003 |
| 2 | -1.87869 | 1651.11 | 0.111843 | 0.620354 | 96.965 | -0.427881 | 0.585939 |
| 3 | -0.451873 | 450.772 | -0.204413 | 0.143294 | 154.233 | -0.344668 | 0.461237 |
| 4 | 0.00622859 | 2520.27 | 0.0712075 | -0.27381 | 67.3403 | 0.440423 | -0.0579716 |
| 5 | -0.660664 | 834.025 | -0.282475 | -1.77769 | 11.2133 | -2.1054 | -0.0196651 |
| 6 | -0.44519 | 1530.67 | 0.917931 | 0.0695716 | 168.417 | 1.51856 | 2.09515 |
| 7 | -0.445143 | 1784.88 | 0.277989 | 0.889556 | 122.632 | -0.0529092 | 0.257764 |
| 8 | -1.42443 | 1723.99 | 0.250602 | -1.16971 | 4.96695 | 0.834453 | 0.0445409 |
| 9 | -0.0553668 | 1281.87 | 0.457432 | -0.344509 | 79.857 | 1.30726 | -0.133961 |
| 10 | -1.7249 | 2124.13 | -0.525607 | -0.976017 | 43.0942 | 0.765786 | -0.71208 |
| 11 | -0.19787 | 467.234 | -0.513265 | -0.80288 | 35.1387 | -0.210073 | -0.13689 |
| 12 | -0.529524 | 2307.59 | -0.682419 | -0.307922 | 150.65 | -0.818542 | -0.238325 |
| 13 | 1.22009 | 976.306 | -0.64402 | 0.895972 | 56.3877 | 0.122387 | -0.727497 |
| 14 | 1.62335 | 1841.63 | -0.772472 | 0.148038 | 128.995 | -1.01428 | -0.825084 |
| 15 | -0.975718 | 619.331 | -1.0753 | -0.587855 | 63.2692 | 1.86365 | -1.97465 |
| 16 | -0.181153 | 985.07 | 1.34426 | -1.8497 | 40.4733 | -0.640471 | 1.69434 |
| 17 | 1.62458 | 2530.76 | 0.0707918 | -1.20434 | 124.468 | 0.512761 | 0.35071 |
| 18 | 0.685257 | 639.204 | 0.0196268 | -0.00221343 | 128.263 | 0.374426 | 0.395698 |
| 19 | -0.211174 | 628.139 | 0.502857 | -2.74734 | 105.706 | 2.36161 | -0.622121 |
| 20 | 0.0476415 | 707.158 | -0.736812 | -1.20521 | 142.67 | 1.44929 | -0.496897 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [11]:

```
fh2d = Hist2D(                                          # h2d object similar to TH2D from ROOT
(tree.thetaEmitted, tree.thetaEscaped),
    Weights(tree.weights),
(minAngle:dEmitted:maxAngle, minAngle:dEmitted:maxAngle),
)

h2d1 = histogram2d(
    tree.thetaEmitted,
    tree.thetaEscaped;
    weights      = tree.weights,
    nbins        = (nBins, nBins),
    xlabel       = "θemitted -> θ",
    ylabel       = "θescaped -> ϕ",
    legend       = :topright,
    title        = string("f(ϕ, θ): θesc vs θemit, ", nrow(tree), " entries"),
    lims         = (0, 180),
    aspect_ratio = 1,
)
```

Out[11]:




In [12]:
```
savefig(h2d1, joinpath(baseDir, string("h2d.png")))
```

In an ideal situation, all the 2D histogram would have bins full only along the diagonal where $\phi = \theta$.

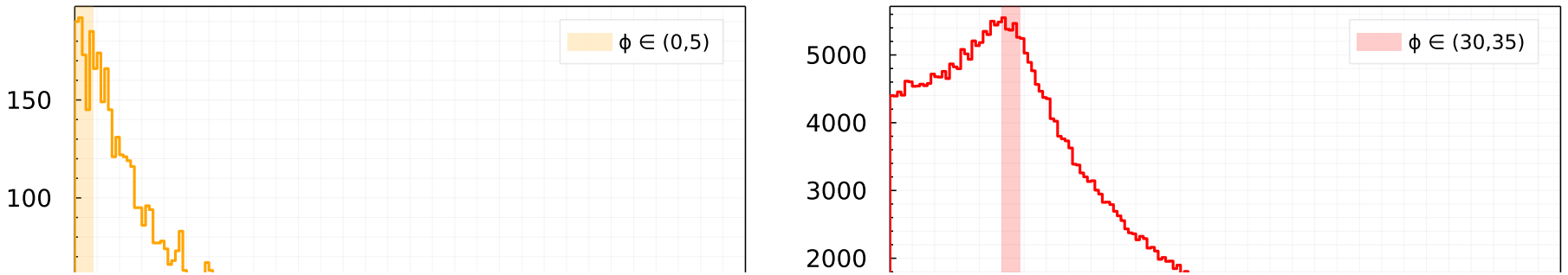
This of course is not the case here. To investigate closer, let's take a look at a few horizontal slices of $\phi$ defined by some $d\phi$ segment. That is we create a 1D histogram of all $\theta$ which fall within the slice $\phi \in (\phi_i, \phi_i + d\phi)$. For example we can take a look at:

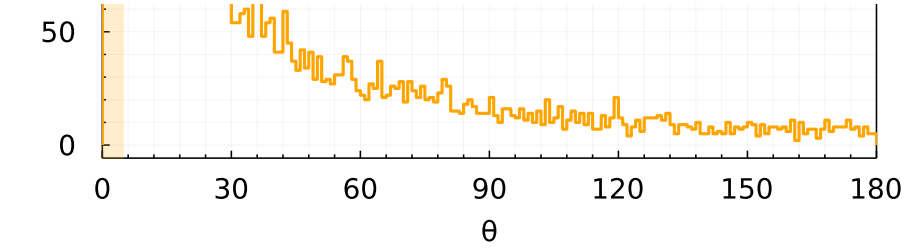
1. $\phi \in (0, 5)$
2. $\phi \in (30, 35)$
3. $\phi \in (85, 90)$
4. $\phi \in (145, 150)$
5. $\phi \in (175, 180)$

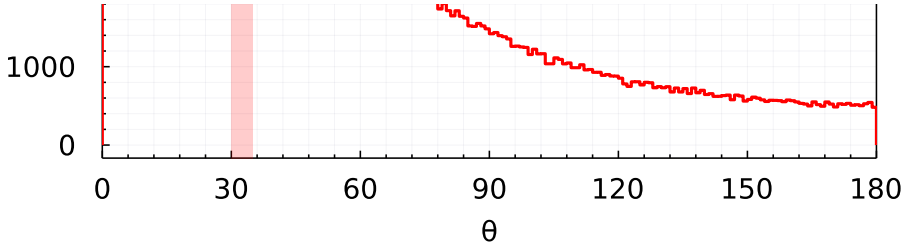
In [13]:

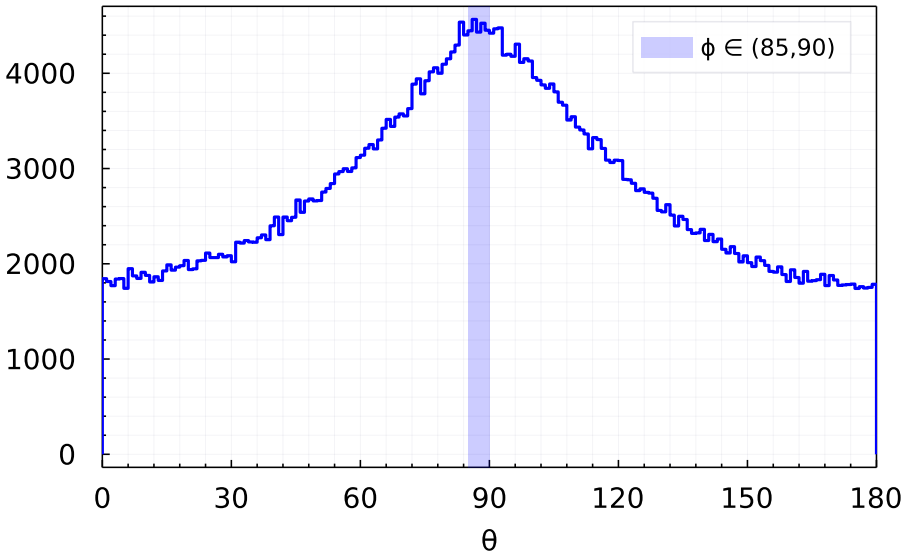
```
with(:gr, size=(1200,1200), legend = :topright, xlims = (0,180), xlabel ="θ", lw = 2, nbins = 180 ) do
    h1 = stephist( tree[ tree.thetaEscaped .< 5.0, :thetaEmitted ],
                weights=tree[tree.thetaEscaped .< 5.0, :weights], label ="", c =:orange)
    vspan!([0,5], alpha = 0.2, c =:orange, label = "ϕ ∈ (0,5)" )
    h2 = stephist( tree[ 30 .< tree.thetaEscaped .< 35, :thetaEmitted ],
                weights=tree[30 .< tree.thetaEscaped .< 35, :weights],label ="", c =:red)
    vspan!([30,35], alpha = 0.2, c =:red, label = "ϕ ∈ (30,35)" )
    h3 = stephist( tree[ 85 .< tree.thetaEscaped .< 90, :thetaEmitted ],
                weights=tree[85 .< tree.thetaEscaped .< 90, :weights], label ="", c =:blue)
    vspan!([85,90], alpha = 0.2, c =:blue, label = "ϕ ∈ (85,90)" )
    h4 = stephist( tree[ 145 .< tree.thetaEscaped .< 150, :thetaEmitted ],
                weights=tree[145 .< tree.thetaEscaped .< 150, :weights],label ="", c =:green, legend = :topleft)
    vspan!([145,150], alpha = 0.2, c =:green, label = "ϕ ∈ (145,150)" )
    h5 = stephist( tree[ 175 .< tree.thetaEscaped .< 180, :thetaEmitted ],
                weights=tree[175 .< tree.thetaEscaped .< 180, :weights],label ="", c =:black, legend = :topleft)
    vspan!([175,180], alpha = 0.2, c =:black, label = "ϕ ∈ (175,180)" )
    plot(h1, h2, h3, h4, h5, layout = @layout [a b ; c d ; e _])
end
```

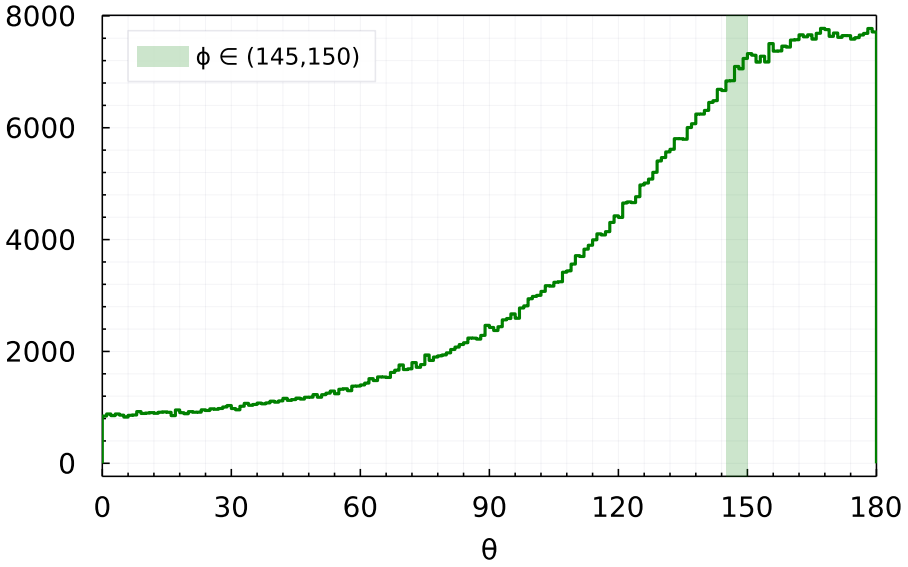
Out[13]:

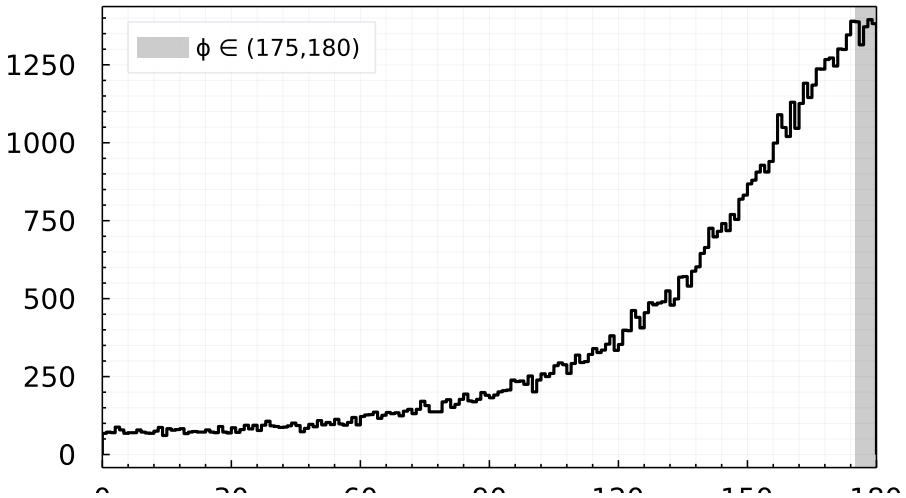


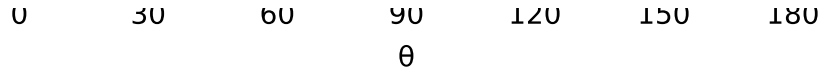

50
0
0
30
60
90
120
150
180
θ


1000
0
0
30
60
90
120
150
180
θ


ϕ ∈ (85,90)
4000
3000
2000
1000
0
0
30
60
90
120
150
180
θ


ϕ ∈ (145,150)
8000
6000
4000
2000
0
0
30
60
90
120
150
180
θ

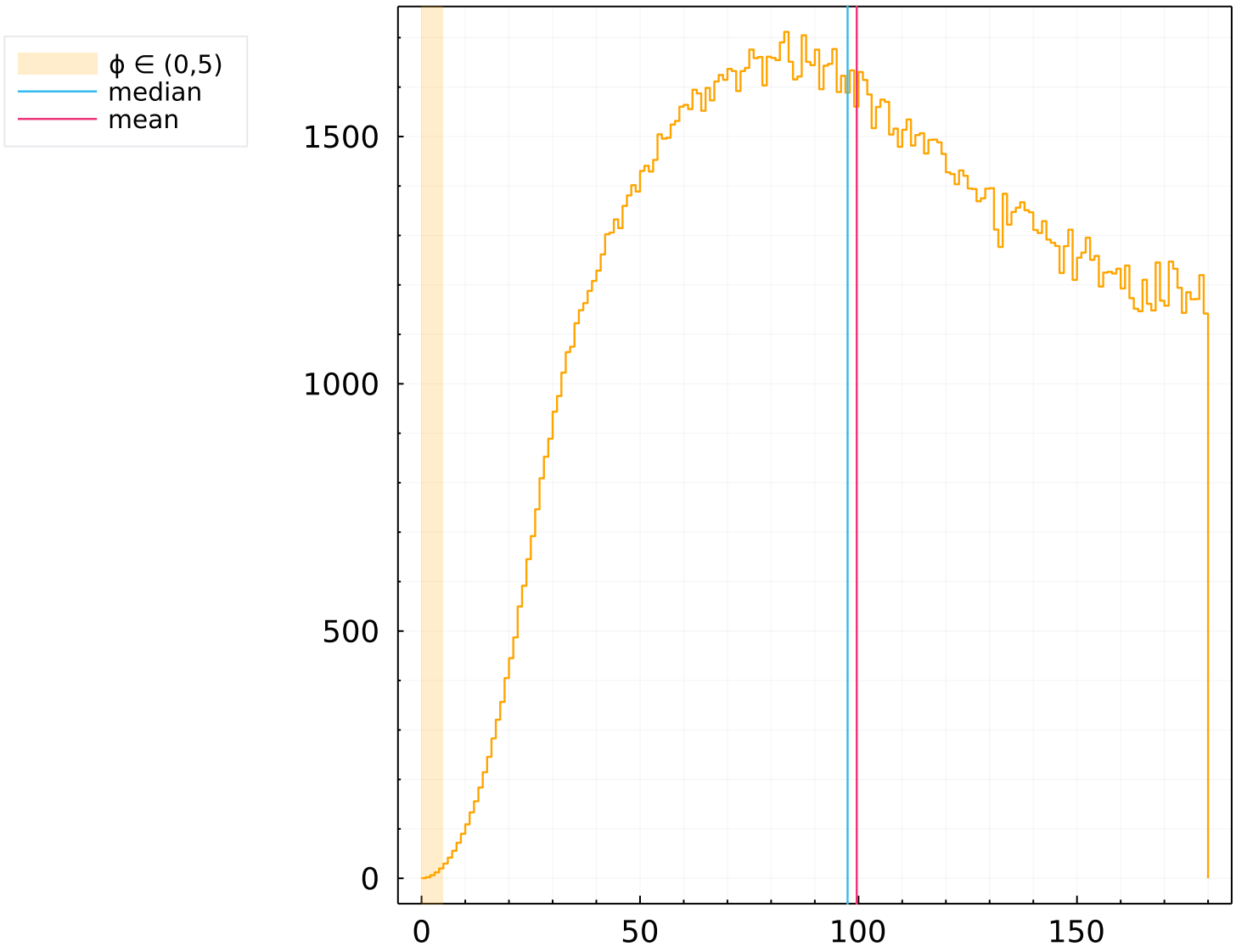

ϕ ∈ (175,180)
1250
1000
750
500
250
0




In [14]:

```
thetas = tree[20 .< tree.thetaEscaped .< 35.0, :thetaEmitted ]
w = Weights(rho.(cosd.(thetas)))

stats =  get_slice_stats(
                            20.0,  # ϕmin
                            35.0,  # ϕmax
                            0,             # Emin
                            3500,          # Emax
                            thetas,
                            dϕ,
                            w
                            )

h = Hist1D(thetas, w, (0:dϕ:180))
mediansFHist = StatsBase.median(h)

h1 = stephist( thetas, nbins = 180, weights= w, label ="", c =:orange)
vspan!([0,5], alpha = 0.2, c =:orange, label = "ϕ ∈ (0,5)" )
vline!([stats[7]], label ="median")
# vline!([mediansFHist], label ="medianFHist")
vline!([stats[5]], label ="mean")
```

Out[14]:




We define a line: `ϕ - θ - k = 0` with factor `k` which representing a diagonal line of `f(ϕ, θ)`. Where:

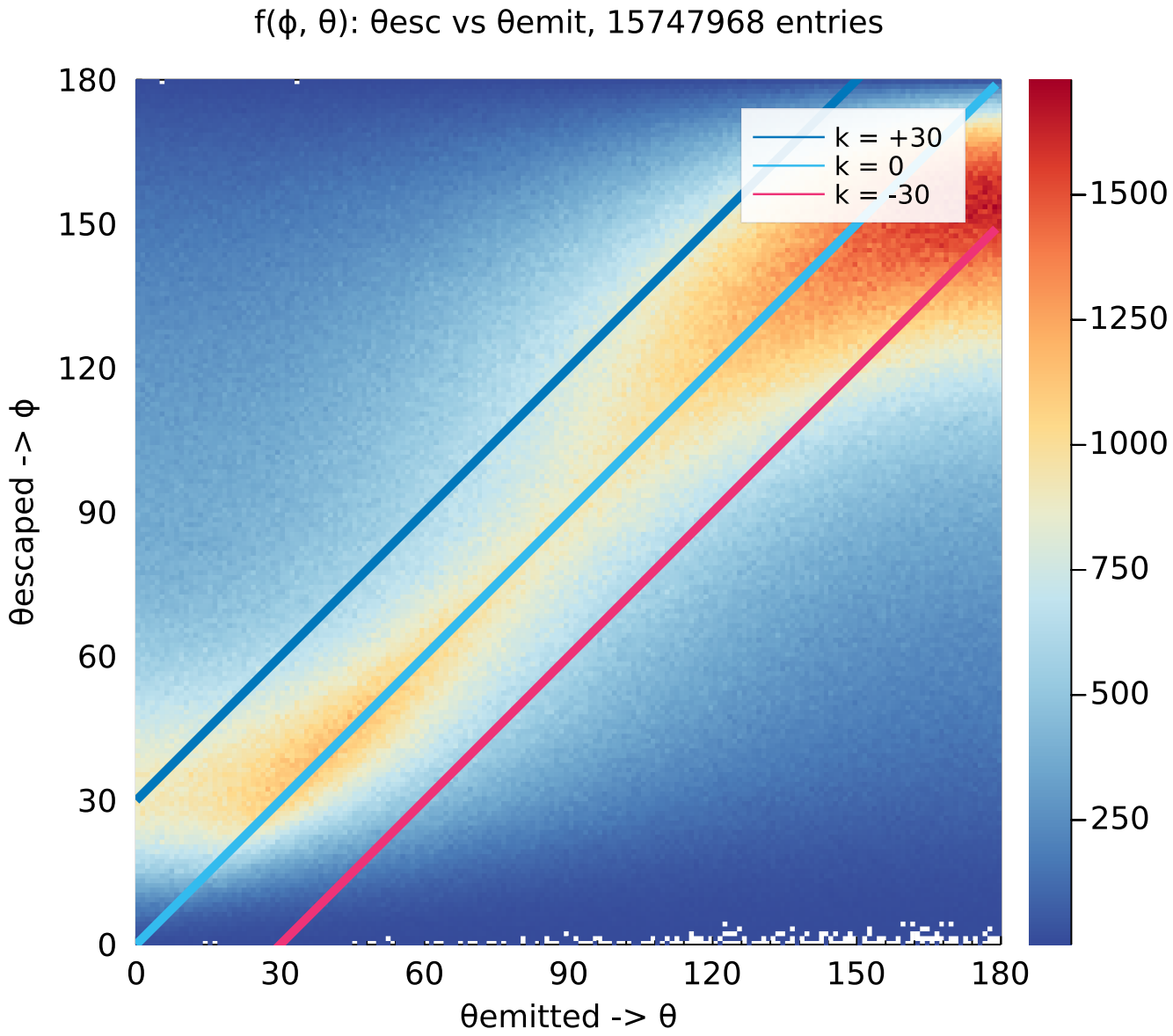
1. `k = 0` represents the line `ϕ = θ`. In other words, the escaped angle is same as emitted.

2. `k < 0` represents the lines where `ϕ < θ`. Escaped underestimates emitted.
3. `k > 0` represents the lines where `ϕ > θ`. Escaped underestimates emitted.

In [15]:

```
plot!(h2d1, xPts, xPts .+ 30, label = "k = +30", lw = 5)
plot!(h2d1,xPts, xPts, label = "k = 0", lw = 5)
plot!(h2d1,xPts, xPts .- 30, label = "k = -30", lw = 5)
```

Out[15]:



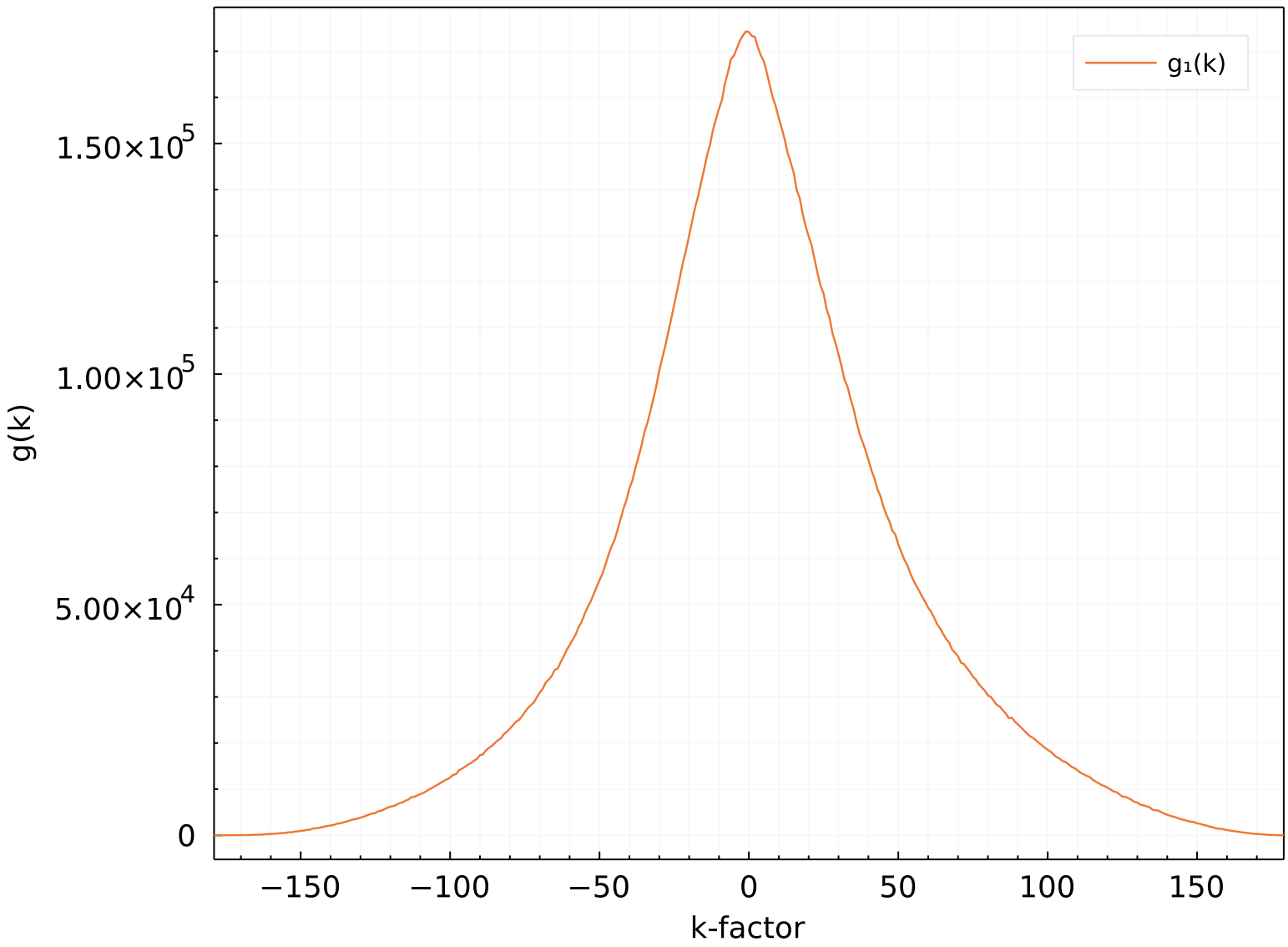

Next we define function `g(k)` representing the integral over the diagonal bins of the lines `k`

In [16]:

```
gs1 = get_diagonal_sums(fh2d)
ks1 = get_k_factors(fh2d);
```

In [17]:
```
gk1 = plot(ks1 .* dEmitted, gs1, legend=:topright, xlims=(-179, 179), xlabel="k-factor", ylabel="g(k)", label="g₁(k)")
```

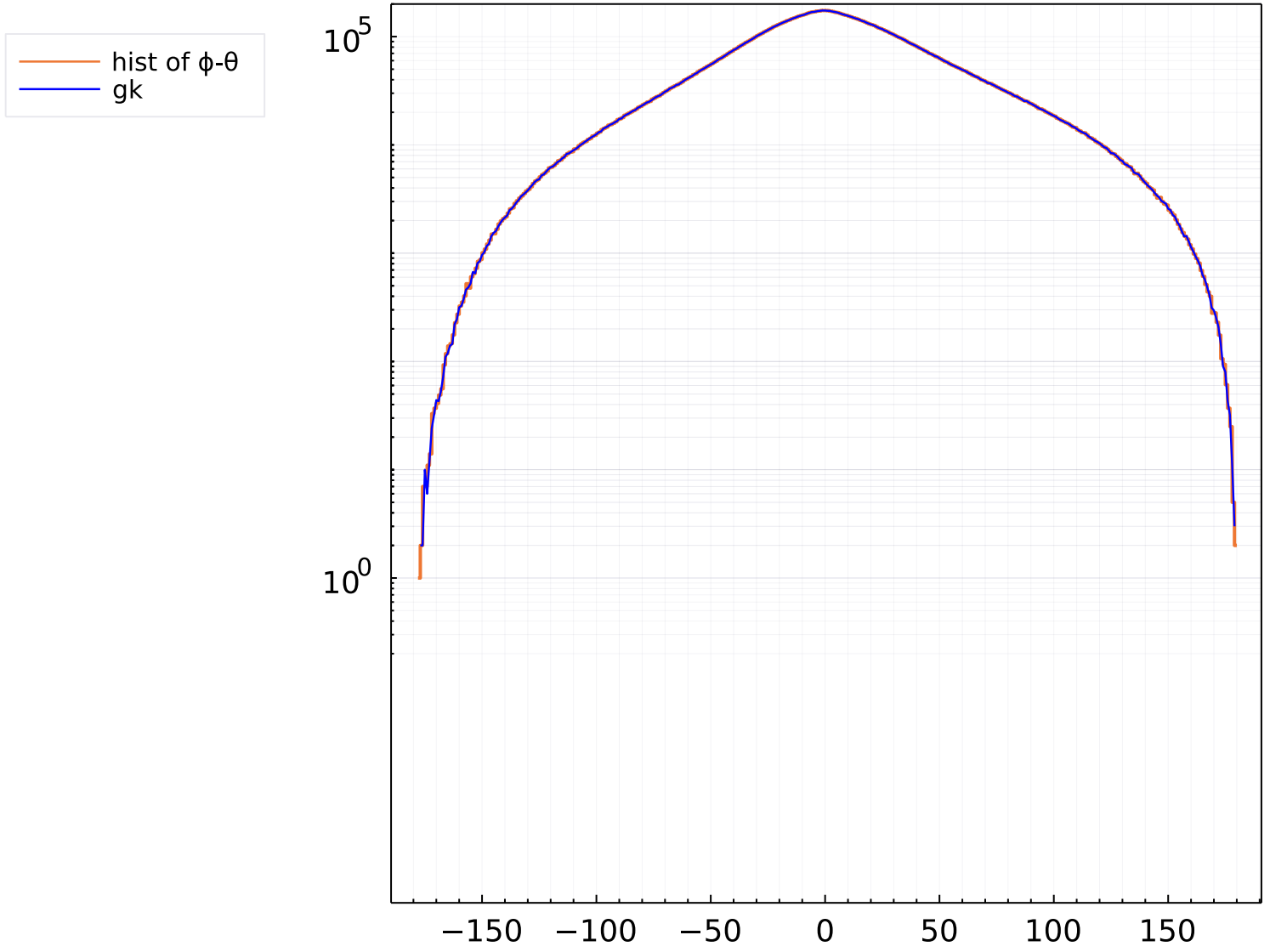
Out[17]:




In [18]:
```
k = (tree.thetaEscaped .- tree.thetaEmitted)
```

```
histogram(k, nbins = Int(180/dϕ*2-1), lw = 2, weights = tree.weights, label ="hist of ϕ-θ")
plot!(ks1 .* dEmitted, gs1, c = :blue, ylims = (1e-3, 2e5), yscale = :log10, label = "gk")
```

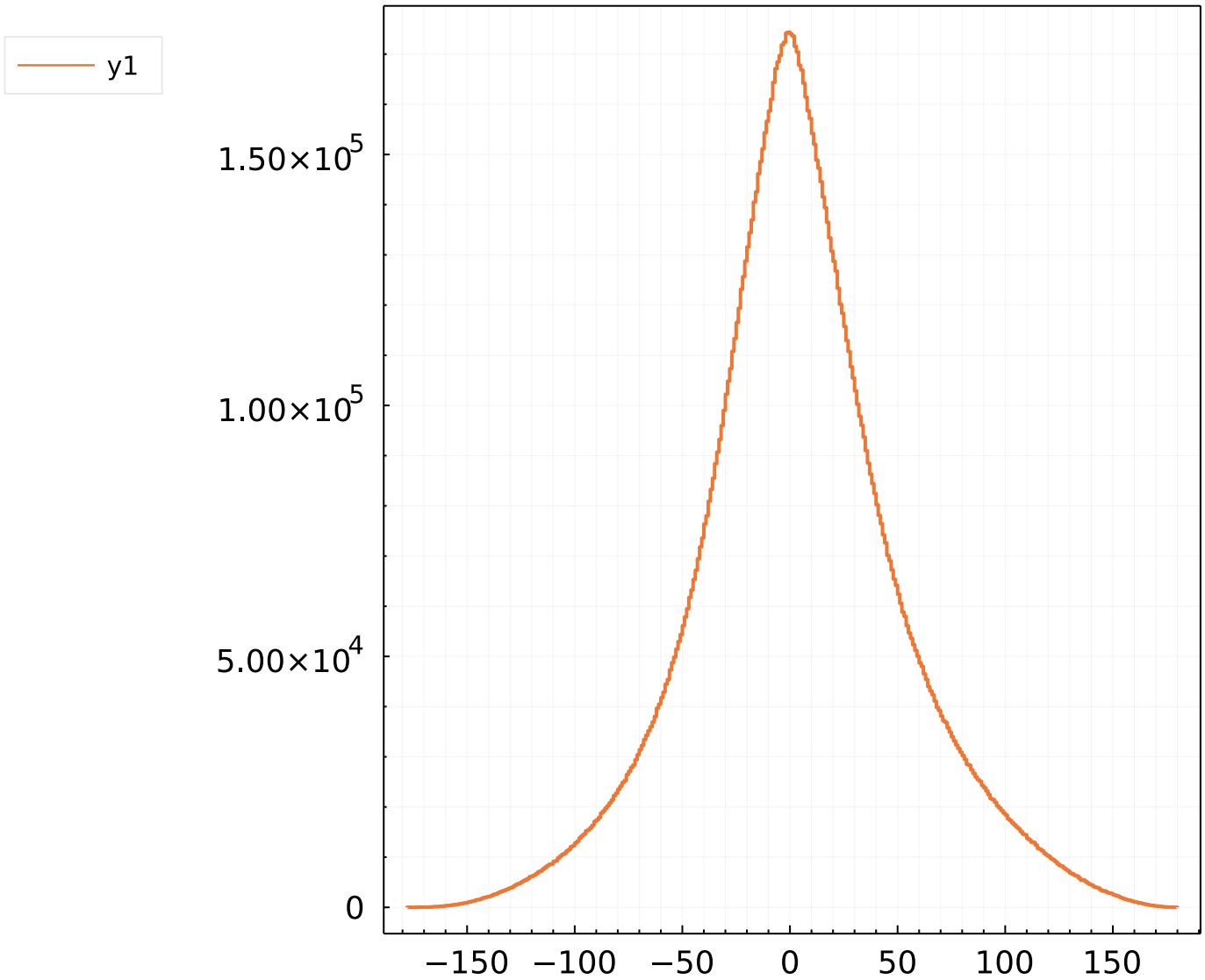
Out[18]:




```
┌ Warning: Invalid negative or zero value 0.0 found at series index 358 for log10 based yscale
└ @ Plots /home/shoram/.julia/packages/Plots/GGa6i/src/utils.jl:91
```

In [19]:
```
hh = histogram(k, nbins = Int(180/dϕ*2-1), lw = 2, weights = tree.weights)
```

Out[19]:

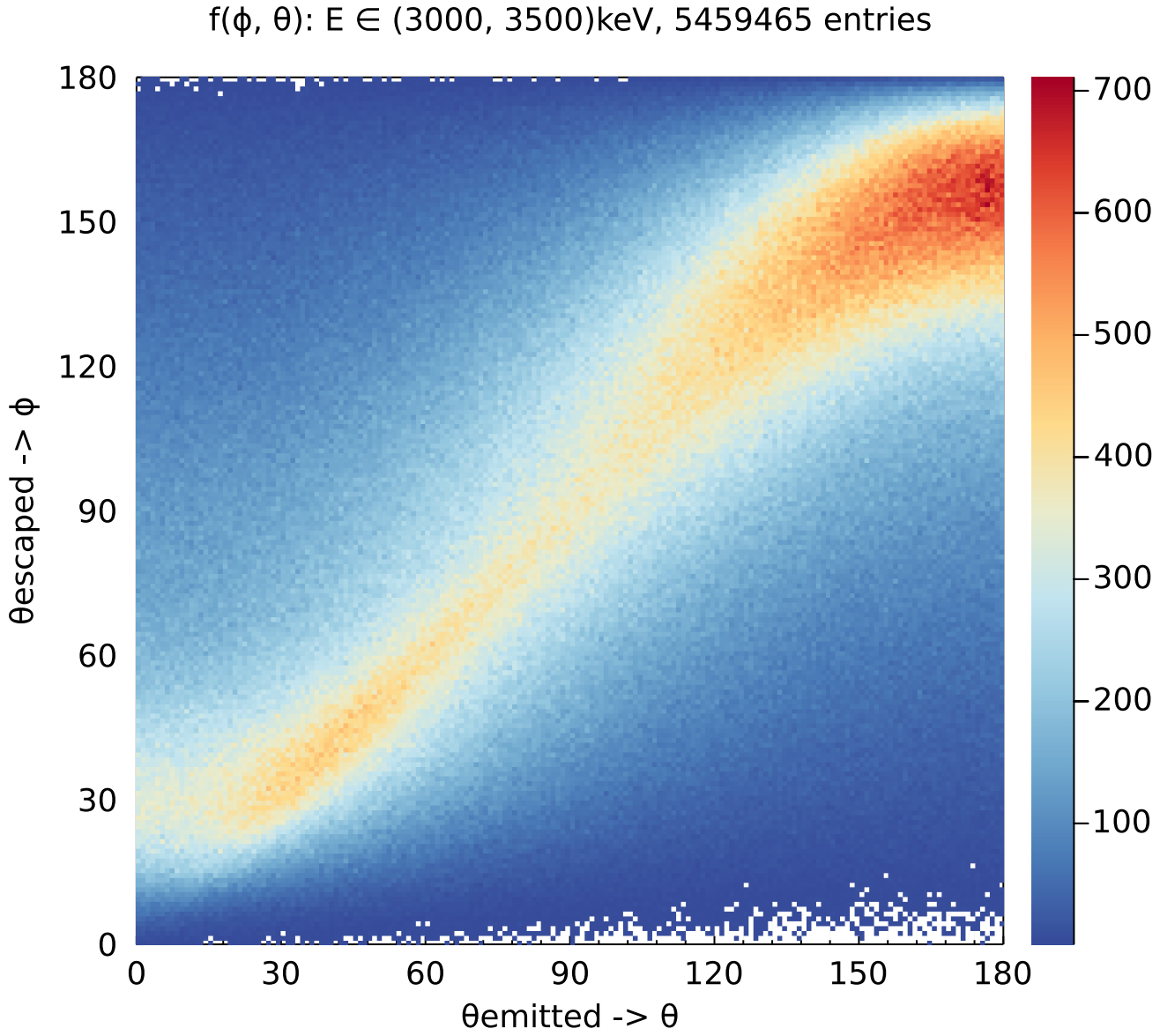



Applying various data cuts, we can observe how `g(k)` behaves. For example, if we apply an energy cut for the sum of electron energies, ie. $E \in (3000, 3500) keV$ we get...

In [20]:

```
gdf = @chain tree begin
    @subset((3000 .<= :ESum .<= 3500))
    @select(:thetaEscaped, :thetaEmitted, :ESum, :weights)
end

fh2d2 = Hist2D(                                          # h2d object similar to TH2D from ROOT
(gdf[!,2], gdf[!,1]),
    Weights(gdf.weights),
(minAngle:dEmitted:maxAngle, minAngle:dEmitted:maxAngle),
)

histogram2d(gdf[!,2], gdf[!,1];
    nbins        = (nBins, nBins),
    weights      = gdf.weights,
    xlabel       = "θemitted -> θ",
    ylabel       = "θescaped -> ϕ",
    legend       = :topright,
    title        = string("f(ϕ, θ): E ∈ (3000, 3500)keV, ", nrow(gdf), " entries"),
    lims         = (0, 180),
    aspect_ratio = 1,)
```

Out[20]:



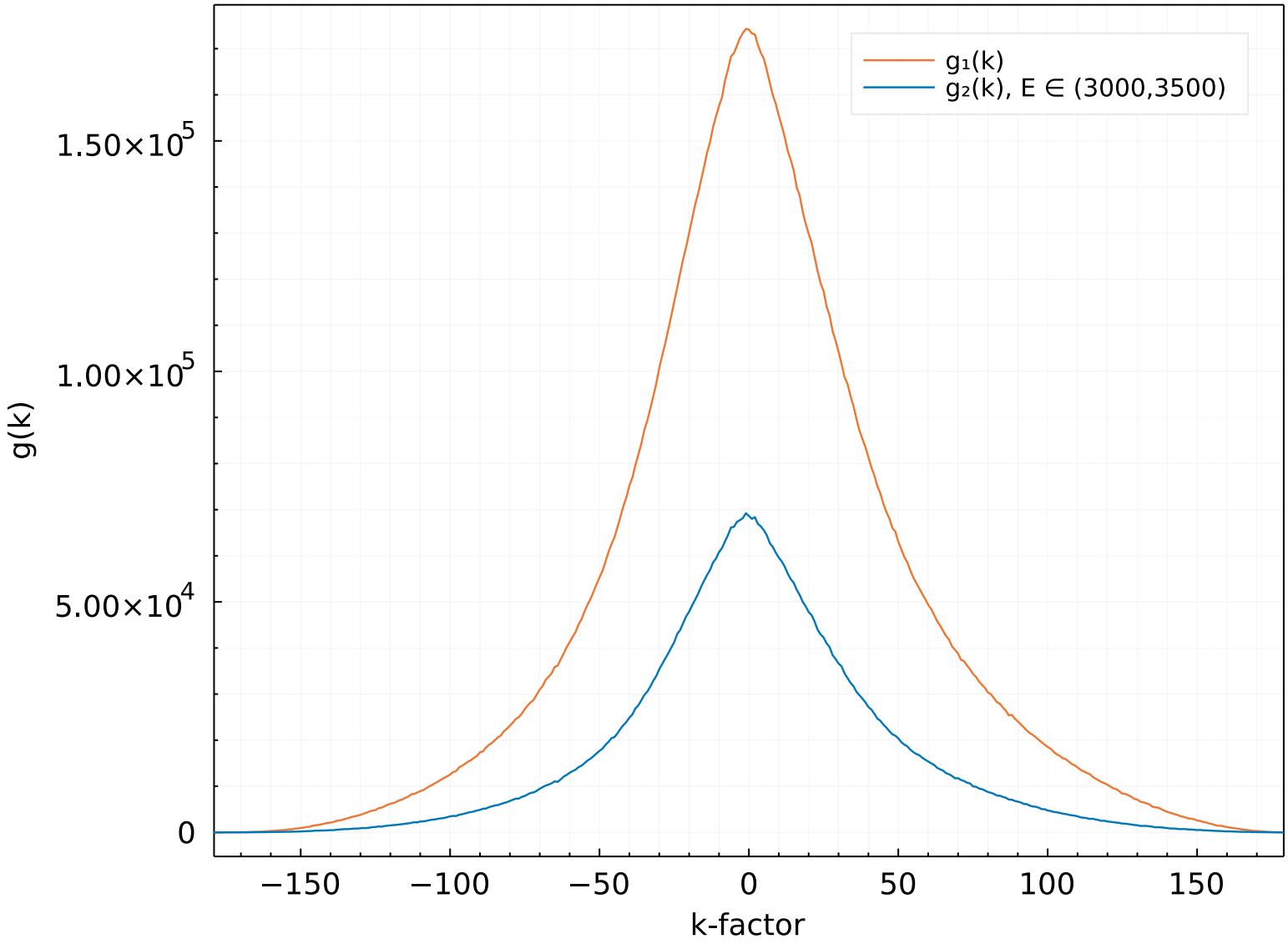

And the corresponding `g(k)` is

In [21]:

```
gs2 = get_diagonal_sums(fh2d2)
ks2 = get_k_factors(fh2d2);
```

In [22]:
```
plot!(gk1, ks2  .* dEmitted, gs2, legend=:topright, xlims=(-179, 179), xlabel="k-factor", ylabel="g(k)", label="g₂(k),
```

Out[22]:



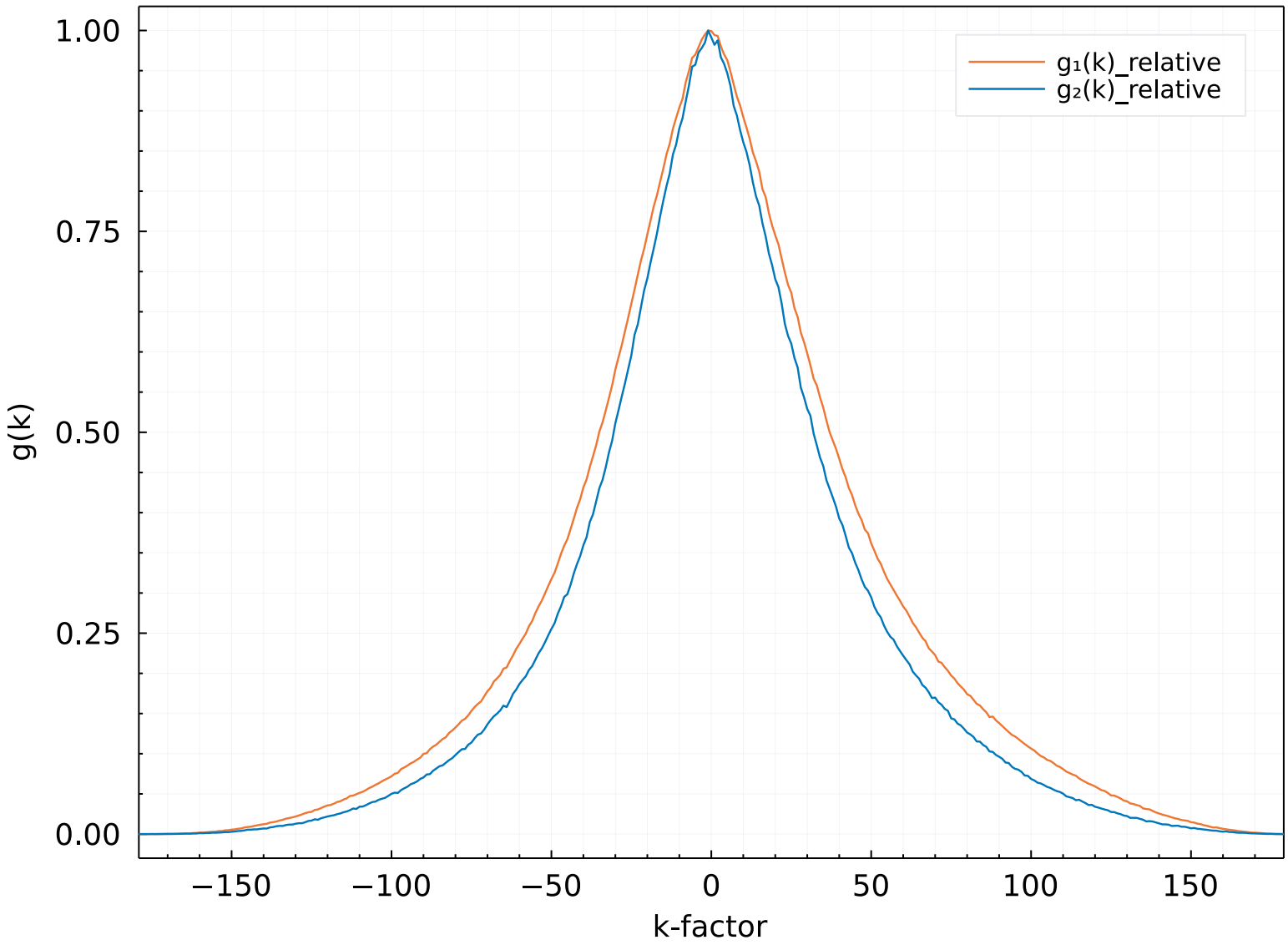

The first thing to notice is the reduced statistics. (For obvious reasons.) The more interesting however,

is to look at the dispersion around the k = 0 line. Ideally, the statistics would be sacrified for the improved reconstruction precision -> narrower g(k). This we can view better by looking at $g_1(k), g_2(k)$ normalized by their maximum.

In [23]:

```
gMax1 = maximum(gs1)
gMax2 = maximum(gs2)
plot(ks1  .* dEmitted, gs1 ./ gMax1, legend=:topright, xlims=(-179, 179), xlabel="k-factor", ylabel="g(k)", label="g₁(k
plot!(ks2 .* dEmitted, gs2 ./ gMax2, legend=:topright, xlims=(-179, 179), xlabel="k-factor", ylabel="g(k)", label="g₂(k
```

Out[23]:



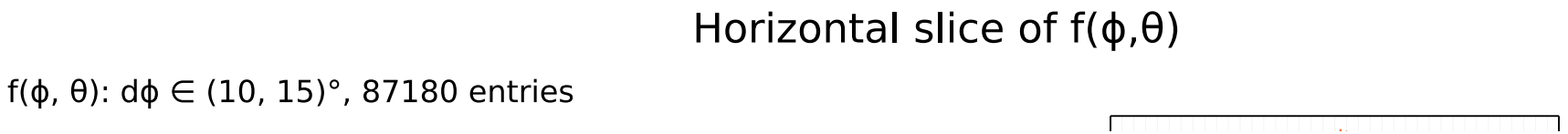

We can see that while applying an energy cut on the data results in decreased statistics, it did provide for a better reconstruction precision. We thus have a tool for comparing the effects of data cuts on the data.

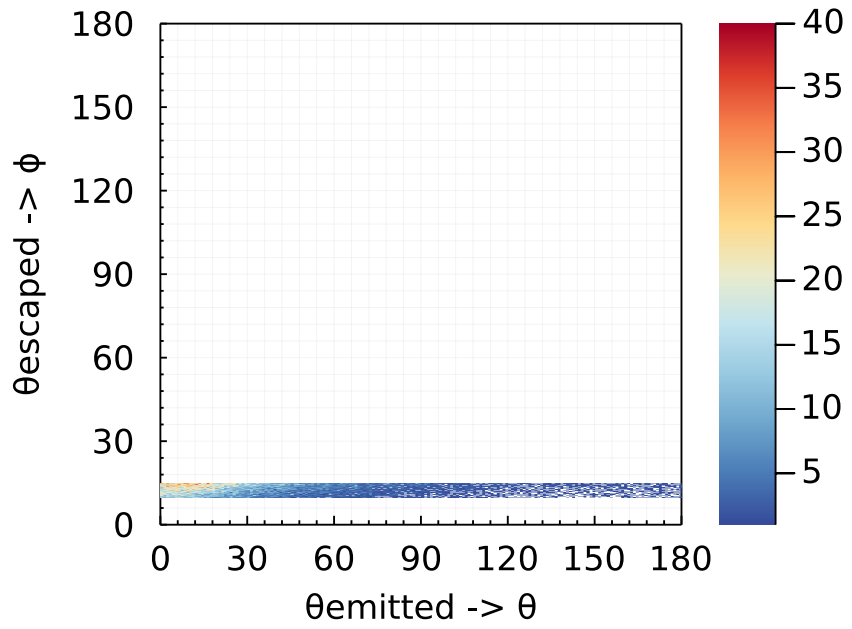
Next we look more in detail at individual $\phi$ slices. We first slice up `f(ϕ, θ)` horizontally in slices of $d\phi$ = $1\,\mathrm{deg}$. We look at the `g(k)` s of the individual slices. I.e. a horizontal slice with $\phi \in (10, 15)\,\mathrm{deg}$ and its $g(k)$ would look as:

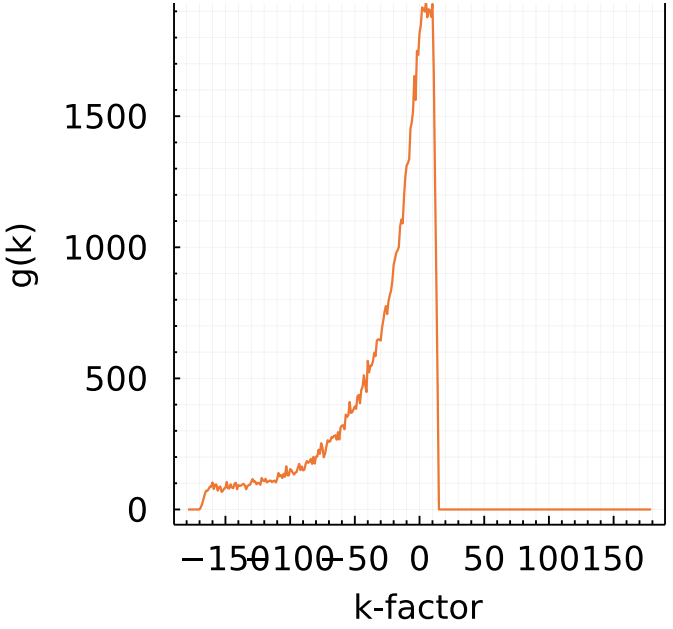
In [24]:

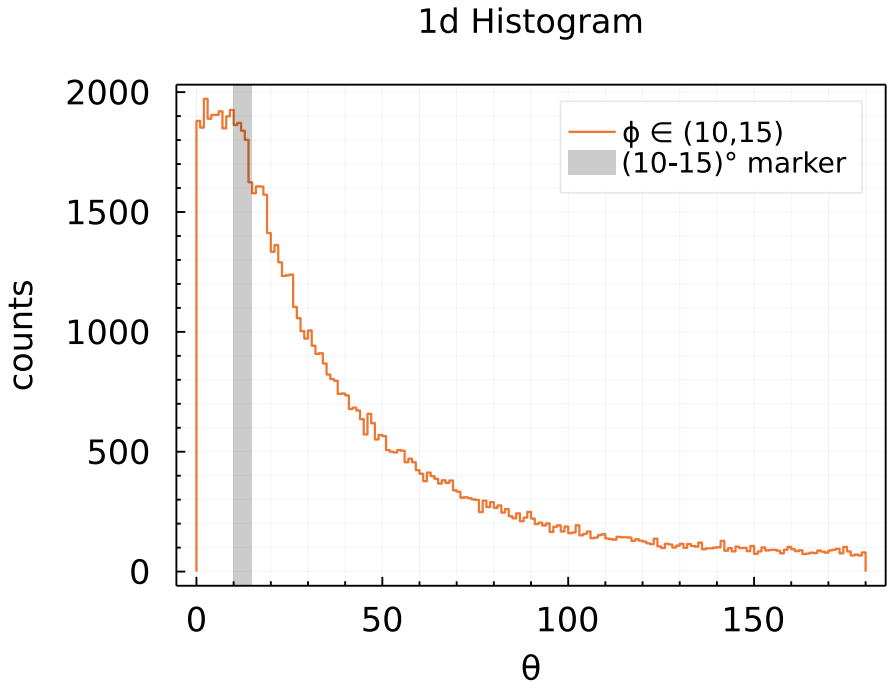
```
gdf = @chain tree begin
    @subset((10 .<= :thetaEscaped .<= 15))
    @select(:thetaEscaped, :thetaEmitted, :weights)
end

fh2d = Hist2D(
(gdf[!,2], gdf[!,1]),
    Weights(gdf.weights),
(minAngle:dEmitted:maxAngle, minAngle:dEmitted:maxAngle),
)

gs4 = get_diagonal_sums(fh2d)
ks4 = get_k_factors(fh2d);

h = histogram2d(gdf[!,2], gdf[!,1];
    nbins        = (nBins, nBins),
    weights      = gdf.weights,
    xlabel       = "θemitted -> θ",
    ylabel       = "θescaped -> ϕ",
    legend       = :topright,
    title        = string("f(ϕ, θ): dϕ ∈ (10, 15)°, ", nrow(gdf), " entries"),
    lims         = (0, 180),
    # aspect_ratio = 1,
    )
h1d = stephist(gdf.thetaEmitted, nbins = nBins, weights = gdf.weights,
                label ="ϕ ∈ (10,15)",xlabel ="θ", ylabel ="counts",
                title = "1d Histogram", legend =:topright)
vspan!([10,15], label ="(10-15)° marker", c =:black, alpha= 0.2)
p = plot(ks4, gs4, label = "", xlabel = "k-factor", ylabel ="g(k)")

l = @layout [a{0.5w} b; c{0.5w} _]
plot(h,p,h1d, layout = l, plot_title = "Horizontal slice of f(ϕ,θ)", size = (1000,800))
```

Out[24]:




θescaped -> ϕ
θemitted -> θ
0
30
60
90
120
150
180
40
35
30
25
20
15
10
5


g(k)
k-factor
0
500
1000
1500
−150
−100
−50
0
50
100
150


1d Histogram
counts
θ
ϕ ∈ (10,15)
(10-15)° marker
0
500
1000
1500
2000
0
50
100
150

Repeating this procedure, slicing $f(\phi, \theta)$ horizontally to cover the whole 0 - 180 degree range yields:

We can look at the individual lines of `g(k)`

In [25]:
```
dfGOrig = get_gs_df(tree, dϕ, sign)
matGOrig = df_to_mat(dfGOrig);
```

In [26]:
```
dfGOrig
```

Out[26]: 359 rows × 181 columns

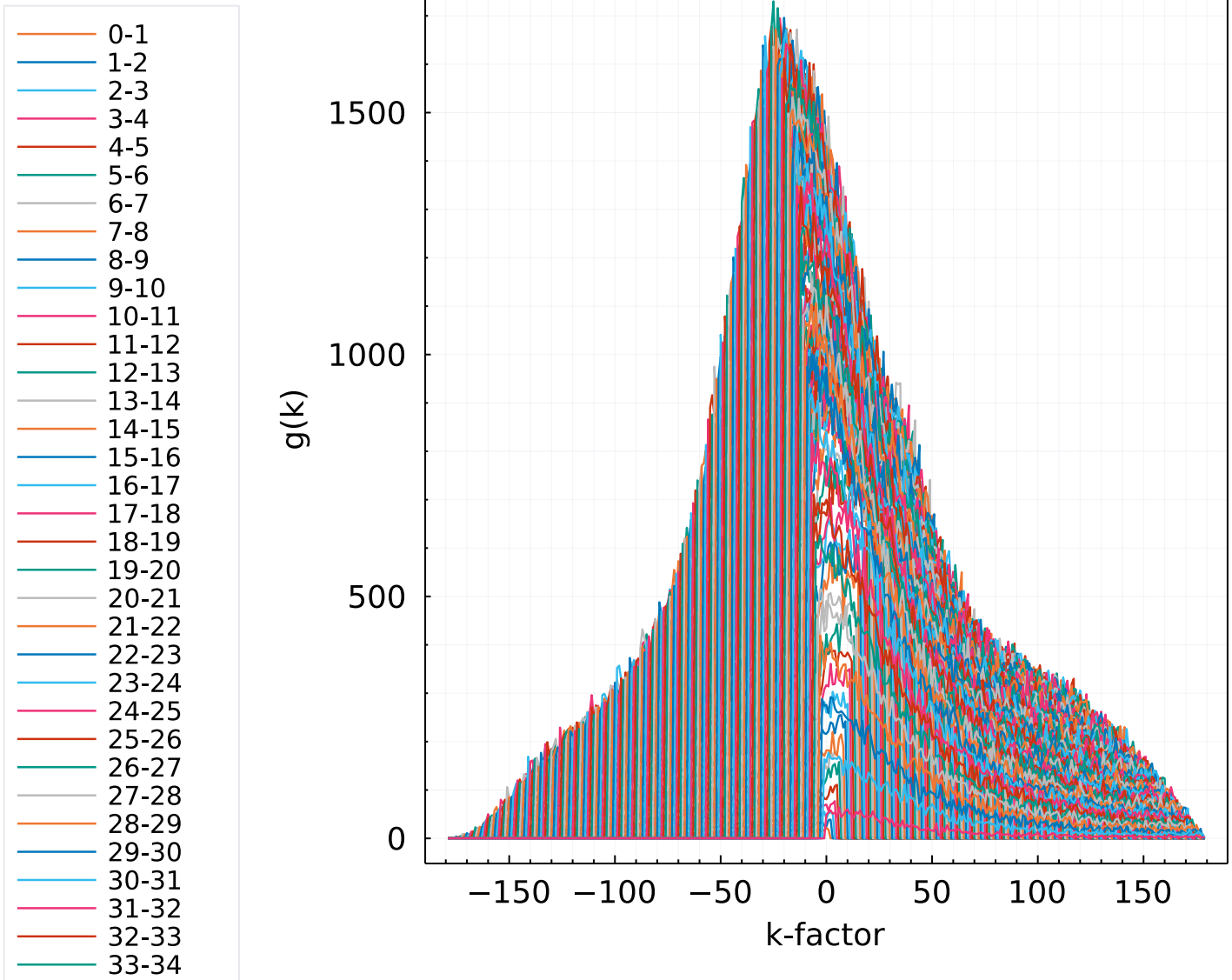
| | k | 0-1 | 1-2 | 2-3 | 3-4 | 4-5 | 5-6 | 6-7 | 7-8 | 8-9 | 9-10 | 10-11 | 11-12 | 12-13 | 13-14 | 14-15 | 15- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float |
| **1** | 179.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **2** | 178.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **3** | 177.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **4** | 176.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **5** | 175.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **6** | 174.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **7** | 173.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **8** | 172.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **9** | 171.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **10** | 170.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **11** | 169.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **12** | 168.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **13** | 167.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **14** | 166.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **15** | 165.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |

| | k | 0-1 | 1-2 | 2-3 | 3-4 | 4-5 | 5-6 | 6-7 | 7-8 | 8-9 | 9-10 | 10-11 | 11-12 | 12-13 | 13-14 | 14-15 | 15- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float64 | Float |
| **16** | 164.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **17** | 163.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **18** | 162.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **19** | 161.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **20** | 160.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [27]:

```
p = plot(xlabel ="k-factor", ylabel ="g(k)")
lbls = names(dfGOrig)
for i in 2:ncol(dfGOrig)-1
    plot!(dfGOrig.k, dfGOrig[!,i+1], label = lbls[i])
end
p
```

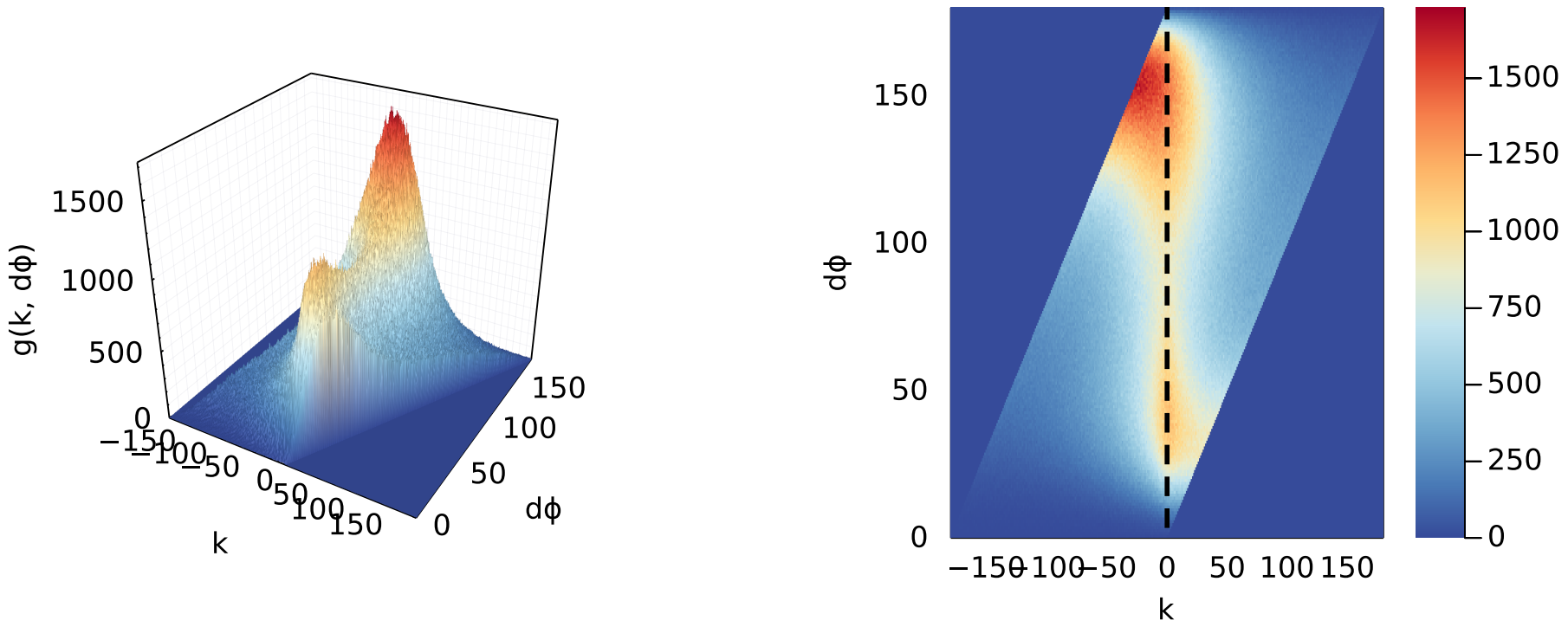
Out[27]:




Not much to see in this example. So many lines are difficult to decipher. However, if one were to look at the individual $d\phi$ cuts as a new dimension, we can look at the graph in the plane of $(k, d\phi)$ with z-direction being the value of $g(k, d\phi)$.

In [28]:
```
xRange = dϕ-180:dϕ:180-dϕ
yRange = 0:dϕ:180-dϕ
sf1 = surface(xRange, yRange,  matGOrig, legend =:none, xlabel ="k", ylabel ="dϕ", zlabel="g(k, dϕ)")
hm1 = heatmap(xRange, yRange,  matGOrig, ylabel ="dϕ", xlabel ="k" )
vline!([0], label ="", c = :black, lw = 3, s=:dash)
plot(sf1,hm1, size =(1000,400), layout = @layout [a{0.4w} b])
```

Out[28]:




Now we can see a few important features. First of all, there are two peaks visible in the left figure, with the higher peak (more statistics) being in the region of $130 < \phi < 170$ . Secondly, we can see the deviation of the peaks from the `k = 0` line in the right figure. There are two hotspots visible. First hotspot (corresponding to the lower peak in figure) is centered around $d\phi \approx 30\,\mathrm{deg}$ and is shifted slightly to the right of the `k = 0` line. The escaped angle overestimates the emitted angle. Second hotspot (corresponding to the higher peak in figure) is centered around $d\phi \approx 150\,\mathrm{deg}$ and is shifted visibly to the left of the `k = 0` line. The escaped angle underestimates the emitted angle. Lastly, we

can see that the regions $\phi \approx 0\,\mathrm{deg}$ and $\phi \approx 180\,\mathrm{deg}$ are squeezed toward higher, lower angles, respectively.

Furthermore, we can also look at how the individual $d\phi$ slices look in terms of statistical variables (mean, mode, median). For each variable, obtained from the $\phi(\theta)$ distributions.

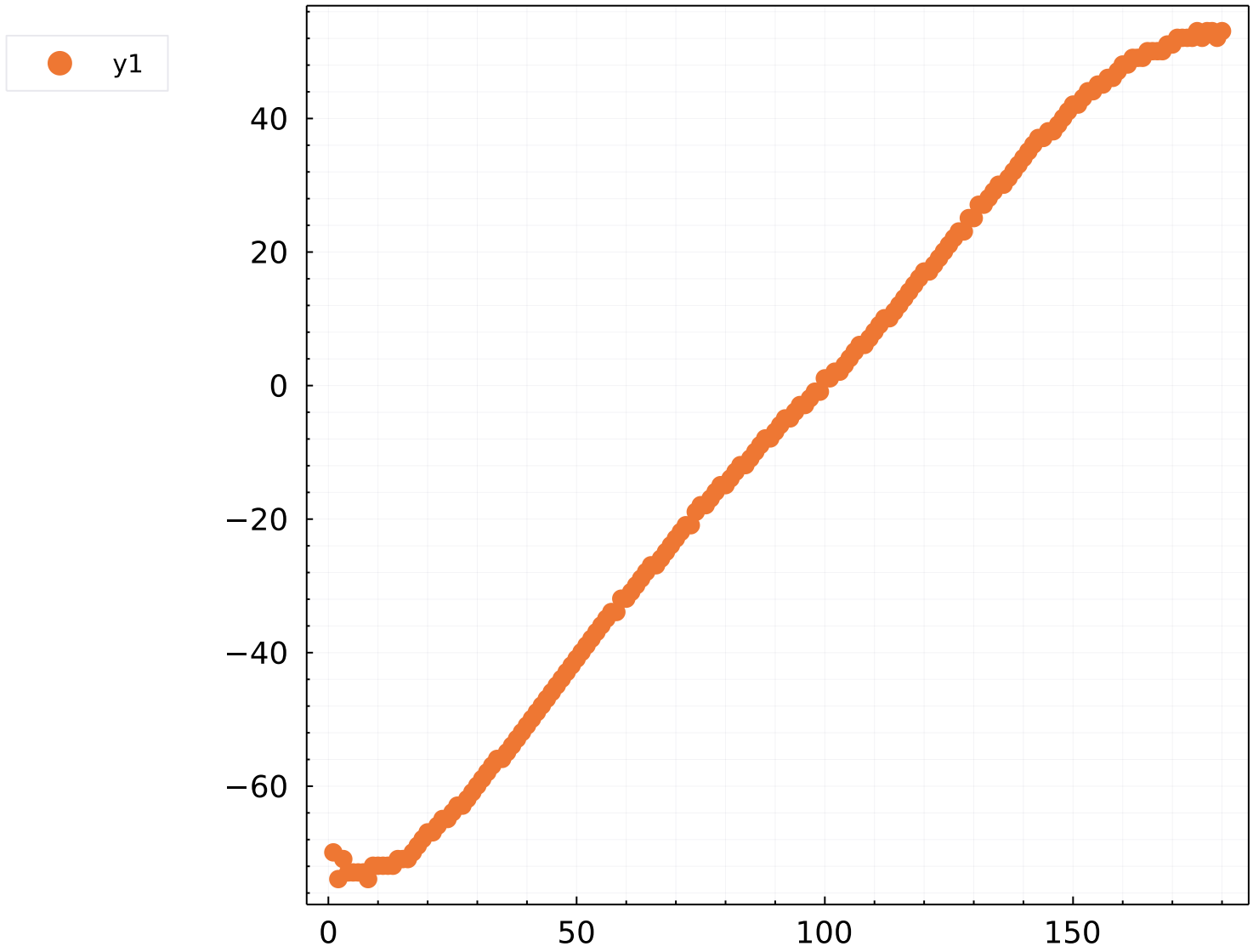
In [29]:

```
means   = Vector{Float64}(undef, Int(180/dϕ))
modes   = Vector{Float64}(undef, Int(180/dϕ))
medians = Vector{Float64}(undef, Int(180/dϕ))
mediansFHist = Vector{Float64}(undef, Int(180/dϕ))

for (i, ϕ) in enumerate(1:dϕ:180)
    cutEdges1 = get_cut_edges(ϕ - 1, 1, dϕ, "p")                    # provides the lower and upper cut
    sdf       = @chain tree begin                                   # filter out the dataframe
        @select(:thetaEscaped, :thetaEmitted, :weights)                     # keeps only the two angles columns
        @subset((cutEdges1[1] .<= :thetaEscaped .<= cutEdges1[2]))  # keeps only rows where ϕ is within the cut edges
    end

#     thetas = sdf.thetaEmitted
#     w = Weights(rho.(cosd.(thetas)))

    stats = get_slice_stats(
                            cutEdges1[1],  # ϕmin
                            cutEdges1[2],  # ϕmax
                            0,             # Emin
                            3500,          # Emax
                            sdf.thetaEmitted,
                            dϕ,
                            Weights(sdf.weights)
                            )

    h = Hist1D(thetas, w, (0:dϕ:180))
    mediansFHist[i] = StatsBase.median(h)

    means[i] = stats[5]
    modes[i] = stats[6] .+ dϕ/2
    medians[i] =  stats[7]
end
```

In [30]:

```
scatter( medians .- mediansFHist )
```

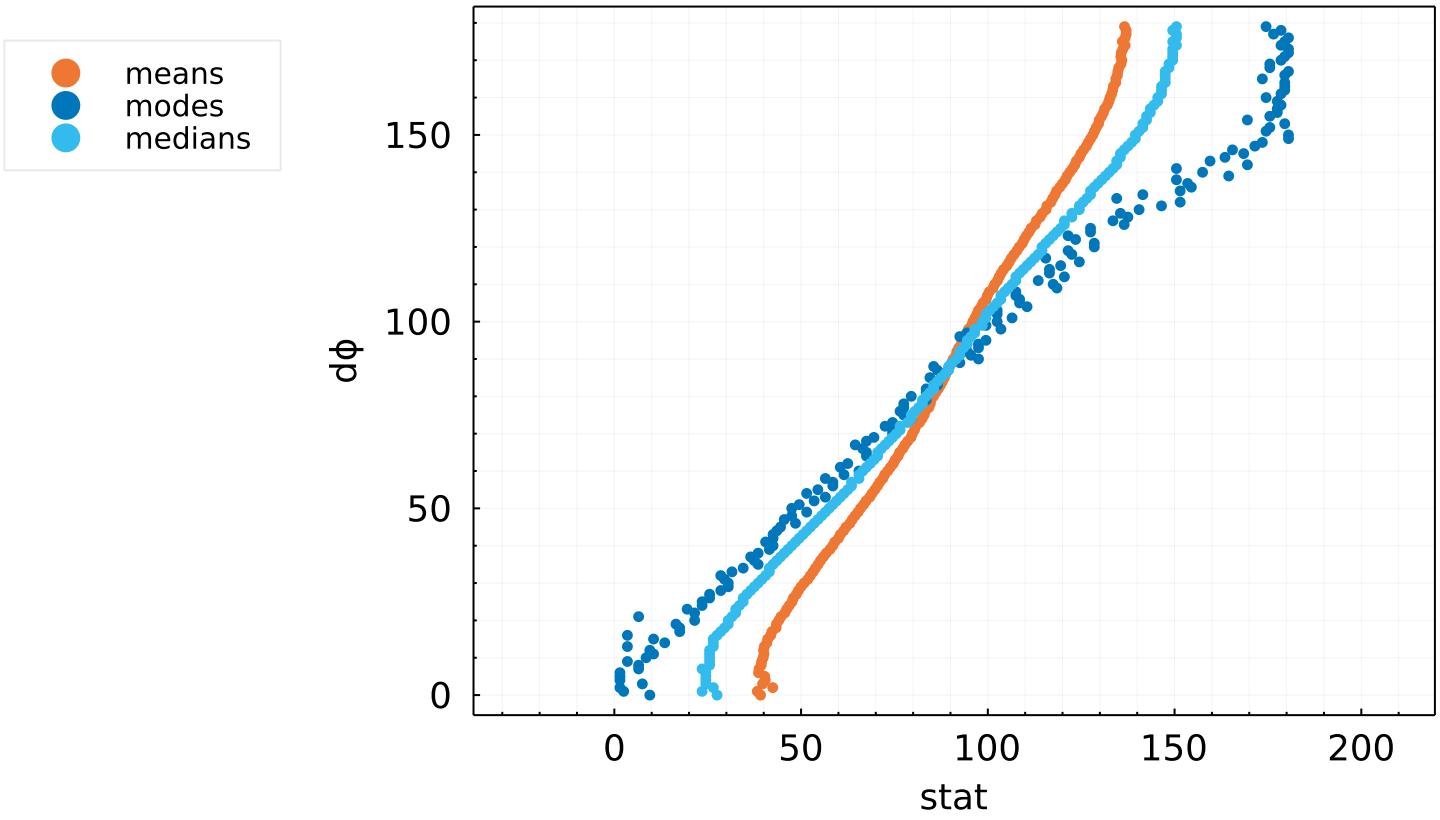
Out[30]:




In [31]:

```
scatter( means, xPts, ms = 3, label = "means" , xlabel = "stat", ylabel ="dϕ", aspect_ratio =1)
scatter!( modes, xPts, ms = 3, label ="modes" )
scatter!( medians, xPts, ms = 3, label ="medians" )
```

Out[31]:

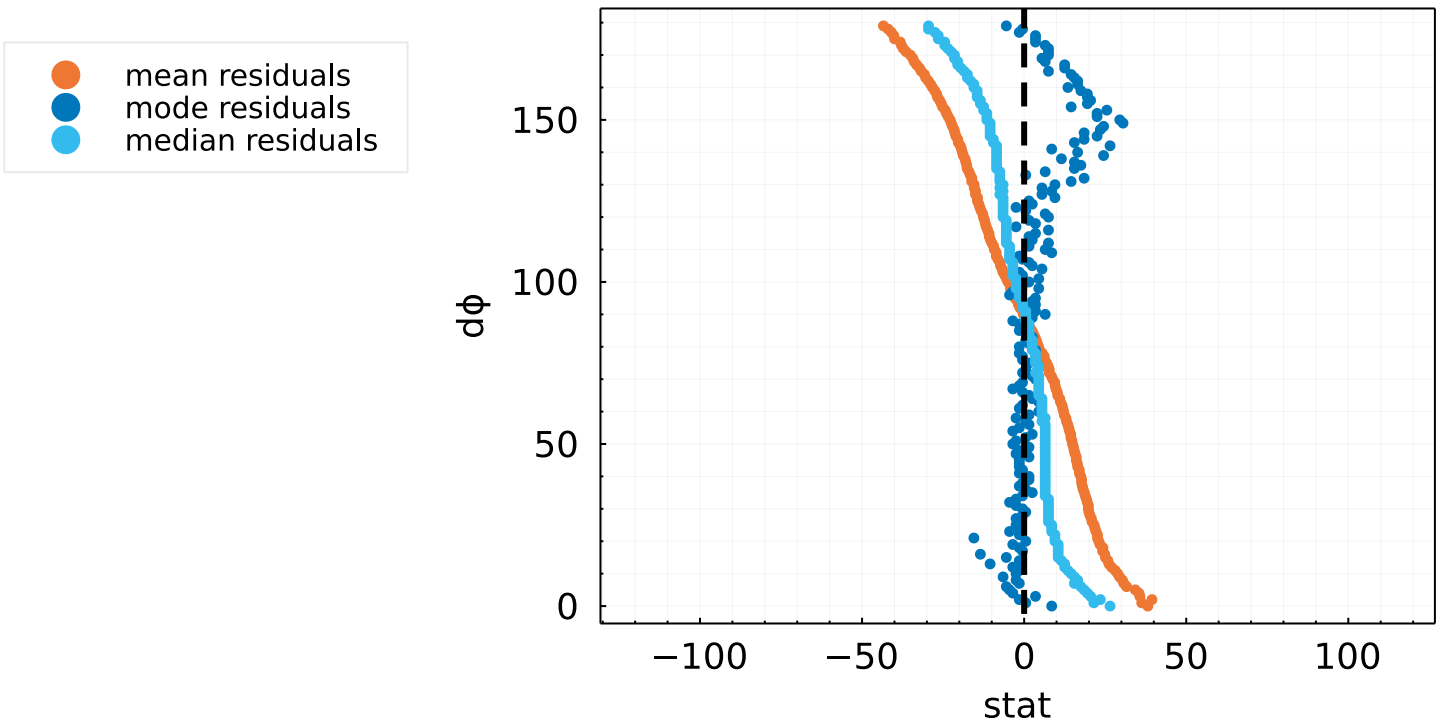



In [32]:

```
res_means = [ means[i] - y(i*dϕ) for i in 1:length(means) ]
res_modes = [ modes[i] - y(i*dϕ) for i in 1:length(modes) ]
res_medians = [ medians[i] - y(i*dϕ) for i in 1:length(medians) ]
```

```
scatter( res_means, xPts, ms = 3, label = "mean residuals" , xlabel = "stat", ylabel ="dϕ", aspect_ratio =1)
scatter!( res_modes, xPts, ms = 3, label ="mode residuals" )
scatter!( res_medians, xPts, ms = 3, label ="median residuals" )
vline!([0], label ="", c = :black, lw = 3, s=:dash)
```

Out[32]:



Ideally, we would want to have the peaks centered around the `k=0` line. To do that, we can perform justified manipulation of $\phi$ in terms of shifting the entire row ($d\phi$ slice) to the right or left by some set amount `s` [$s \in (-180, 180)$]. The new (modified) angle is defined as $\phi' = \phi + s$. This may reduce the obtained statistics (if an angle is shifted toward unphysical values, ie. $\phi > 180$), however if better precision is achieved, it may be worthwile.

In the first case, let's try to shift each row so that the `mean` is centered at `k=0` (shift residual to 0). To determine whether this gives us better or worse result, we compare the rms value of each slice before and after shifting.

In [33]:
```
modTree2 = @chain tree begin
    @select(:thetaEmitted, :thetaEscaped, :weights)
    @rtransform :bin =  get_bin_center(:thetaEscaped, Int(180/dϕ))  # create a vector of bin centers (which bin ϕ falls
    @transform :thetaEscapedOld = :thetaEscaped                     # create a copy of the old ϕ (for comparison only)

    @rtransform :thetaEscaped = :thetaEscapedOld + res_means[Int(ceil(:bin/dϕ))] # shift ϕ by s: ϕ' = ϕ + s
    @subset( 0 .< :thetaEscaped .< 180) # keep only physical angles
end
```

Out[33]: 15,747,968 rows × 5 columns

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| 1 | 89.6031 | 92.5524 | 1 | 94.5 | 94.6325 |
| 2 | 96.965 | 120.722 | 1 | 139.5 | 139.497 |
| 3 | 154.233 | 120.144 | 1 | 138.5 | 138.388 |
| 4 | 67.3403 | 129.132 | 1 | 154.5 | 154.281 |
| 5 | 11.2133 | 67.4527 | 1 | 53.5 | 53.0724 |
| 6 | 168.417 | 85.1413 | 1 | 81.5 | 81.0901 |
| 7 | 122.632 | 79.0214 | 1 | 69.5 | 69.6012 |
| 8 | 4.96695 | 53.8336 | 1 | 34.5 | 34.9839 |

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| **9** | 79.857 | 73.0331 | 1 | 61.5 | 61.1214 |
| **10** | 43.0942 | 53.2077 | 1 | 33.5 | 33.994 |
| **11** | 35.1387 | 97.3047 | 1 | 104.5 | 104.069 |
| **12** | 150.65 | 96.4782 | 1 | 101.5 | 101.923 |
| **13** | 56.3877 | 69.4157 | 1 | 55.5 | 55.6631 |
| **14** | 128.995 | 92.7602 | 1 | 94.5 | 94.8403 |
| **15** | 63.2692 | 58.4208 | 1 | 40.5 | 40.9504 |
| **16** | 40.4733 | 68.9002 | 1 | 55.5 | 55.1476 |
| **17** | 124.468 | 123.096 | 1 | 142.5 | 142.771 |
| **18** | 128.263 | 43.8765 | 1 | 20.5 | 20.8424 |
| **19** | 105.706 | 135.385 | 1 | 172.5 | 172.673 |
| **20** | 142.67 | 102.595 | 1 | 112.5 | 112.558 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [34]:
```
nrow(modTree2)/nrow(tree) # how much statistics is left (#_of_original/#_of_shifted)
```

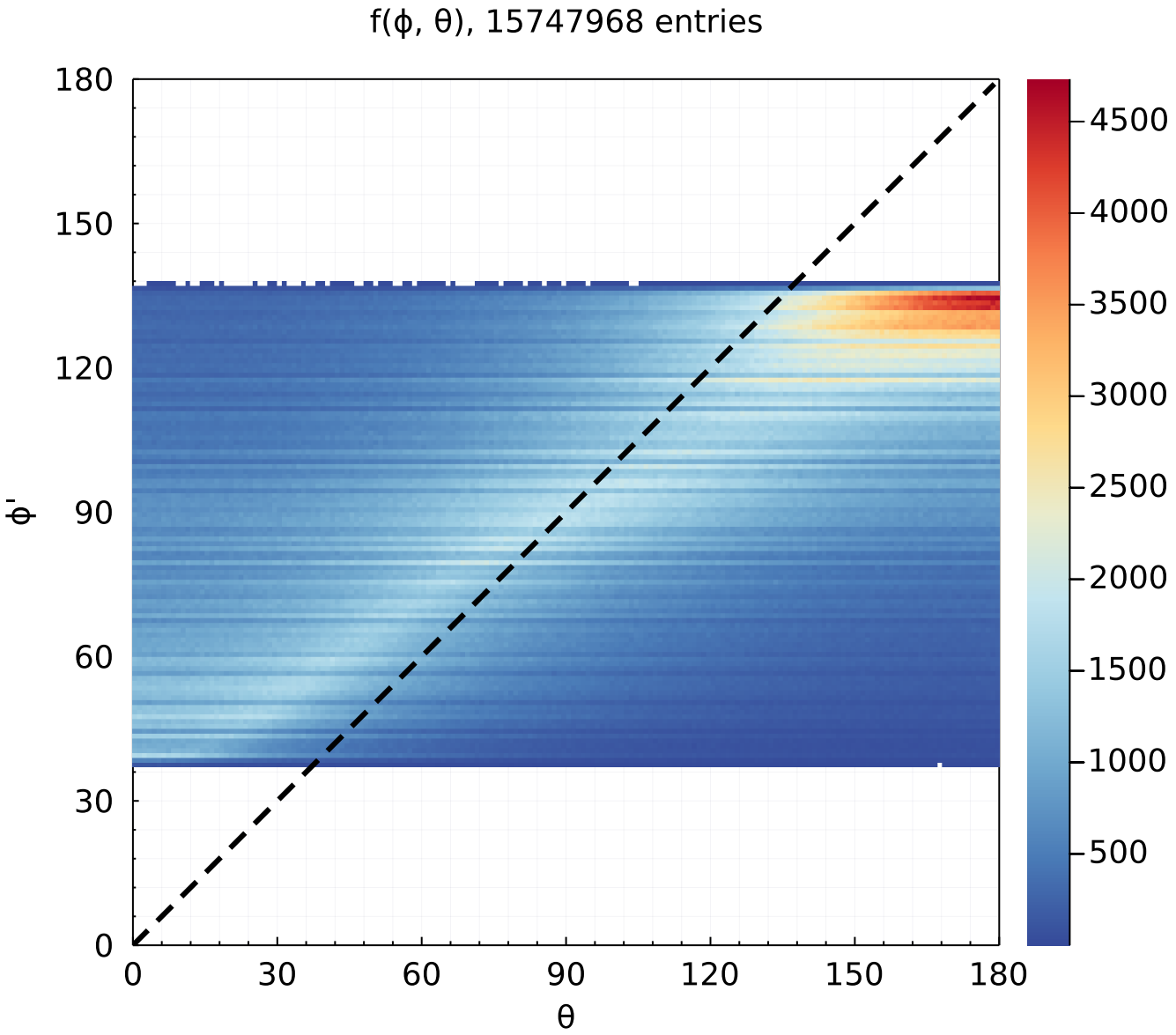
Out[34]:
```
1.0
```

In [35]:
```
dfG2 = get_gs_df(modTree2, dϕ, sign)
matG2 = df_to_mat(dfG2);
```

In [36]:
```
h2d2 = histogram2d(modTree2.thetaEmitted, modTree2.thetaEscaped,
    nbins        = (nBins, nBins),
    weights      = modTree2.weights,
    xlabel       = "θ",
    ylabel       = "ϕ'",
    legend       = :topright,
    title        = string("f(ϕ, θ), ", nrow(modTree2), " entries"),
    lims         = (0, 180),
```

```
    aspect_ratio = 1,)
plot!(xPts, xPts, label ="", c= :black, style= :dash, lw =3)
```

Out[36]:



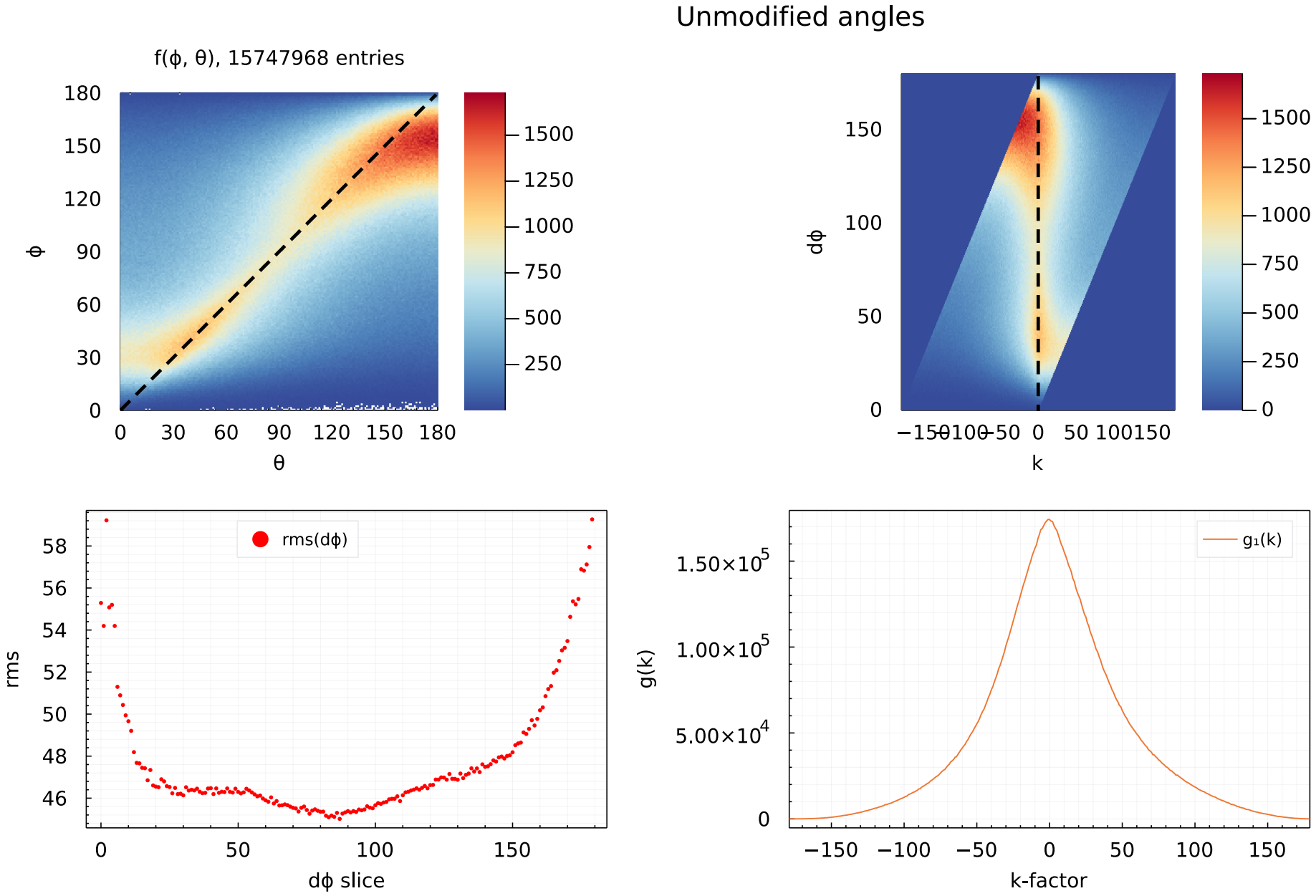

The seemingly discrete distribution is due to the fact that each slice has been shifted by a discrete number and this has shifted some angles toward certain slices more, or less. To view the effects of the

shift, let's compare the 2d histograms, contour plots and the rms values for the two datasets.

In [37]:

```
h2d1 = histogram2d(
    tree.thetaEmitted,
    tree.thetaEscaped;
    nbins        = (nBins, nBins),
    weights      = tree.weights,
    xlabel       = "θ",
    ylabel       = "ϕ",
    legend       = :topright,
    title        = string("f(ϕ, θ), ", nrow(tree), " entries"),
    lims         = (0, 180),
    aspect_ratio = 1,
)
plot!(xPts, xPts, label ="", c= :black, style= :dash, lw =3)
rms1 = [ get_rms(dfGOrig[:,i], dfGOrig[:,1]) for i in 2:ncol(dfGOrig) ]

sct1 = scatter( xPts, rms1, label ="rms(dϕ)", ms=2, legend=:top, xlabel ="dϕ slice", ylabel ="rms", c= :red )

fh2d1 = Hist2D((tree.thetaEmitted,tree.thetaEscaped),
    Weights(tree.weights),
    (minAngle:dEmitted:maxAngle, minAngle:dEmitted:maxAngle))
gs1 = get_diagonal_sums(fh2d1)
ks1 = get_k_factors(fh2d1);

gk1 = plot(ks1 .* dEmitted, gs1, legend=:topright, xlims=(-179, 179), xlabel="k-factor", ylabel="g(k)", label="g₁(k)")

# plot(title, h2d1, hm1, sct1, layout = @layout[a{0.05h};b c; d _] , size = (1100, 800))
plot(h2d1, hm1, sct1, gk1, layout = @layout[a b; c d] , size = (1200, 800), plot_title= "Unmodified angles")
```

Out[37]:

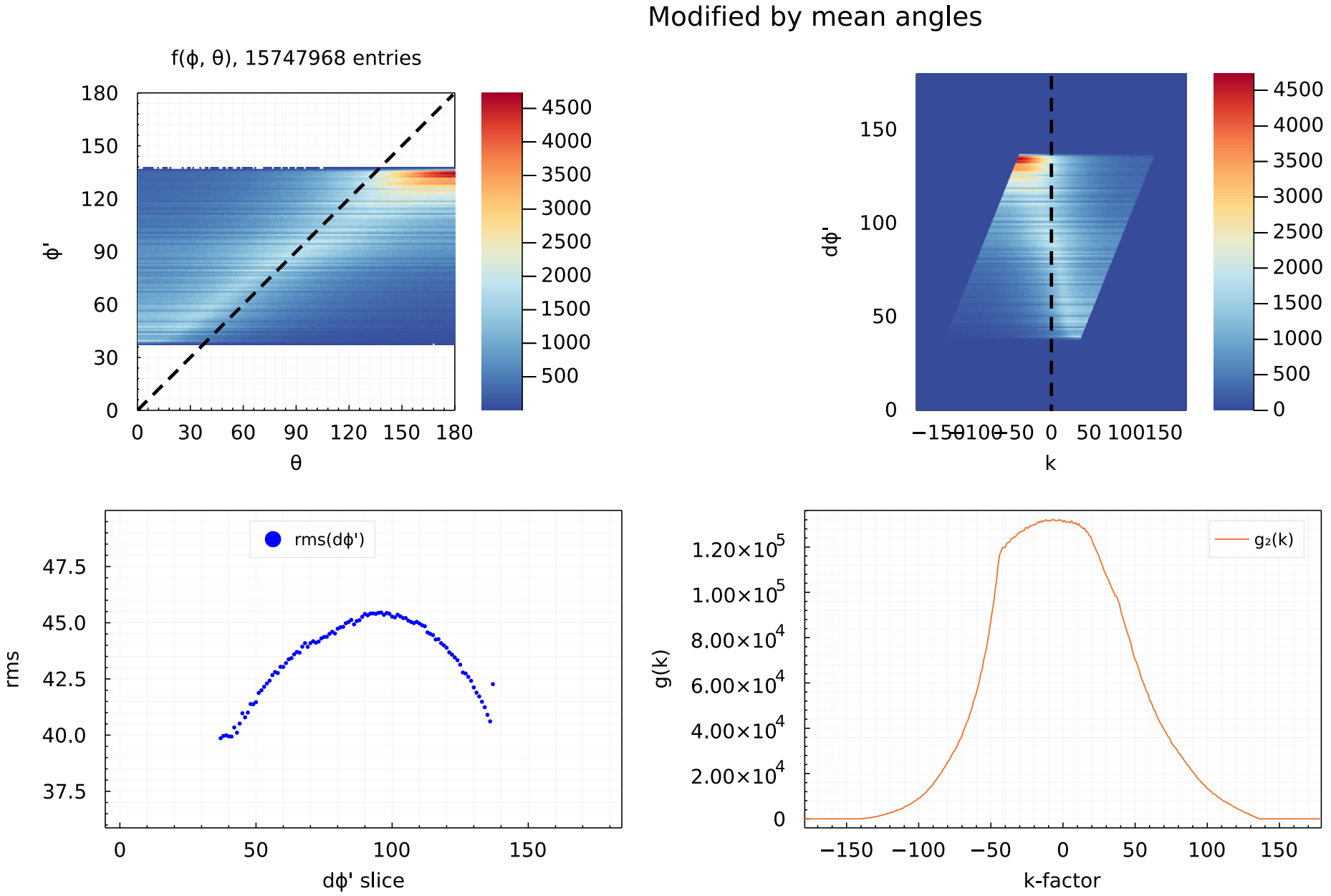



In [38]:

```
hm2 = Plots.heatmap(xRange, yRange,  matG2, ylabel ="dϕ'", xlabel ="k" )
vline!([0], label ="", c = :black, lw = 3, s=:dash)
```

```
rms2 = [ get_rms(dfG2[:,i], dfG2[:,1]) for i in 2:ncol(dfG2) ]
yMin = 0.9*minimum(filter(x -> x .> 0, rms2)) # get the minimum rms value, excluding 0
yMax = 1.1*maximum(filter(x -> x .> 0, rms2))

sct2 = scatter( xPts, rms2, label ="rms(dϕ')", ms=2,
                legend=:top, xlabel ="dϕ' slice", c =:blue,
                ylabel ="rms", ylims = (yMin, yMax) )

fh2d2 = Hist2D((modTree2.thetaEmitted, modTree2.thetaEscaped),
    Weights(modTree2.weights),
    (minAngle:dEmitted:maxAngle, minAngle:dEmitted:maxAngle))
gs2 = get_diagonal_sums(fh2d2)
ks2 = get_k_factors(fh2d2);

gk2 = plot(ks2 .* dEmitted, gs2, legend=:topright, xlims=(-179, 179),
           xlabel="k-factor", ylabel="g(k)", label="g₂(k)")



plot(h2d2, hm2, sct2, gk2, layout = @layout[a b; c d] , size = (1200, 800),
    plot_title= "Modified by mean angles")
```

Out[38]:

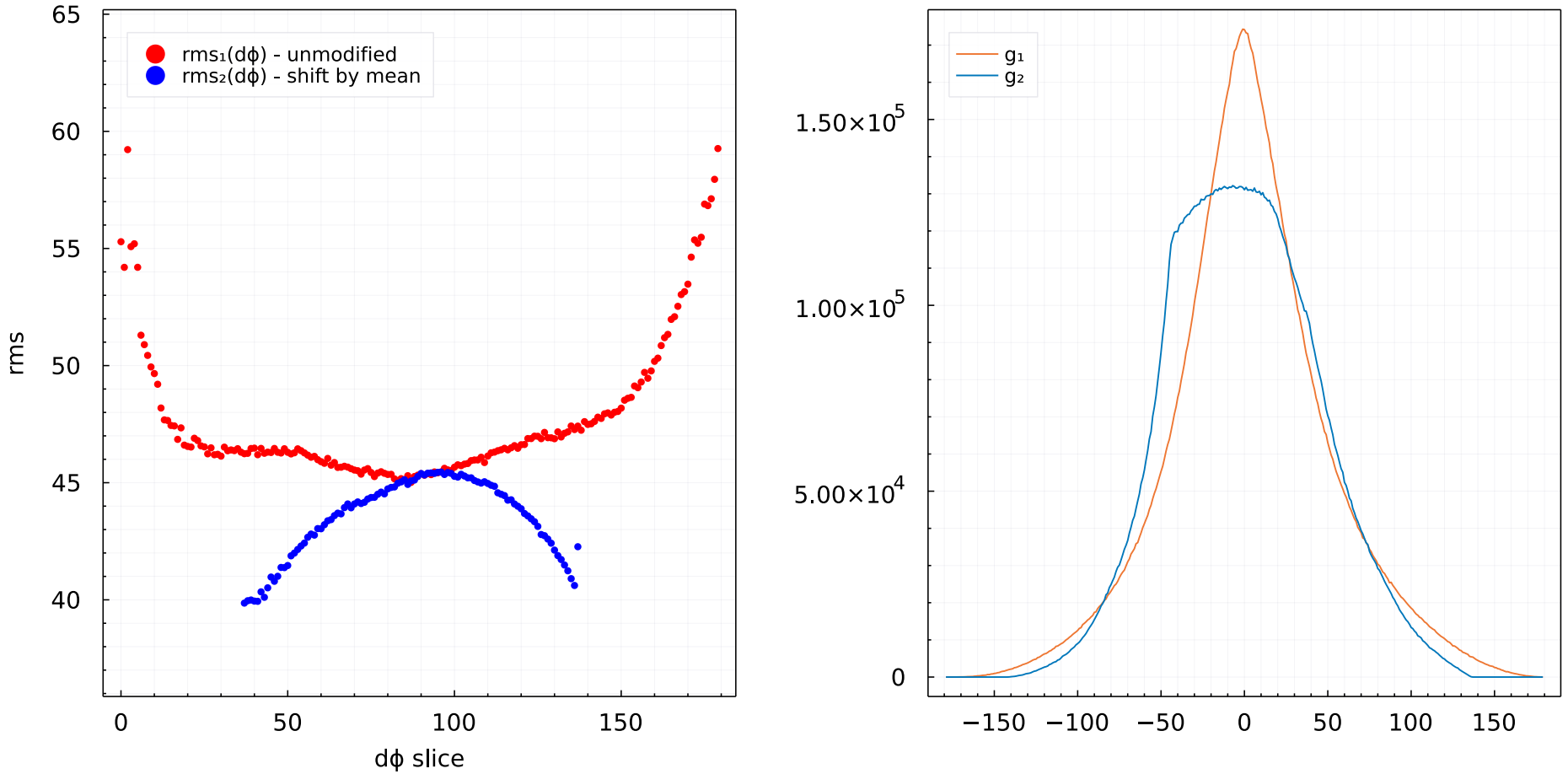



In [39]:

```
scComp2 = scatter( [xPts xPts], [rms1 rms2],  label =["rms₁(dϕ) - unmodified" "rms₂(dϕ) - shift by mean"],
        ms=3, c = [:red :blue],
        legend=:top, xlabel ="dϕ slice", ylabel ="rms", ylims = (yMin, 1.1*maximum(rms1)))
```

```
gkComp2 = plot( [ks1 .* dEmitted, ks2 .* dEmitted], [gs1, gs2], label = ["g₁" "g₂"], legend = :topleft  )
plot(scComp2, gkComp2, size = (1200, 600))
```

Out[39]:




In [40]:

```
rmsTotalUnModded = get_rms(gs1, ks1 .* dEmitted )
```

Out[40]:

```
46.98587209537262
```

In [41]:

```
rmsTotalModded = get_rms(gs2, ks2 .* dEmitted )
```

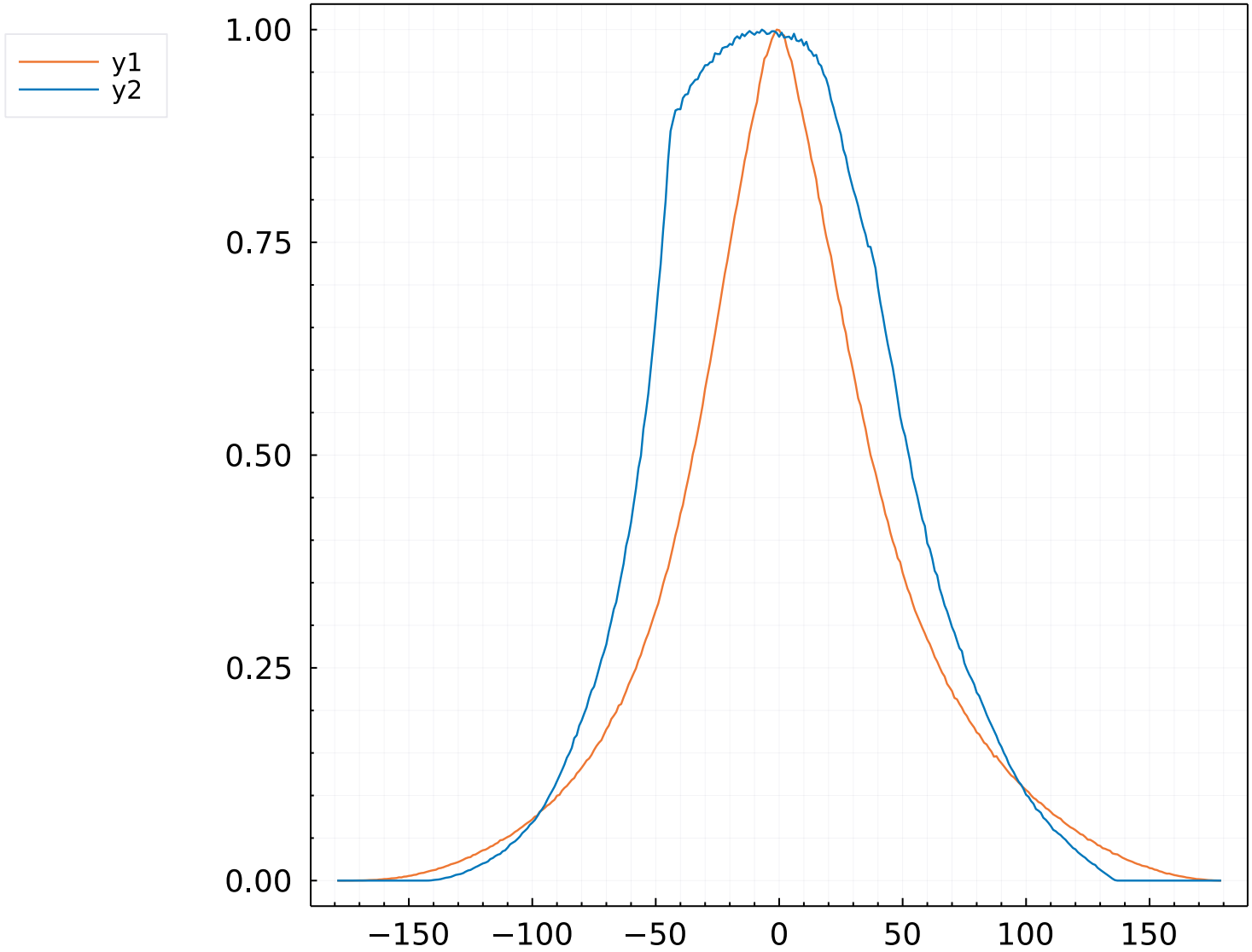
Out[41]:

```
43.84175272517404
```

In [42]:

```
plot(ks1 .* dEmitted, gs1 ./ maximum(gs1)  )
plot!(ks2 .* dEmitted, gs2 ./ maximum(gs2)  )
```

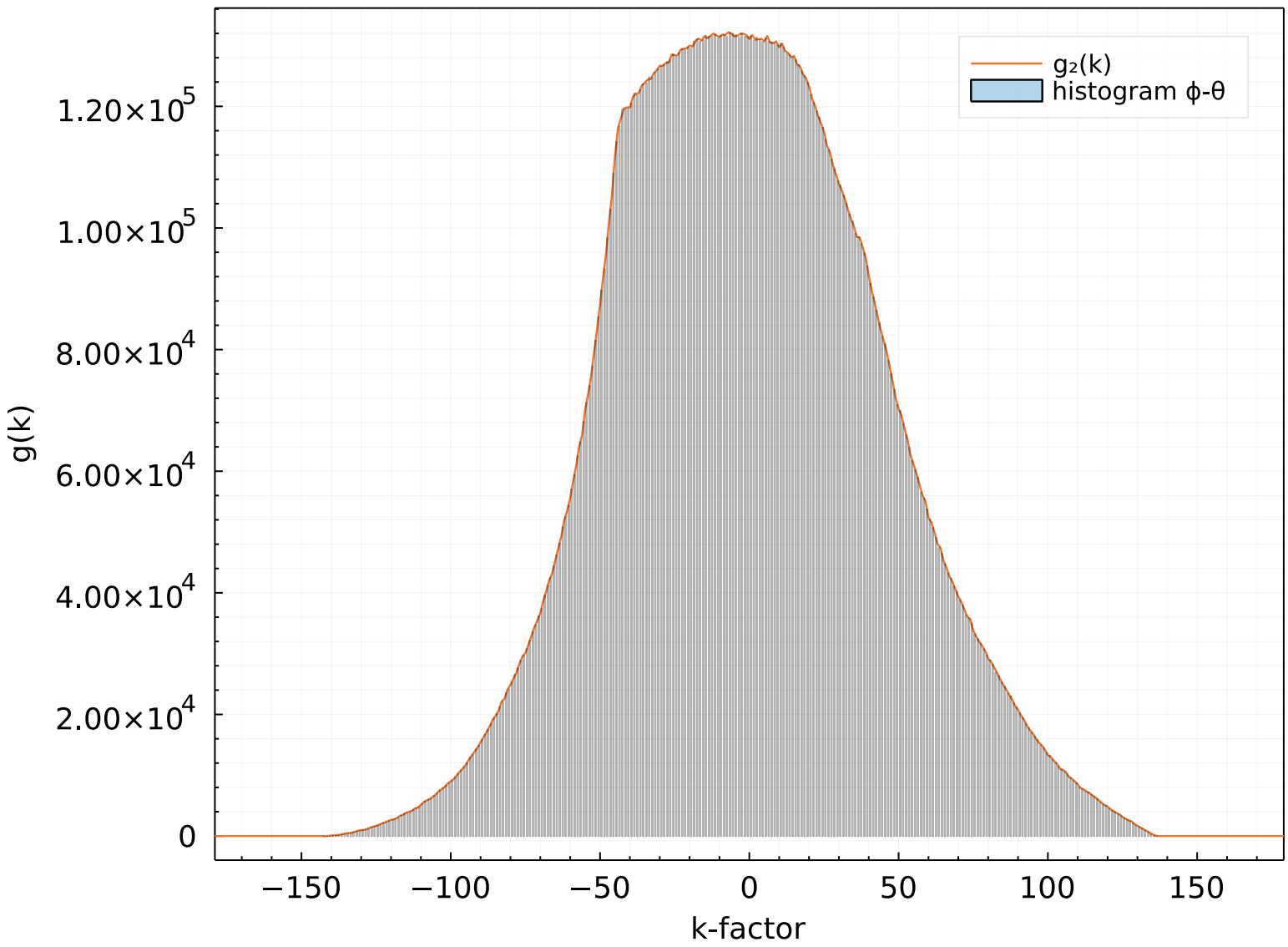
Out[42]:




In [43]:

```
k2 = modTree2.thetaEscaped .- modTree2.thetaEmitted

barhist!(gk2, k2, nbins = Int(180/dϕ*2-1),
        weights = modTree2.weights,
        label = "histogram ϕ-θ",
        alpha = 0.3, bar_width = 0.0)
```

Out[43]:




For comparison let's shift now by `median`.

In [44]:
```
modTree3 = @chain tree begin
```

```
    @select(:thetaEmitted, :thetaEscaped, :weights)
    @rtransform :bin =  get_bin_center(:thetaEscaped, Int(180/dϕ))  # create a vector of bin centers (which bin ϕ falls
    @transform :thetaEscapedOld = :thetaEscaped                     # create a copy of the old ϕ (for comparison only)

    @rtransform :thetaEscaped = :thetaEscapedOld + res_medians[Int(ceil(:bin/dϕ))] # shift ϕ by s: ϕ' = ϕ + s
    @subset( 0 .< :thetaEscaped .< 180) # keep only physical angles
end
```

Out[44]: 15,747,968 rows × 5 columns

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| **1** | 89.6031 | 94.1325 | 1 | 94.5 | 94.6325 |
| **2** | 96.965 | 130.997 | 1 | 139.5 | 139.497 |
| **3** | 154.233 | 129.888 | 1 | 138.5 | 138.388 |
| **4** | 67.3403 | 141.781 | 1 | 154.5 | 154.281 |
| **5** | 11.2133 | 59.5724 | 1 | 53.5 | 53.0724 |
| **6** | 168.417 | 83.5901 | 1 | 81.5 | 81.0901 |
| **7** | 122.632 | 74.1012 | 1 | 69.5 | 69.6012 |
| **8** | 4.96695 | 41.4839 | 1 | 34.5 | 34.9839 |
| **9** | 79.857 | 66.6214 | 1 | 61.5 | 61.1214 |
| **10** | 43.0942 | 41.494 | 1 | 33.5 | 33.994 |
| **11** | 35.1387 | 100.569 | 1 | 104.5 | 104.069 |
| **12** | 150.65 | 99.4232 | 1 | 101.5 | 101.923 |
| **13** | 56.3877 | 62.1631 | 1 | 55.5 | 55.6631 |
| **14** | 128.995 | 94.3403 | 1 | 94.5 | 94.8403 |
| **15** | 63.2692 | 47.4504 | 1 | 40.5 | 40.9504 |
| **16** | 40.4733 | 61.6476 | 1 | 55.5 | 55.1476 |
| **17** | 124.468 | 134.271 | 1 | 142.5 | 142.771 |
| **18** | 128.263 | 30.3424 | 1 | 20.5 | 20.8424 |

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| **19** | 105.706 | 149.173 | 1 | 172.5 | 172.673 |
| **20** | 142.67 | 107.058 | 1 | 112.5 | 112.558 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

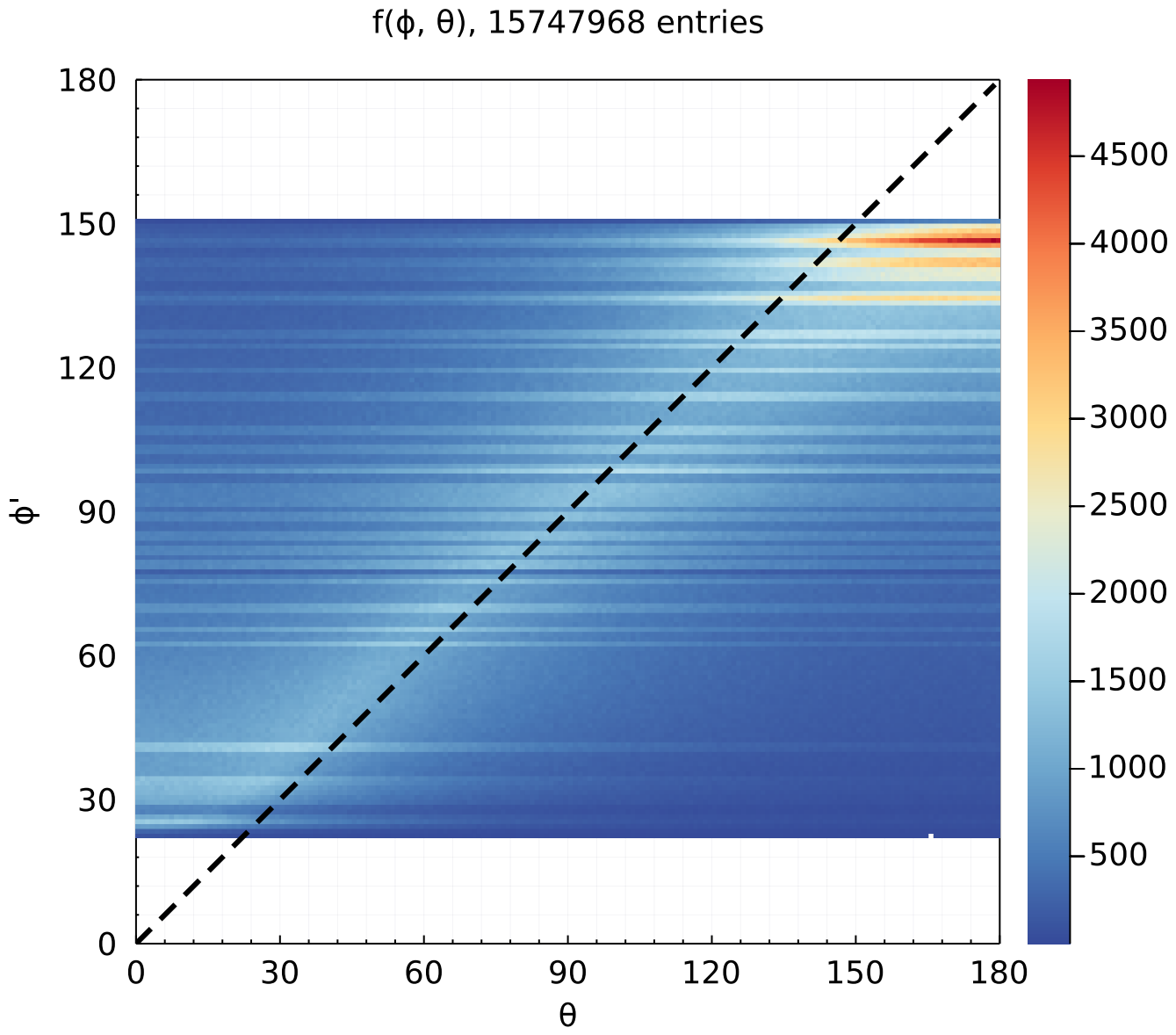
In [45]:

```
dfG3 = get_gs_df(modTree3, dϕ, sign)
matG3 = df_to_mat(dfG3);
```

In [46]:

```
h2d3 = histogram2d(modTree3.thetaEmitted, modTree3.thetaEscaped,
    nbins        = (nBins, nBins),
    weights      = modTree3.weights,
    xlabel       = "θ",
    ylabel       = "ϕ'",
    legend       = :topright,
    title        = string("f(ϕ, θ), ", nrow(modTree3), " entries"),
    lims         = (0, 180),
    aspect_ratio = 1,)
plot!(xPts, xPts, label ="", c= :black, style= :dash, lw =3)
```

Out[46]:

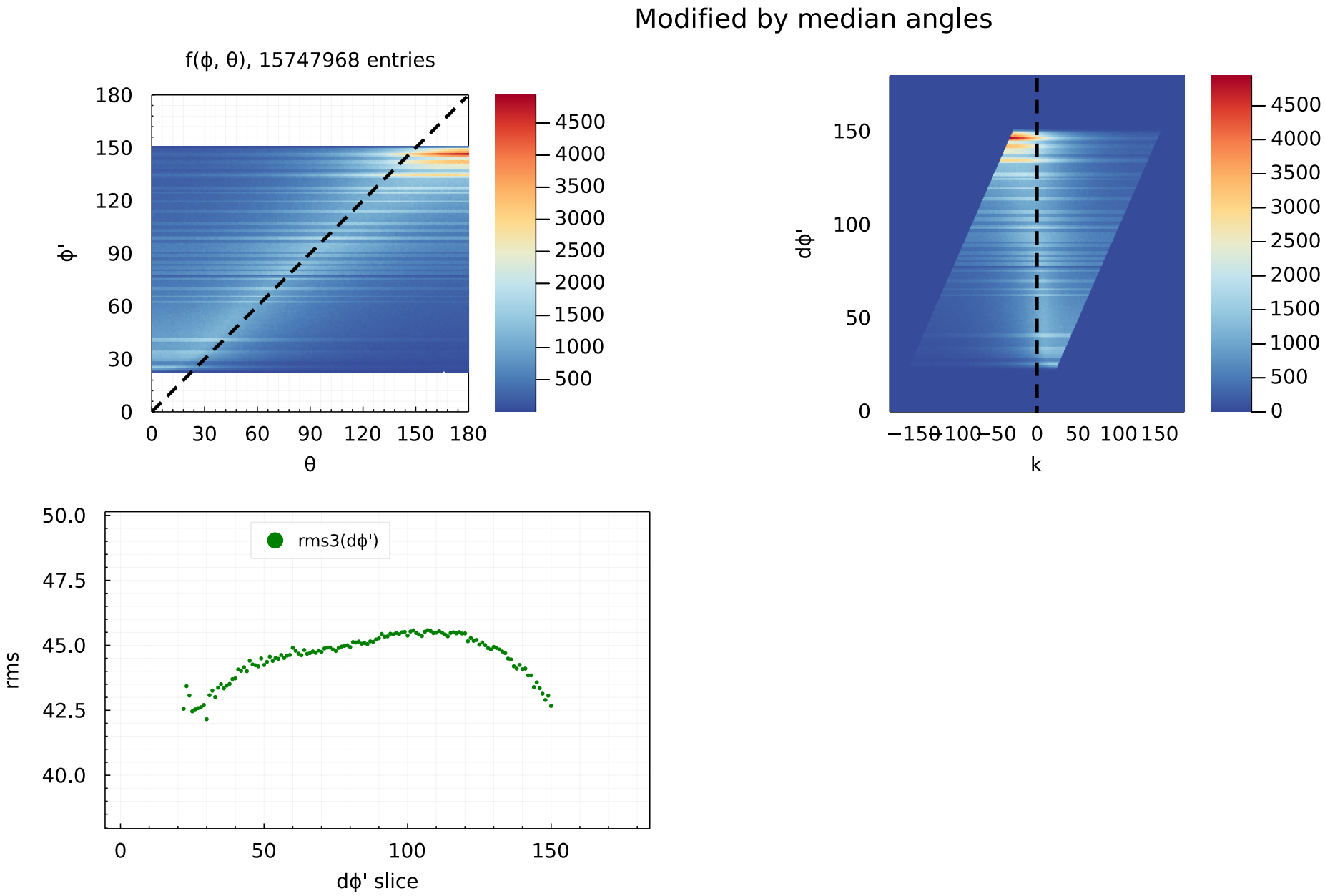



In [47]:

```
hm3 =heatmap(xRange, yRange,  matG3, ylabel ="dϕ'", xlabel ="k" )
vline!([0], label ="", c = :black, lw = 3, s=:dash)

rms3 = [ get_rms(dfG3[:,i], dfG3[:,1]) for i in 2:ncol(dfG3) ]
yMin = 0.9*minimum(filter(x -> x .> 0, rms3)) # get the minimum rms value, excluding 0
```

```
yMax = 1.1*maximum(filter(x -> x .> 0, rms3))

sct3 = scatter( xPts, rms3, label ="rms3(dϕ')", ms=2,
                legend=:top, xlabel ="dϕ' slice", c =:green,
                ylabel ="rms", ylims = (yMin, yMax) )


plot(h2d3, hm3, sct3, layout = @layout[a b; c _] , size = (1200, 800), plot_title= "Modified by median angles")
```

Out[47]:

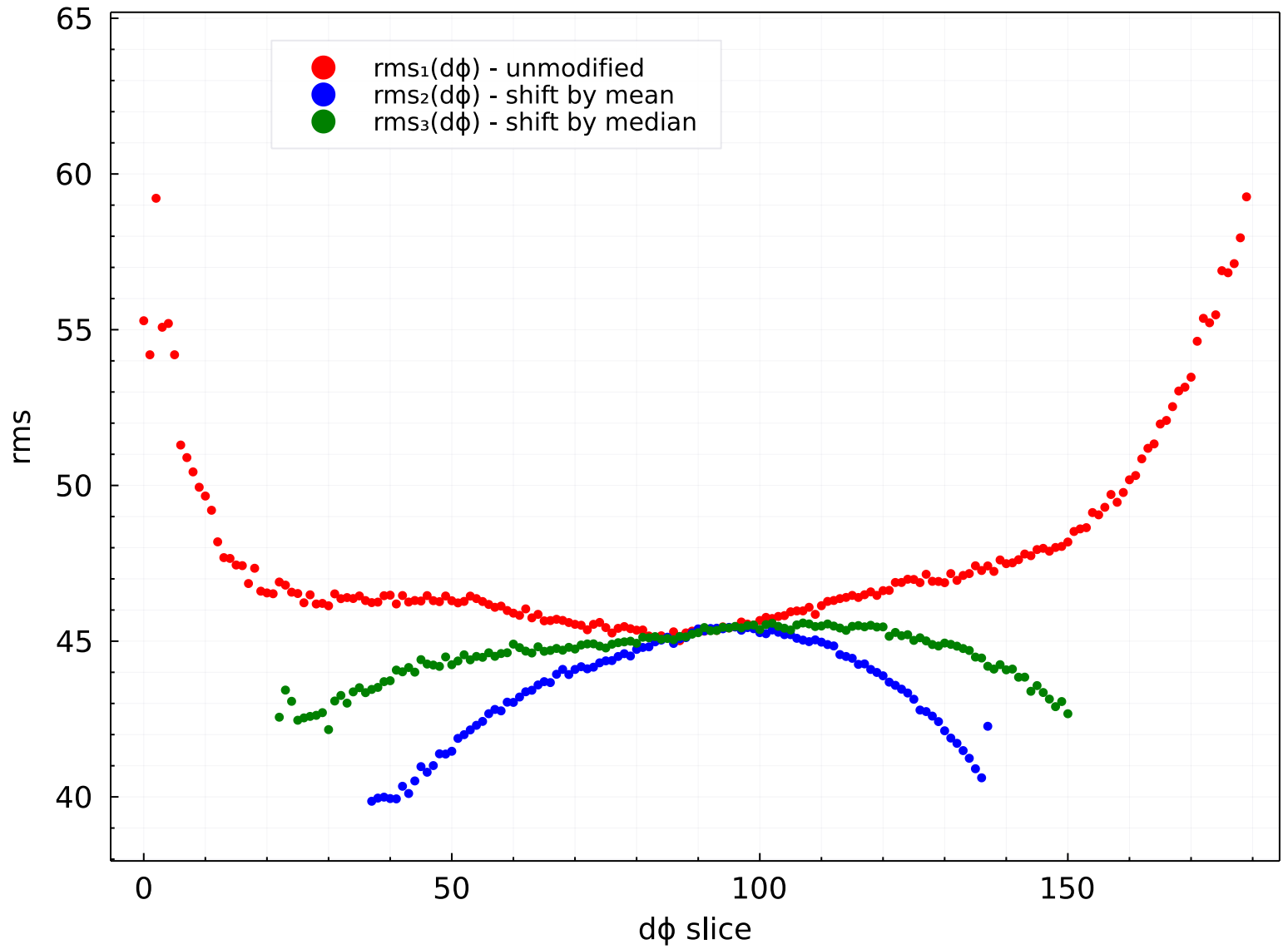



In [48]:

```
scatter( [xPts xPts xPts], [rms1 rms2 rms3],
    label =["rms₁(dϕ) - unmodified" "rms₂(dϕ) - shift by mean" "rms₃(dϕ) - shift by median"],
```

```
        ms=3, c = [:red :blue :green],
        legend=:top, xlabel ="dϕ slice", ylabel ="rms", ylims = (yMin, 1.1*maximum(rms1)))
```

Out[48]:




Lastly, shift by `mode`.

In [49]:

```
modTree4 = @chain tree begin
    @select(:thetaEmitted, :thetaEscaped, :weights)
    @rtransform :bin =  get_bin_center(:thetaEscaped, Int(180/dϕ))  # create a vector of bin centers (which bin ϕ falls
    @transform :thetaEscapedOld = :thetaEscaped                     # create a copy of the old ϕ (for comparison only)

    @rtransform :thetaEscaped = :thetaEscapedOld + res_modes[Int(ceil(:bin/dϕ))] # shift ϕ by s: ϕ' = ϕ + s
    @subset( 0 .< :thetaEscaped .< 180) # keep only physical angles
end
```

Out[49]: 15,549,016 rows × 5 columns

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| **1** | 89.6031 | 97.1325 | 1 | 94.5 | 94.6325 |
| **2** | 96.965 | 163.997 | 1 | 139.5 | 139.497 |
| **3** | 154.233 | 149.888 | 1 | 138.5 | 138.388 |
| **4** | 67.3403 | 168.781 | 1 | 154.5 | 154.281 |
| **5** | 11.2133 | 55.5724 | 1 | 53.5 | 53.0724 |
| **6** | 168.417 | 82.5901 | 1 | 81.5 | 81.0901 |
| **7** | 122.632 | 69.1012 | 1 | 69.5 | 69.6012 |
| **8** | 4.96695 | 34.4839 | 1 | 34.5 | 34.9839 |
| **9** | 79.857 | 59.6214 | 1 | 61.5 | 61.1214 |
| **10** | 43.0942 | 31.494 | 1 | 33.5 | 33.994 |
| **11** | 35.1387 | 109.569 | 1 | 104.5 | 104.069 |
| **12** | 150.65 | 106.423 | 1 | 101.5 | 101.923 |
| **13** | 56.3877 | 54.1631 | 1 | 55.5 | 55.6631 |
| **14** | 128.995 | 97.3403 | 1 | 94.5 | 94.8403 |
| **15** | 63.2692 | 42.4504 | 1 | 40.5 | 40.9504 |
| **16** | 40.4733 | 53.6476 | 1 | 55.5 | 55.1476 |
| **17** | 124.468 | 169.271 | 1 | 142.5 | 142.771 |

| | thetaEmitted | thetaEscaped | weights | bin | thetaEscapedOld |
|---|---|---|---|---|---|
| | Float64 | Float64 | Int64 | Float64 | Float64 |
| **18** | 128.263 | 21.3424 | 1 | 20.5 | 20.8424 |
| **19** | 142.67 | 120.058 | 1 | 112.5 | 112.558 |
| **20** | 68.2757 | 83.1263 | 1 | 81.5 | 81.6263 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

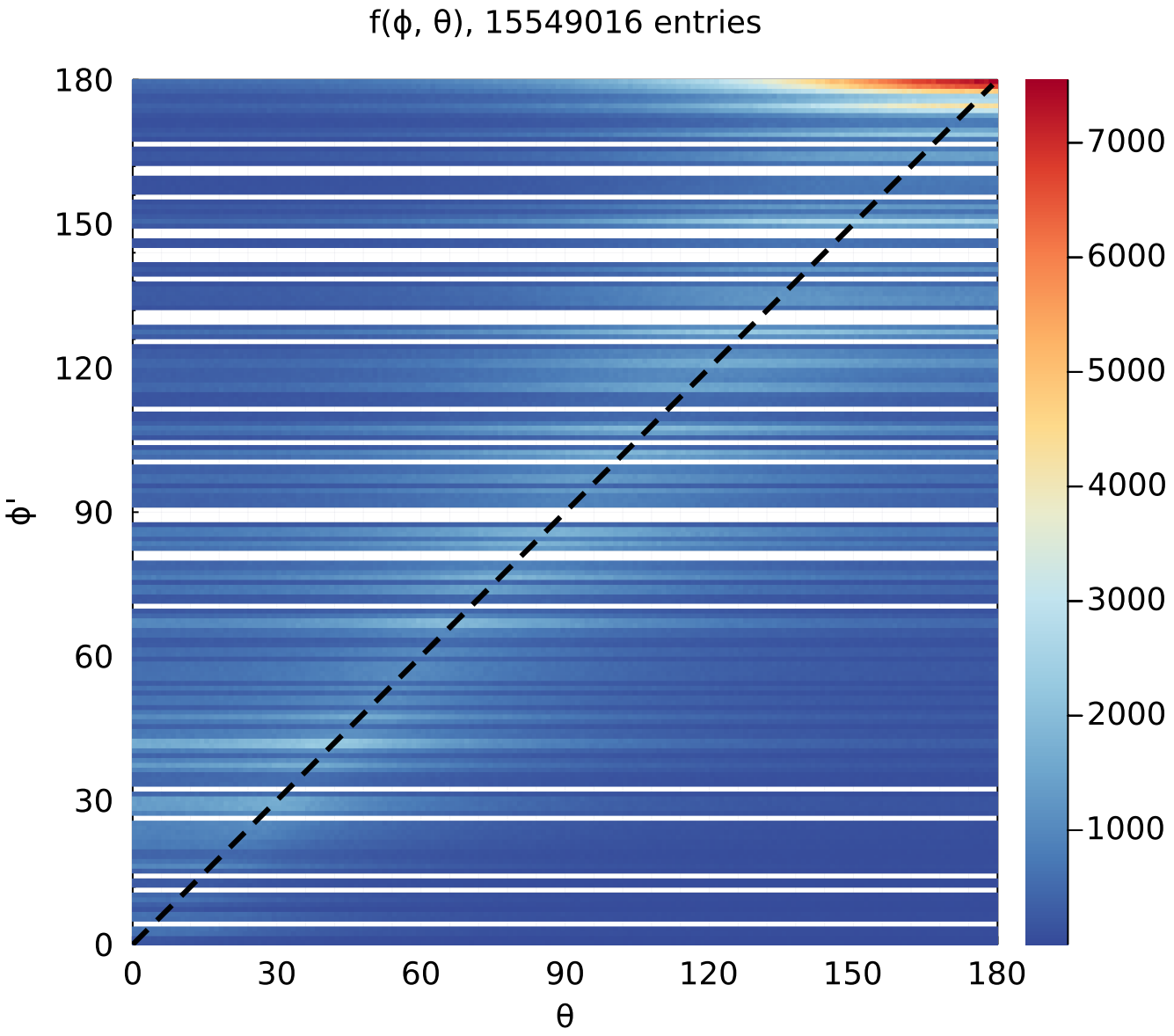
In [50]:

```
dfG4 = get_gs_df(modTree4, dϕ, sign)
matG4 = df_to_mat(dfG4);
```

In [51]:

```
h2d4 = histogram2d(modTree4.thetaEmitted, modTree4.thetaEscaped,
    nbins        = (nBins, nBins),
    weights      = modTree4.weights,
    xlabel       = "θ",
    ylabel       = "ϕ'",
    legend       = :topright,
    title        = string("f(ϕ, θ), ", nrow(modTree4), " entries"),
    lims         = (0, 180),
    aspect_ratio = 1,)
plot!(xPts, xPts, label ="", c= :black, style= :dash, lw =3)
```

Out[51]:



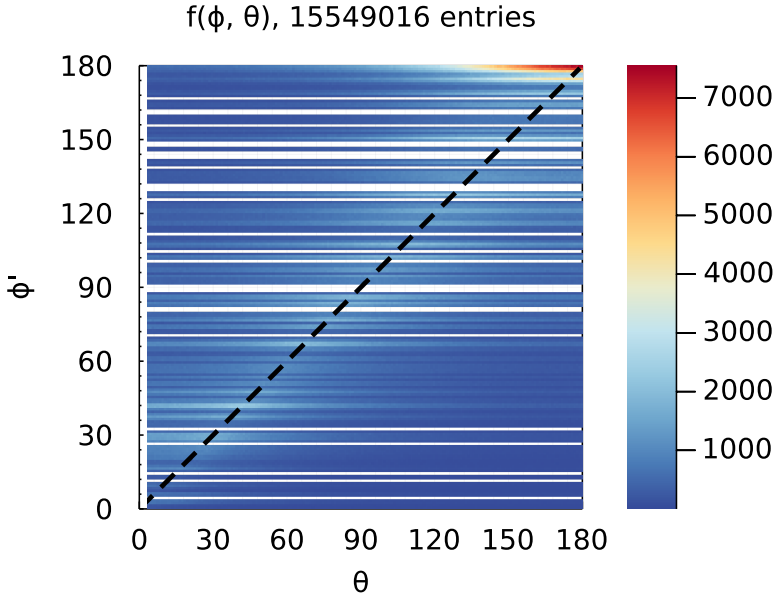

In [77]:

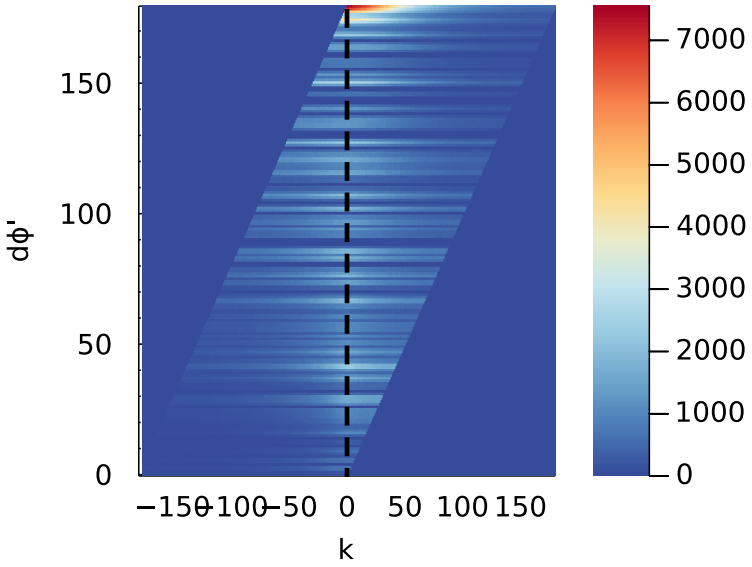
```
hm4 =heatmap(xRange, yRange,  matG4, ylabel ="dϕ'", xlabel ="k" )
vline!([0], label ="", c = :black, lw = 3, s=:dash)

rms4 = [ get_rms(dfG4[:,i], dfG4[:,1]) for i in 2:ncol(dfG4) ]
yMin4 = 0.9*minimum(filter(x -> x .> 0, rms4)) # get the minimum rms value, excluding 0
```

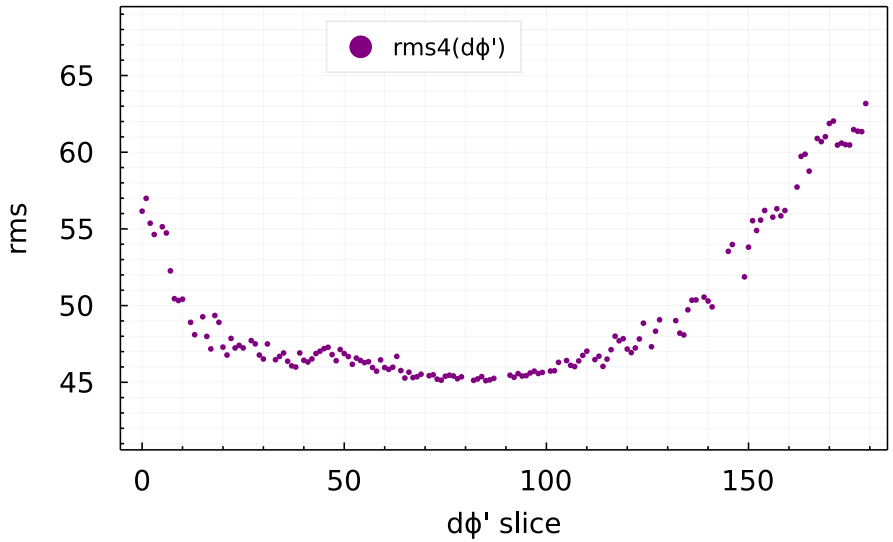
```
yMax4 = 1.1*maximum(filter(x -> x .> 0, rms4))

sct4 = scatter( xPts, rms4, label ="rms4(dϕ')", ms=2,
                legend=:top, xlabel ="dϕ' slice", c =:purple,
                ylabel ="rms", ylims = (yMin4, yMax4) )


plot(h2d4, hm4, sct4, layout = @layout[a b; c _] , size = (1200, 800), plot_title= "Modified by modes angles")
```

Out[77]:

Modified by modes angles



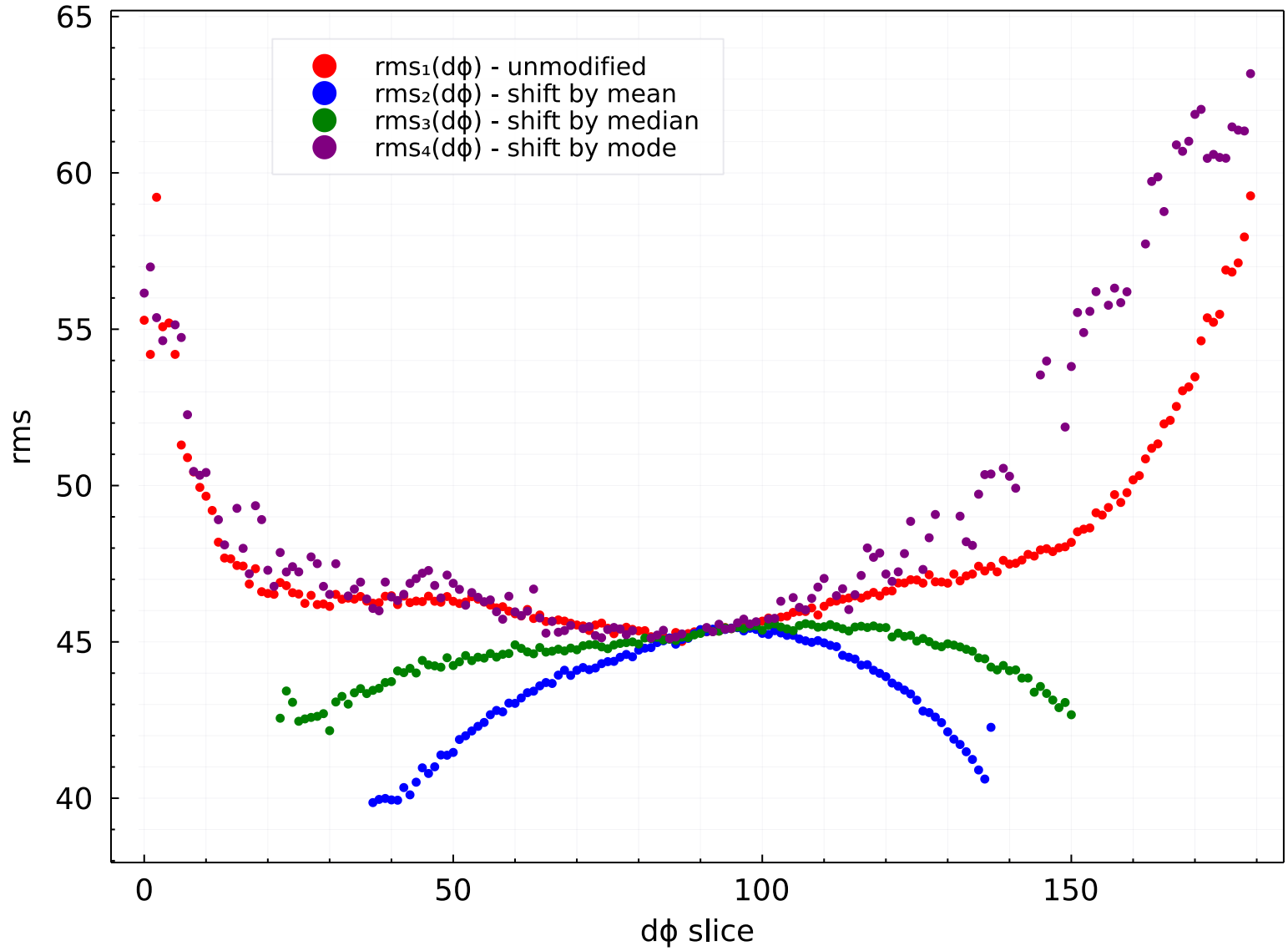

Finally, all shifts compared by RMS in one graph. It appears that shifting

# by `mean` value of each row produces the best result!

In [62]:

```
scatter( [xPts xPts xPts xPts], [rms1 rms2 rms3 rms4],
    label =["rms₁(dϕ) - unmodified" "rms₂(dϕ) - shift by mean" "rms₃(dϕ) - shift by median" "rms₄(dϕ) - shift by mode"]
        ms=3, c = [:red :blue :green :purple],
        legend=:top, xlabel ="dϕ slice", ylabel ="rms", ylims = (yMin, 1.1*maximum(rms1)))
```

Out[62]:

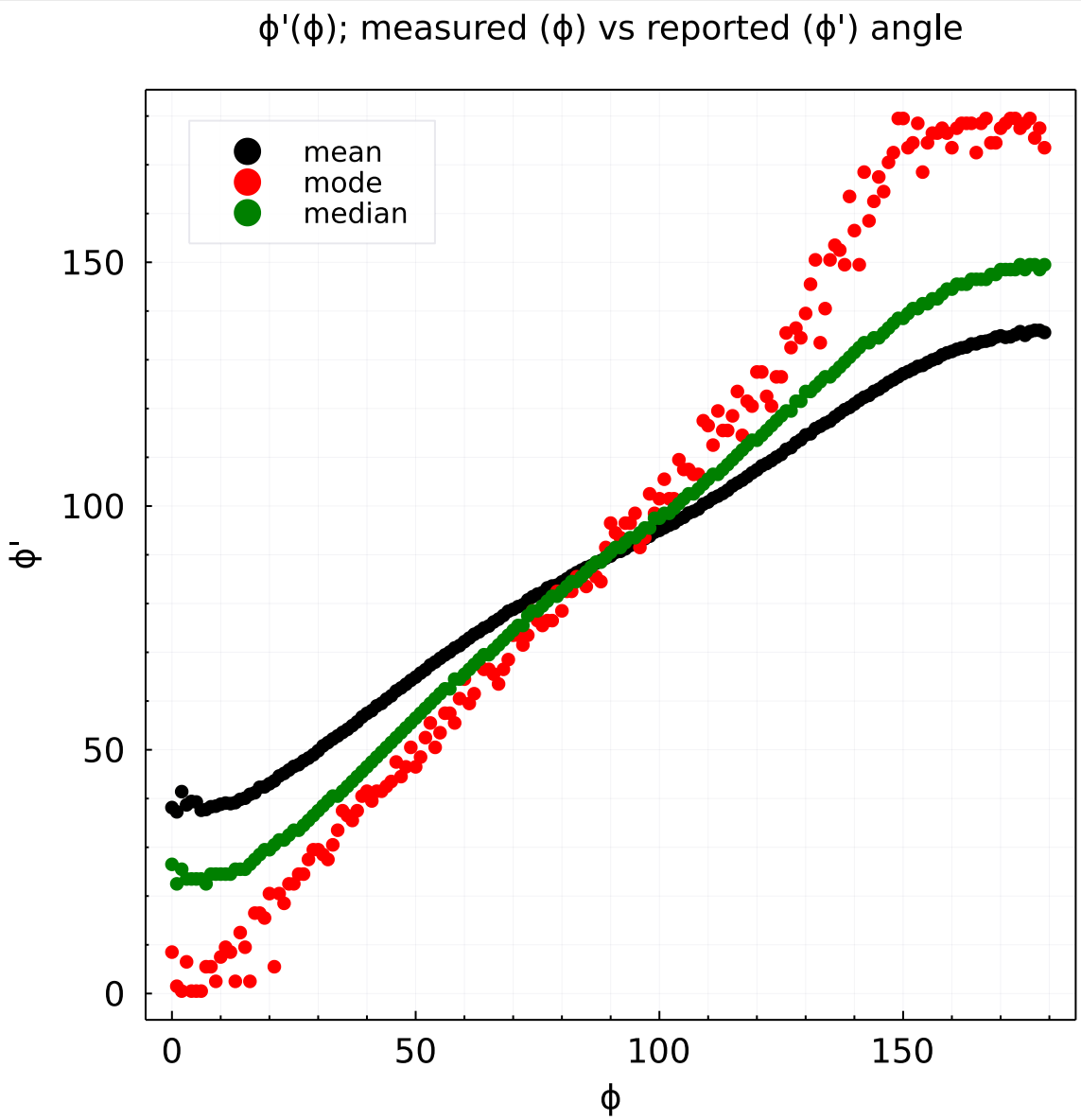



In [61]:

```
scatter(
    xPts, get_phi_prime.(xPts, res_means), lims = (0,180), ms = 4, c= :black,
    label ="mean", xlabel = "ϕ", ylabel ="ϕ'", aspect_ratio = 1,
    legend =:topleft, title = "ϕ'(ϕ); measured (ϕ) vs reported (ϕ') angle", widen =:true
)
```

```
scatter!(xPts, get_phi_prime.(xPts, res_modes), lims = (0,180), ms = 4, c= :red, label ="mode" )
scatter!(xPts, get_phi_prime.(xPts, res_medians), lims = (0,180), ms = 4, c= :green, label ="median" )
```

Out[61]:

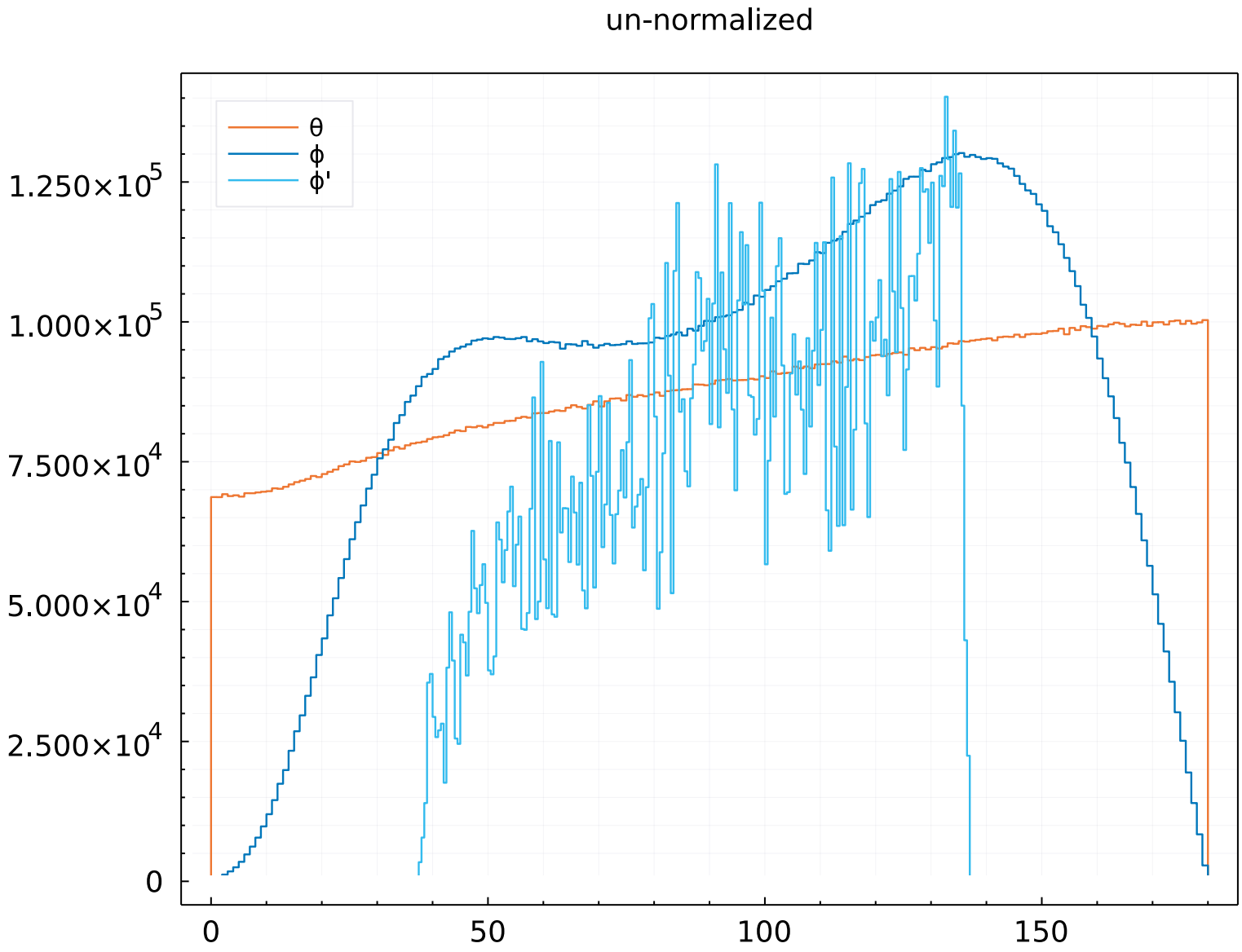



In [55]:

```
histogram(tree.thetaEmitted, weights = tree.weights,label = "θ", legend =:topleft, title= "un-normalized")
```

```
histogram!(tree.thetaEscaped, weights = tree.weights, label = "ϕ")
histogram!(modTree2.thetaEscaped, weights = modTree2.weights, label = "ϕ'")
```

Out[55]:

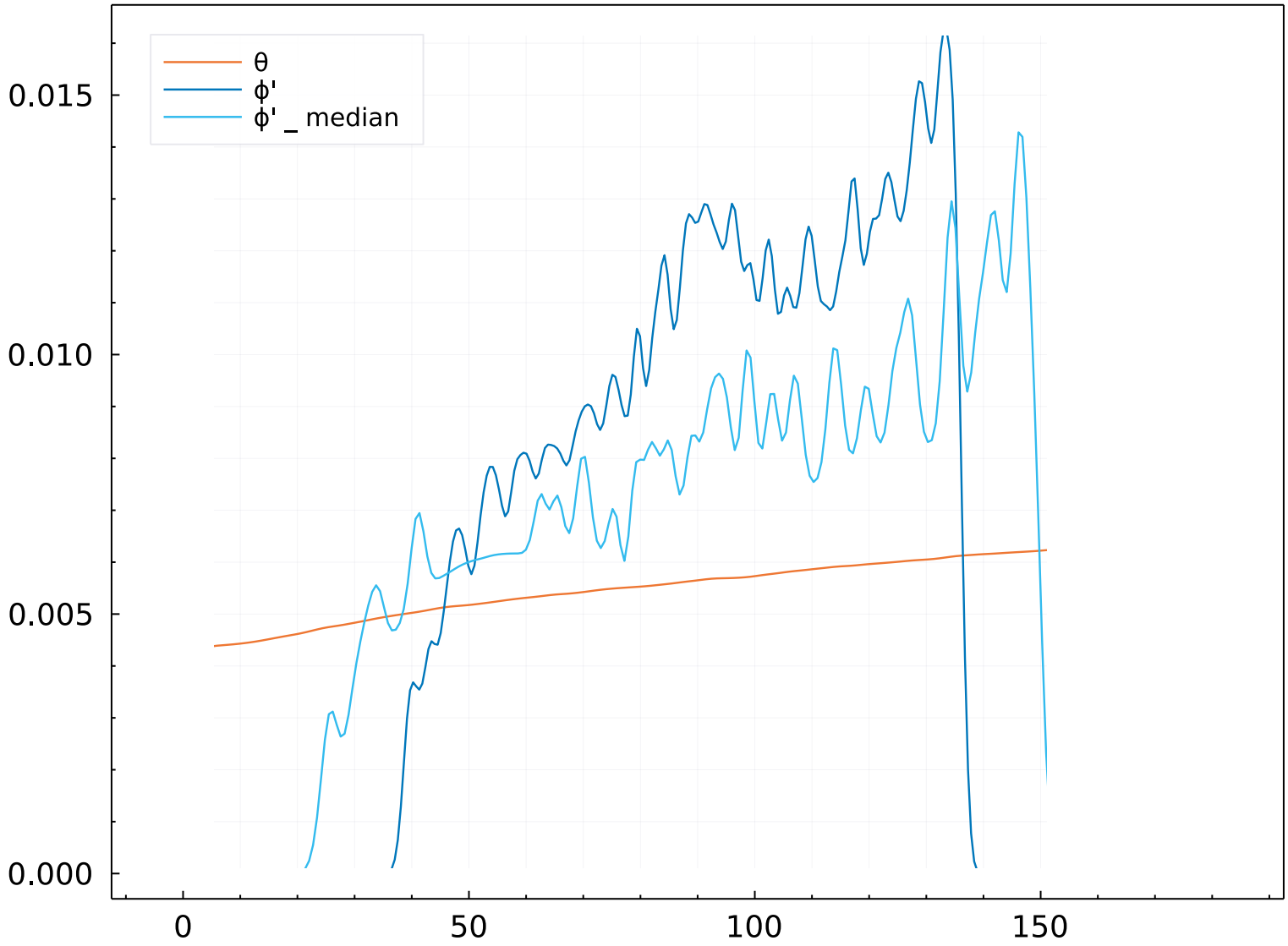



In [60]:

```
StatsPlots.density(tree.thetaEmitted, weights = tree.weights, label = "θ", legend =:topleft)
# StatsPlots.density!(tree.thetaEscaped, weights = tree.weights, label = "ϕ")
```

```
StatsPlots.density!(modTree2.thetaEscaped,weights = modTree2.weights, label = "ϕ'")
StatsPlots.density!(modTree3.thetaEscaped,weights = modTree3.weights, label = "ϕ' _ median")
# StatsPlots.density!(modTree4.thetaEscaped,weights = modTree4.weights, label = "ϕ' _ modus ")
```

Out[60]:




In [ ]:

In [ ]:

```
```

In [57]:

```
# function norm_bin(w, dϕ, n) # normalize each bin by weight/(n*dϕ^2)
#     if n == 0.0
#         return 0.0
#     else
#         return w/(dϕ^2*n)
#     end
# end
```

In [58]:

```
# for i in eachindex(h2w2.hist.weights)
#     h2w2.hist.weights[i] = norm_bin(h2w2.hist.weights[i], dϕ, bincounts(h22)[i])
# end
```