Predicting performance in Psychometric Tests

Load packages and dataset

```
begin
    using DataFrames
    using Empirikos
    using Plots
    using PGFPlotsX
    using LaTeXStrings
    using MosekTools
    using JuMP
end
```

```
begin
    pgfplotsx()
    empty!(PGFPlotsX.CUSTOM_PREAMBLE)
    push!(PGFPlotsX.CUSTOM_PREAMBLE, raw"\usepackage{amssymb}")
    push!(PGFPlotsX.CUSTOM_PREAMBLE, raw"\newcommand{\PP}[2][]
{\mathbb{P}_{#1}\left[#2\right]}")
    push!(PGFPlotsX.CUSTOM_PREAMBLE, raw"\newcommand{\EE}[2][]
{\mathbb{E}_{#1}\left[#2\right]}")
end;
```

lord_cressie =

	x	N1	N2	post_mean	Lower1	Upper1	Lower2	Upper2
1	20	63	2	0.898	0.852	0.952	0.713	0.999
2	19	141	2	0.863	missing	missing	missing	missing
3	18	220	2	0.823	0.78	0.846	0.637	0.936
4	17	319	2	0.773	missing	missing	missing	missing
5	16	424	6	0.716	0.69	0.743	0.588	0.846
6	15	622	4	0.663	missing	missing	missing	missing
7	14	776	8	0.619	0.605	0.646	0.521	0.739
8	13	1001	4	0.583	missing	missing	missing	missing
9	12	1203	9	0.553	0.534	0.564	0.447	0.627
10	11	1443	19	0.526	missing	missing	missing	missing
	more							
21	0	2	0	0.28	0.01	0.373	0.0	0.479

```
lord_cressie = Empirikos.LordCressie.load_table() |> DataFrame
```

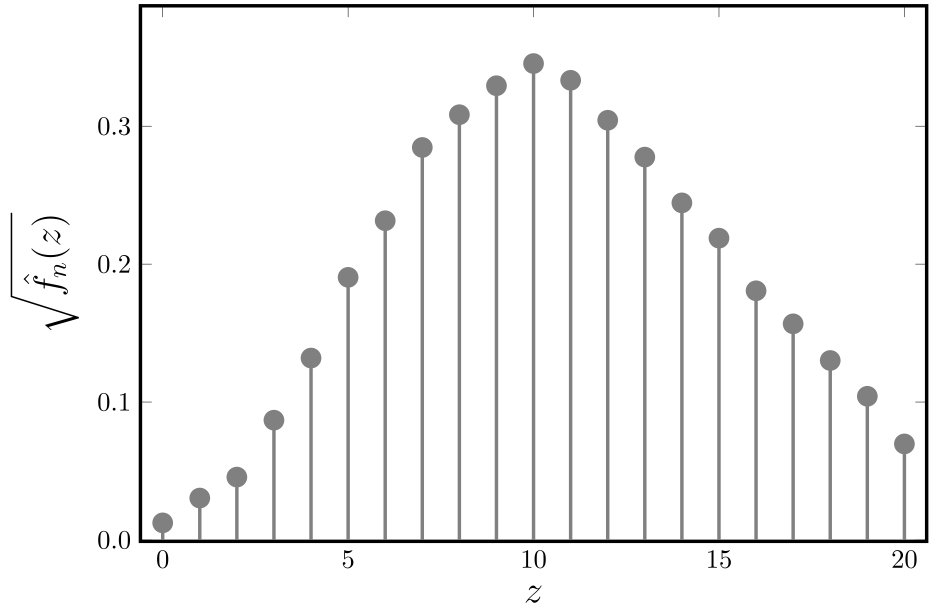
```
Zs = Empirikos.MultinomialSummary(BinomialSample.(lord_cressie.x, 20),
                                  lord_cressie.N1);

```

Plot empirical frequencies (Fig. 1a)

empirical_probs =
[0.000153965, 0.000923788, 0.00207852, 0.00754426, 0.017398, 0.0362587, 0.0535797, 0.08098

```
empirical_probs = weights(Zs)/nobs(Zs)
```



lord_cressie_freq =

```
lord_cressie_freq = plot(
     0:20,
     sqrt.(empirical_probs), seriestype=:sticks, frame=:box,
     grid=nothing, color=:grey, markershape=:circle,
     legendfonthalign = :left,
     markerstrokealpha = 0, ylim=(-0.001,sqrt(0.15)),
     xguide=L"z",yguide=L"\sqrt{\hat{f}_n(z)}",thickness_scaling=1.3,
     label=nothing, size=(500,350)
    )
```

```
# savefig(lord_cressie_freq, "lord_cressie_pdf.tikz")
```

Construct confidence intervals

We first create a discretization object that will combine all counts ≤ 1.

discr = Discretizer([(.. 1], 2, 3, 4, 5, 6, 7, 8, 9, more ,[20 ..)])

```
discr = integer_discretizer(1:20)
```

Zs_collapse =

MultinomialSummary(SortedDict(Z ∈ (.. 1] | n=20 ⇒ 14, Z=2 | n=20 ⇒ 27, Z=3 | n=20 ⇒

```
Zs_collapse = discr(Zs)
```

quiet_mosek =

OptimizerWithAttributes(Optimizer (generic function with 2 methods), [RawParameter("QUIET

```
quiet_mosek = optimizer_with_attributes(Mosek.Optimizer, "QUIET" => true)

```

```
𝒢 = DiscretePriorClass(range(0.0,stop=1.0,length=300));
```

```
postmean_targets = Empirikos.PosteriorMean.(BinomialSample.(0:20,20));
```

$\chi^2 - F$-localization intervals

```
chisq_floc =  ChiSquaredFLocalization(0.05)
```

```
chisq_floc = Empirikos.ChiSquaredFLocalization(0.05)
```

```
floc_method_chisq =
EB intervals with F-Localization: ChiSquaredFLocalization{Float64}(0.05)
                  𝒢: DiscretePriorClass | support = 0.0:0.0033444816053511705:1.0
```

```
floc_method_chisq = FLocalizationInterval(flocalization = chisq_floc,
                                          convexclass= 𝒢, solver=quiet_mosek)
```

```
chisq_cis =
 [lower = 0.003731, upper = 0.365, α = 0.05  (PosteriorMean{BinomialSample{Int64, Int64}}(
```

```
chisq_cis = confint.(floc_method_chisq, postmean_targets, Zs_collapse)
```

```
lower_chisq_ci =
 [0.00373087, 0.106113, 0.235178, 0.305643, 0.340864, 0.364113, 0.387145, 0.410672, 0.43582
```

```
lower_chisq_ci = getproperty.(chisq_cis, :lower)
```

```
upper_chisq_ci =
 [0.364997, 0.391926, 0.393431, 0.39507, 0.398937, 0.406593, 0.419615, 0.439773, 0.463009,
```

```
upper_chisq_ci = getproperty.(chisq_cis, :upper)
```

	x	N1	N2	post_mean	Lower1	Upper1	Lower2	Upper2	lower_chisq	u
1	20	63	2	0.898	0.852	0.952	0.713	0.999	0.852604	0
2	19	141	2	0.863	missing	missing	missing	missing	0.813444	0
3	18	220	2	0.823	0.78	0.846	0.637	0.936	0.780462	0
4	17	319	2	0.773	missing	missing	missing	missing	0.73414	0
5	16	424	6	0.716	0.69	0.743	0.588	0.846	0.689779	0
6	15	622	4	0.663	missing	missing	missing	missing	0.647067	0
7	14	776	8	0.619	0.605	0.646	0.521	0.739	0.60518	0
8	13	1001	4	0.583	missing	missing	missing	missing	0.565566	0
9	12	1203	9	0.553	0.534	0.564	0.447	0.627	0.533905	0
10	11	1443	19	0.526	missing	missing	missing	missing	0.507679	0
	more									
21	0	2	0	0.28	0.01	0.373	0.0	0.479	0.00373087	0

```
begin
    lord_cressie.lower_chisq = reverse(lower_chisq_ci)
    lord_cressie.upper_chisq = reverse(upper_chisq_ci)
    lord_cressie
end
```

DKW F-localization intervals

```
floc_method_dkw =
EB intervals with F-Localization: DvoretzkyKieferWolfowitz{Float64, Int64}(0.05, 1000)
                  𝒢: DiscretePriorClass | support = 0.0:0.0033444816053511705:1.0
```

```
floc_method_dkw = FLocalizationInterval(
                            flocalization = DvoretzkyKieferWolfowitz(0.05),
                            convexclass = 𝒢, solver=quiet_mosek)

```

```
dkw_cis = confint.(floc_method_dkw, postmean_targets, Zs_collapse);

```

```
lower_dkw_ci =
 [0.000191288, 0.026525, 0.114512, 0.221245, 0.299652, 0.344311, 0.375797, 0.40453, 0.42863
```

```
lower_dkw_ci = getproperty.(dkw_cis, :lower)
```

```
upper_dkw_ci =
 [0.418168, 0.440815, 0.440832, 0.440867, 0.441617, 0.443737, 0.449232, 0.45908, 0.4773, 0.
```

```
upper_dkw_ci = getproperty.(dkw_cis, :upper)
```

Amari intervals

amari_chisq =
```
AMARI with  F-Localization: ChiSquaredFLocalization{Float64}(0.01)
            𝒢: DiscretePriorClass | support = 0.0:0.0033444816053511705:1.0
```

```
amari_chisq = Empirikos.AMARI(
                                convexclass = 𝒢,
                                flocalization = Empirikos.ChiSquaredFLocalization(0.01),
                                solver=quiet_mosek,
                                discretizer=discr
                                )

```

postmean_ci_amari =
```
[lower = 0.003125, upper = 0.3717, α = 0.05  (PosteriorMean{BinomialSample{Int64, Int64}}
```

```
postmean_ci_amari = confint.(amari_chisq, postmean_targets, Zs_collapse)
```

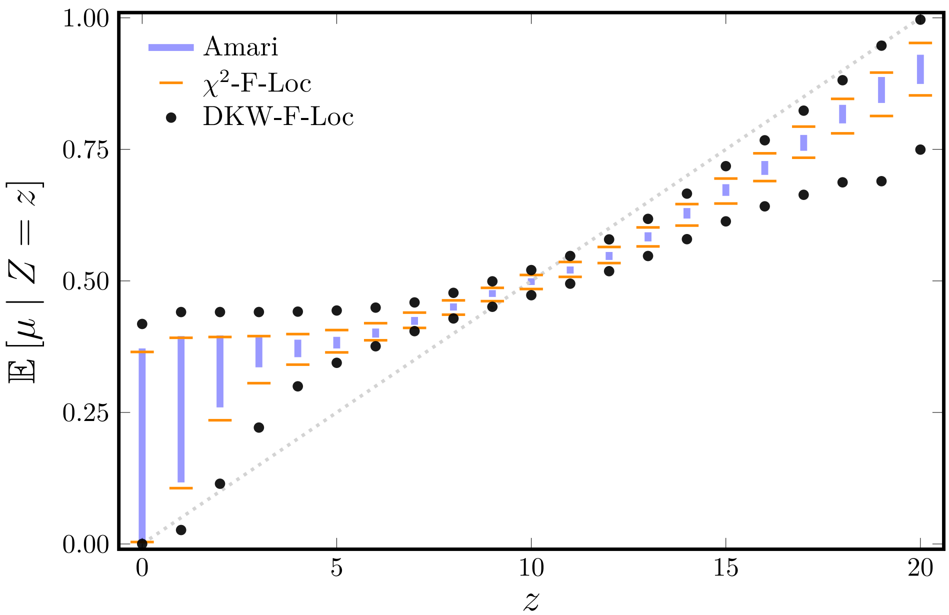
lower_amari_ci =
```
[0.00312456, 0.117026, 0.259604, 0.335781, 0.355055, 0.37159, 0.392273, 0.416569, 0.443707
```

```
lower_amari_ci = getproperty.(postmean_ci_amari, :lower)
```

upper_amari_ci =
```
[0.371684, 0.395092, 0.396436, 0.392493, 0.388285, 0.393683, 0.409529, 0.431459, 0.457264,
```

```
upper_amari_ci = getproperty.(postmean_ci_amari, :upper)
```

Plot confidence intervals (Fig.2b)



postmean_plot =

```
postmean_plot = begin
plot(0:20, upper_amari_ci, fillrange=lower_amari_ci ,seriestype=:sticks,
            frame=:box,
            grid=nothing,
            xguide = L"z",
            yguide = L"\EE{\mu \mid Z=z}",
            legend = :topleft,
            linewidth=2,
            linecolor=:blue,
            alpha = 0.4,
            background_color_legend = :transparent,
            legendfonthalign = :left,
            foreground_color_legend = :transparent, ylim=(-0.01,1.01),
thickness_scaling=1.3,
            label="Amari",
            size=(500,350))

plot!(0:20, [lower_chisq_ci upper_chisq_ci], seriestype=:scatter,  markershape=:hline,
            label=[L"\chi^2\textrm{-F-Loc}" nothing], markerstrokecolor= :darkorange,
markersize=4.5)

plot!(0:20, [lower_dkw_ci upper_dkw_ci], seriestype=:scatter,  markershape=:circle,
             label=["DKW-F-Loc" nothing], color=:black, alpha=0.9, markersize=2.0,
markerstrokealpha=0)

plot!([0;20], [0.0; 1.0], seriestype=:line, linestyle=:dot, label=nothing,
color=:lightgrey)
end
```

```
#savefig(postmean_plot, "lord_cressie_posterior_mean.tikz")
```