Supplemental material

Daniele Zago and Giovanna Capizzi

2022-09-28

1 In-depth analysis

In this section we perform an in-depth analysis of the performance of the CLM and C&M2020 methods when changing the control chart design parameter. Herein, the OC performance of the proposed cautious learning parameter update rules are compared to the fixed and adaptive estimators.

1.1 Simulation settings

We use the same simulation settings as in Section ?? of the paper regarding the intial sample size, cautious learning parameters, parameter shift magnitude and location, and control limit design. The IC process parameter is fixed to $\theta = 4$ and we study the performance of the proposed methodology as λ varies in $\{0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2\}$.

1.2 Results

From the results in Figures 1, 3, 5, 7, 9, 11 and 13, it is possible to see that the parameter estimator based on time-delayed update rules provide consistently improved results when compared to both adaptive and fixed estimators. This appears to be valid across both shift sizes and control chart design parameter, with differences between the *CLM and the C&M2020 approaches.*

Overall, the suggested settings of $A = 1.5$ and $B = 50$ for the *C&M2020* update rule results in a more conservative update rule than the CLM approach. Indeed, when the control chart displays a high sensitivity to small shifts such as in Figure 1, the CLM approach provides a protection against early shifts that is comparable to the *C&M2020* update rule. However, in these cases the CLM approach provides a better protection to shifts that happen later in the monitoring phase due to the higher precision of the estimates.

When the control chart is less sensitive to small parameter shifts (Figure 13), it is also less apt at preventing small shifts from biasing the estimates. In these cases the conservativeness of the *C&M2020* approach allows the prevention of early and small parameter shifts from biasing the estimates.

Overall, among the proposed delayed-parameter update rules we do not observe a clear winner. Both the CLM and C&M2020 proposals seem to be dependent on the sensitivity of the control chart to different magnitudes of parameter shift. In any case, the proposed

delayed estimators seem to provide a better overall performance than the fully adaptive estimator.

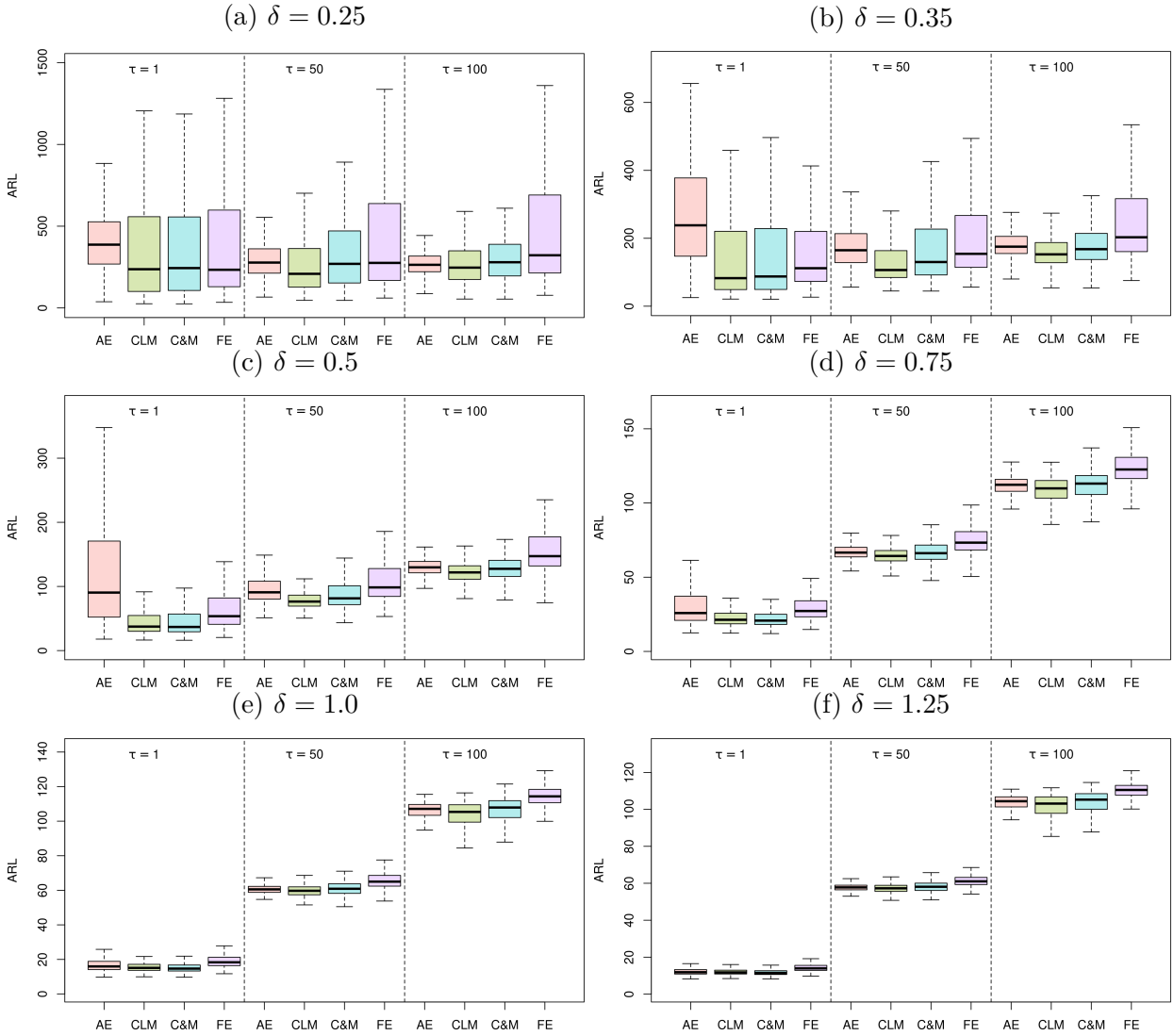


Figure 1: OC performance of the EWMA ($\lambda = 0.05$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

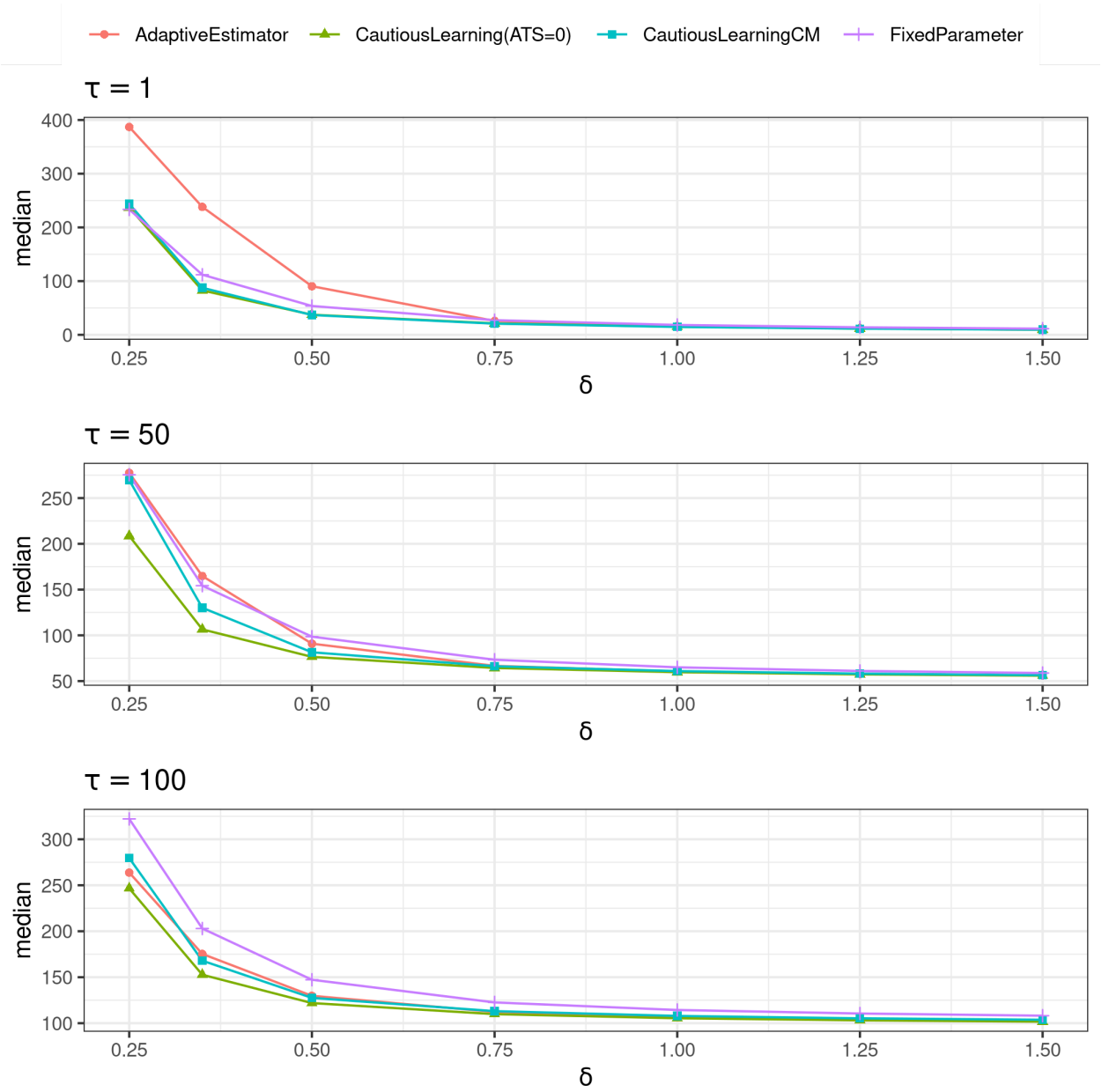


Figure 2: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.05$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

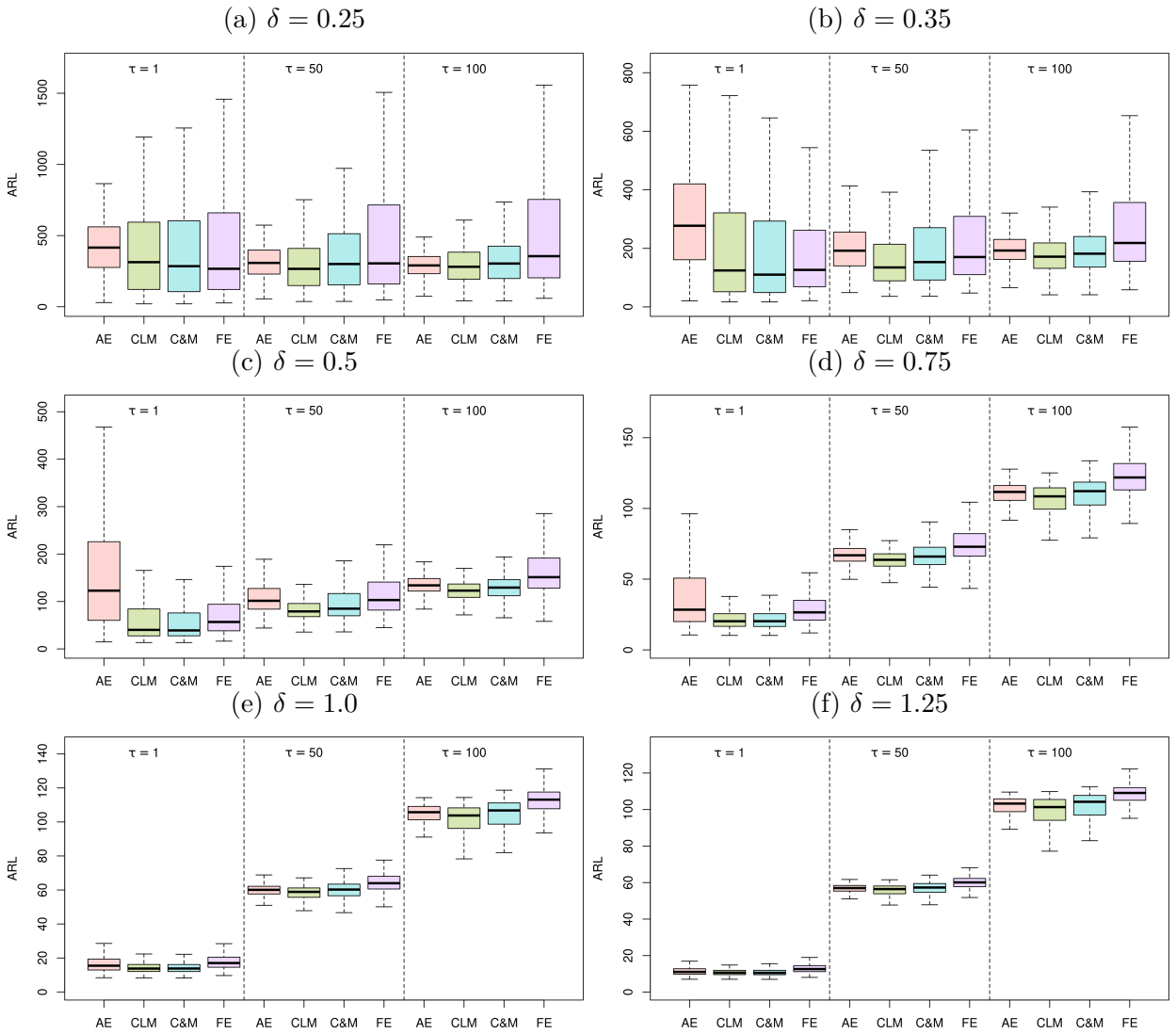


Figure 3: OC performance of the EWMA ($\lambda = 0.075$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

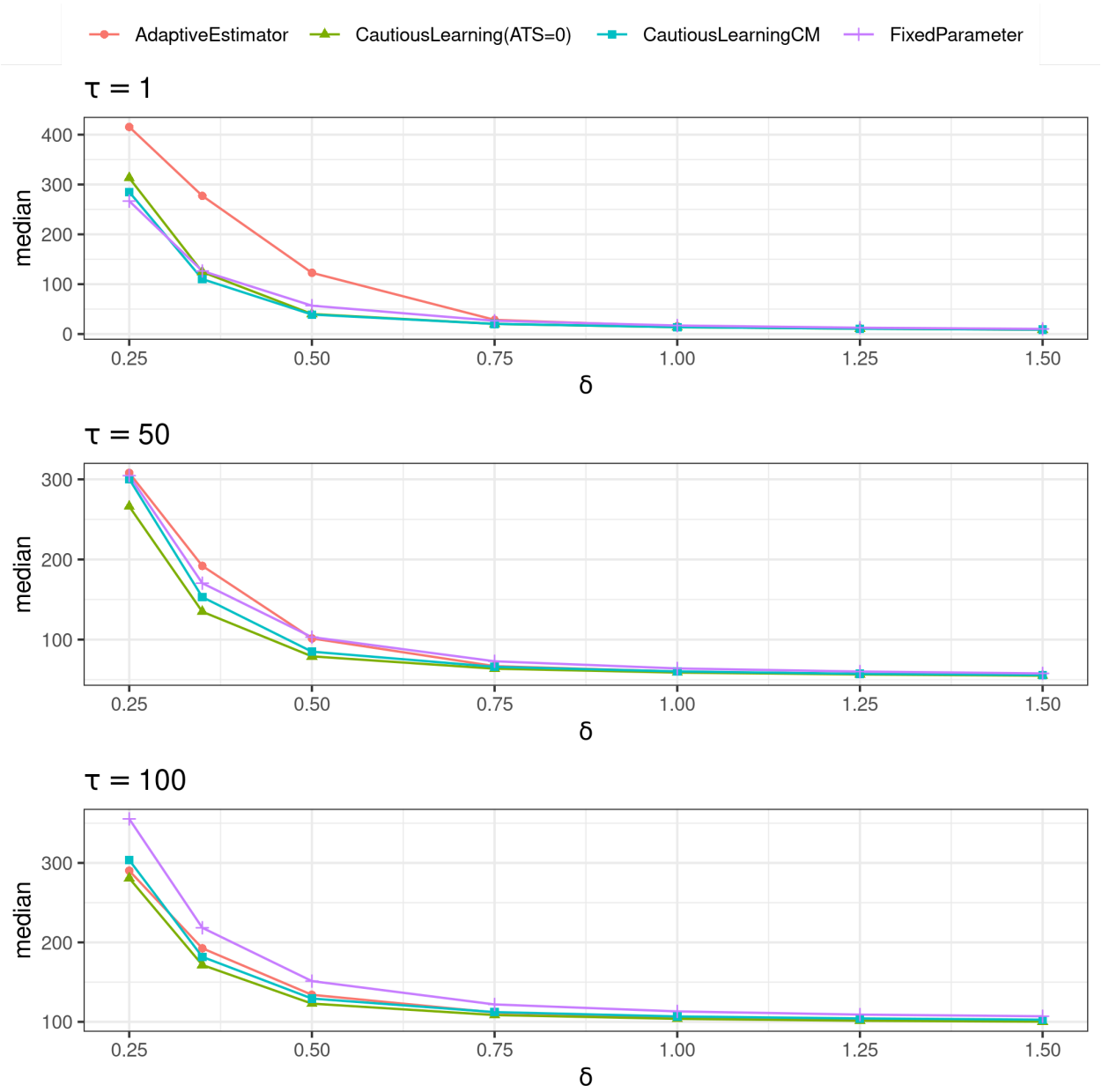


Figure 4: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.075$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

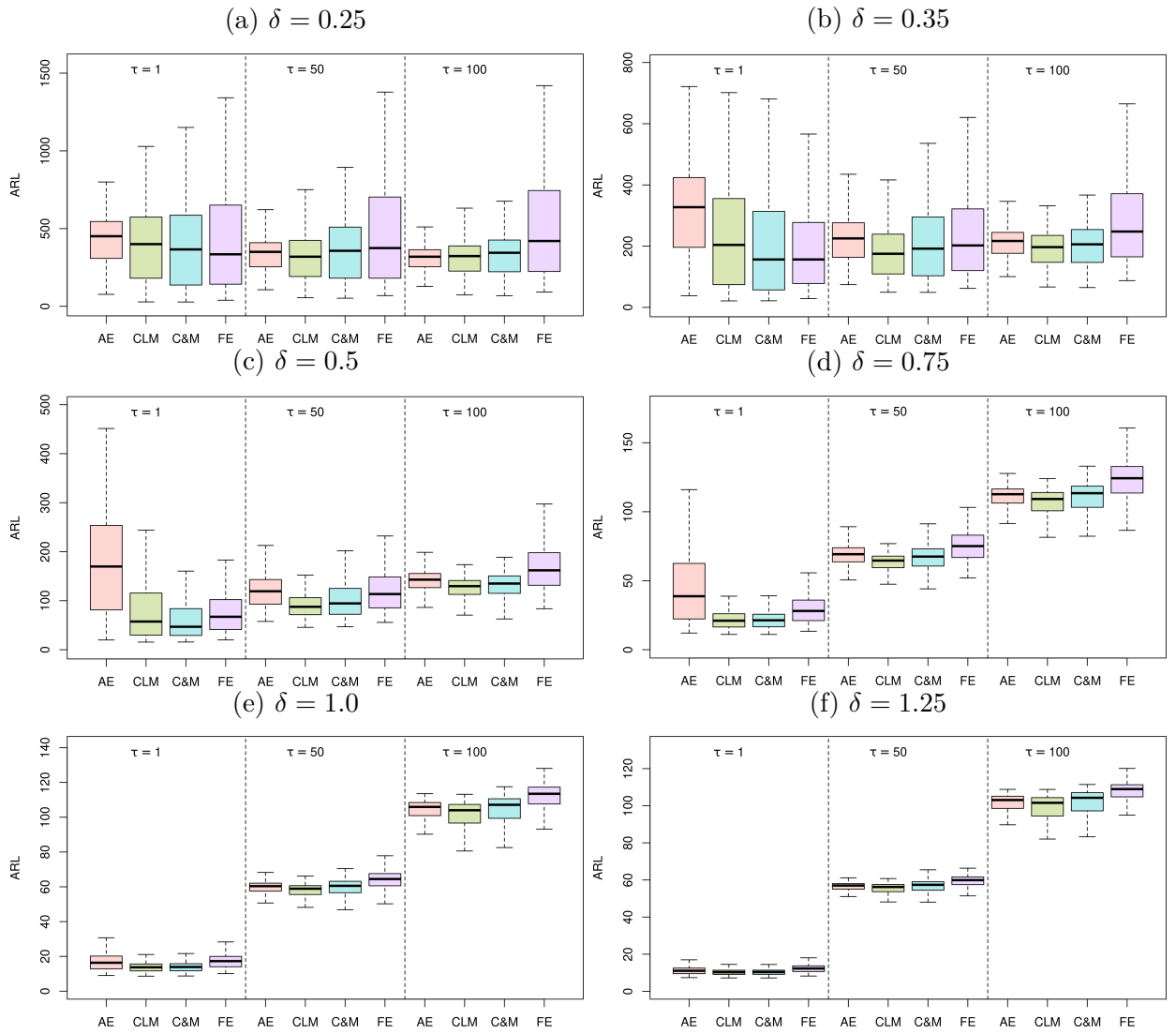


Figure 5: OC performance of the EWMA ($\lambda = 0.1$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

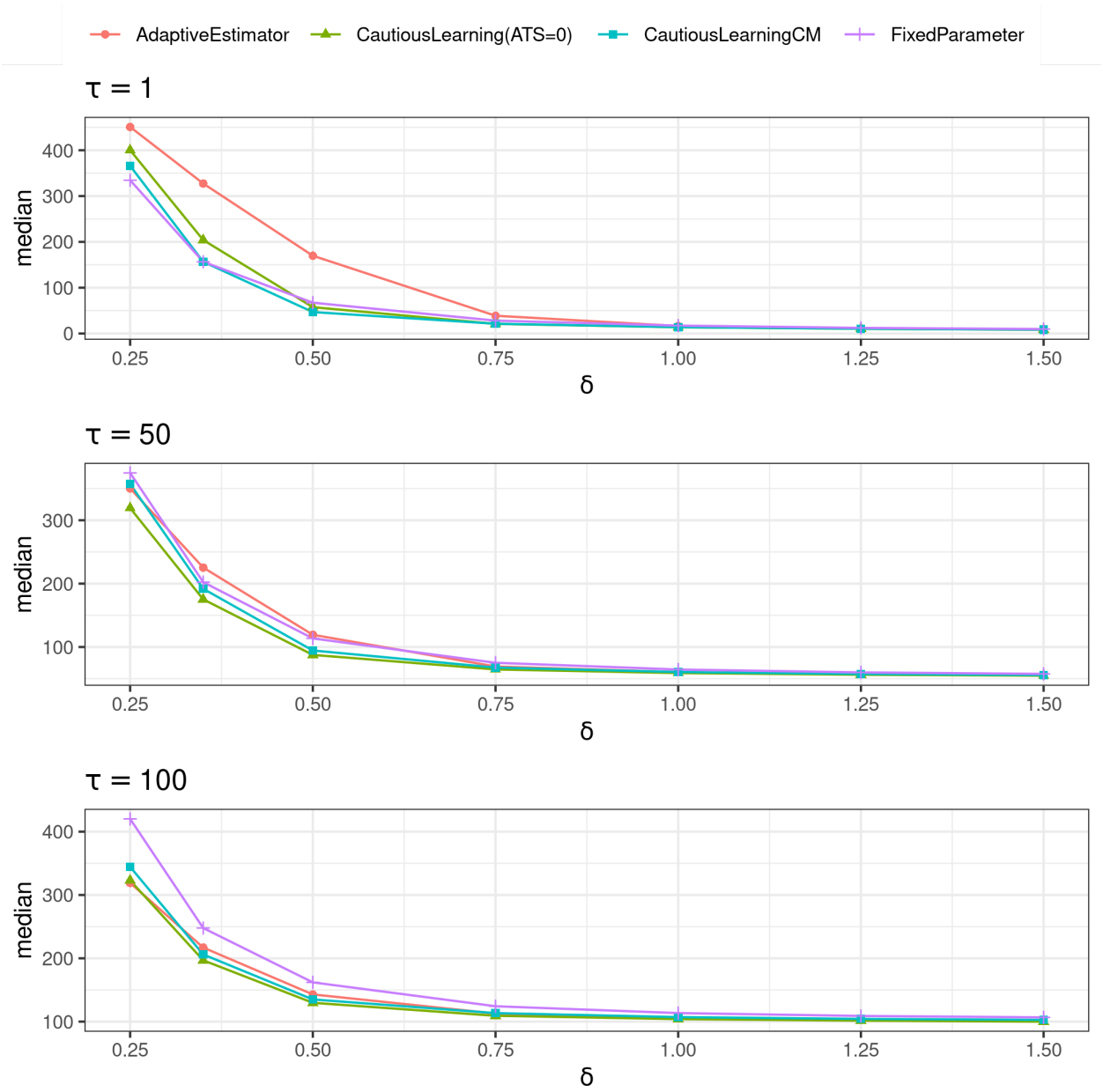


Figure 6: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.1$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

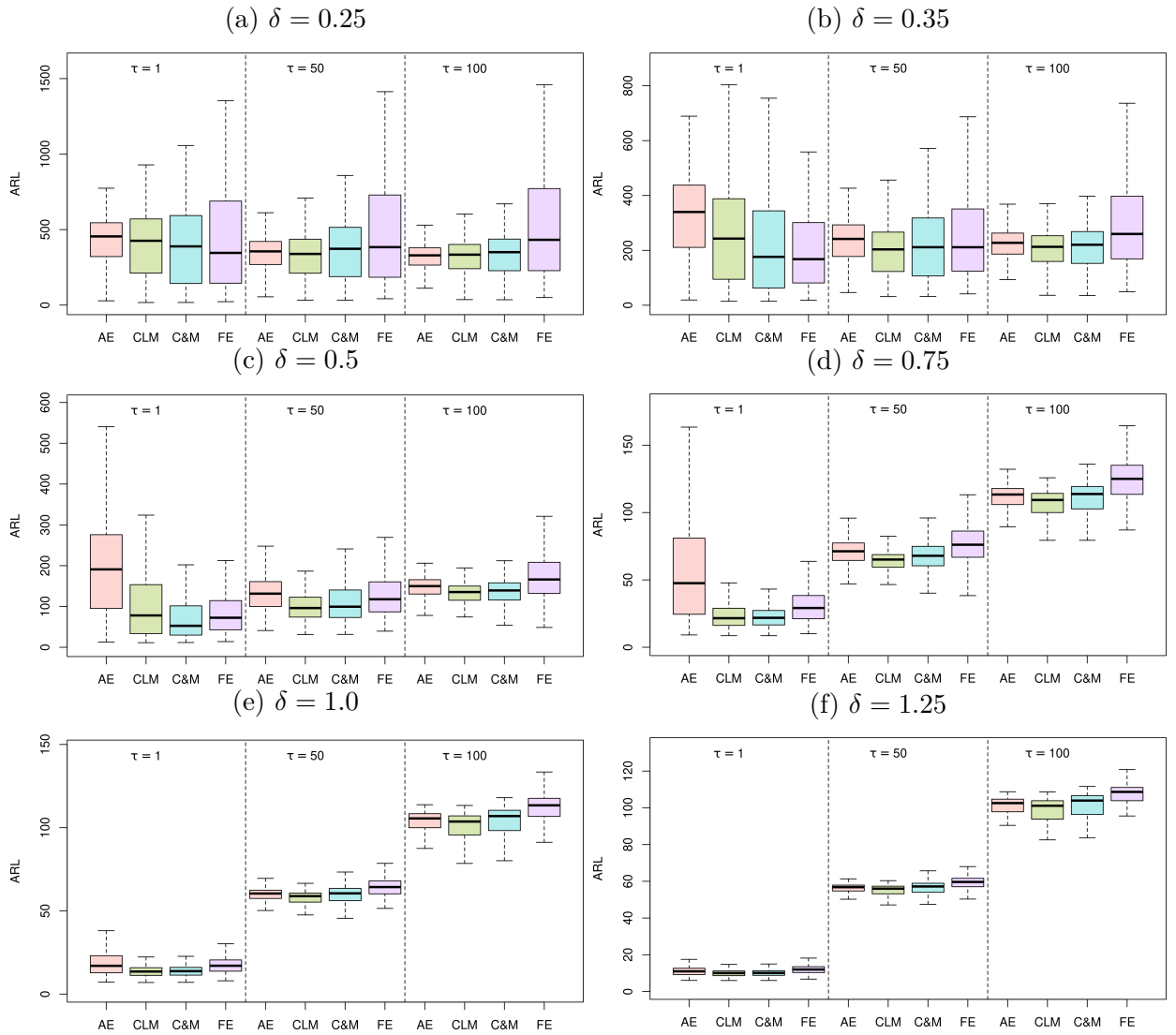


Figure 7: OC performance of the EWMA ($\lambda = 0.125$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

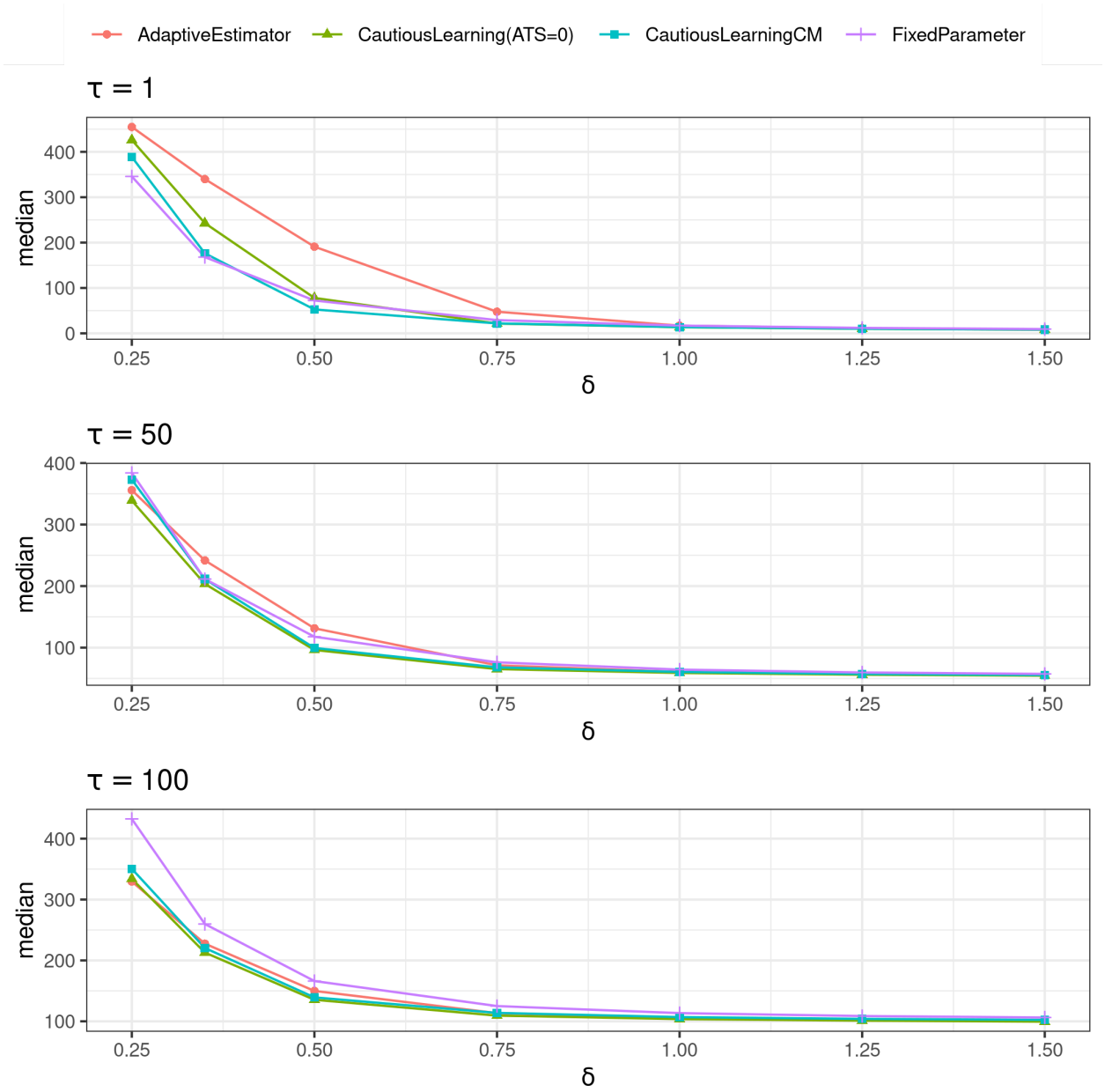


Figure 8: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.125$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

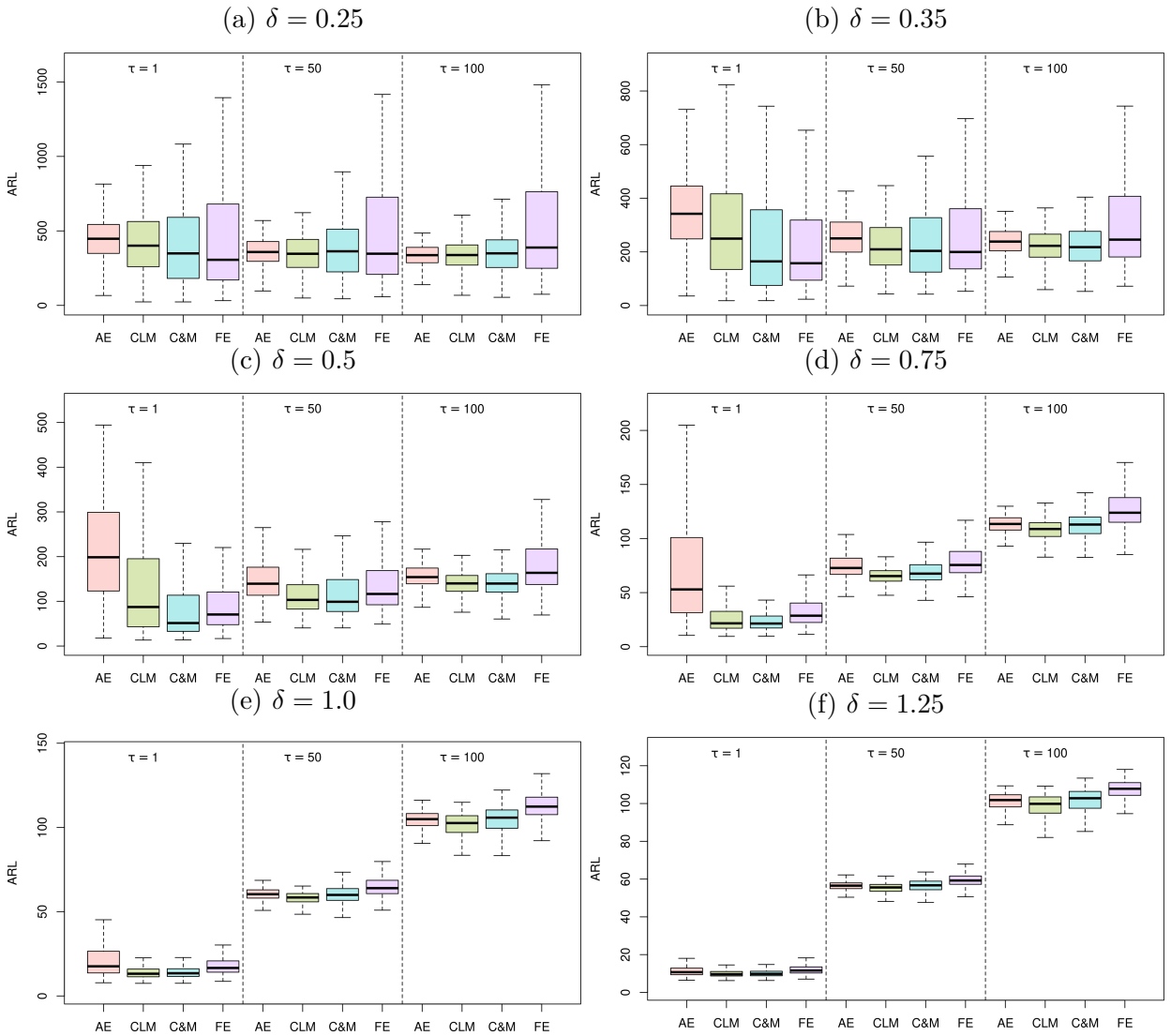


Figure 9: OC performance of the EWMA ($\lambda = 0.15$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

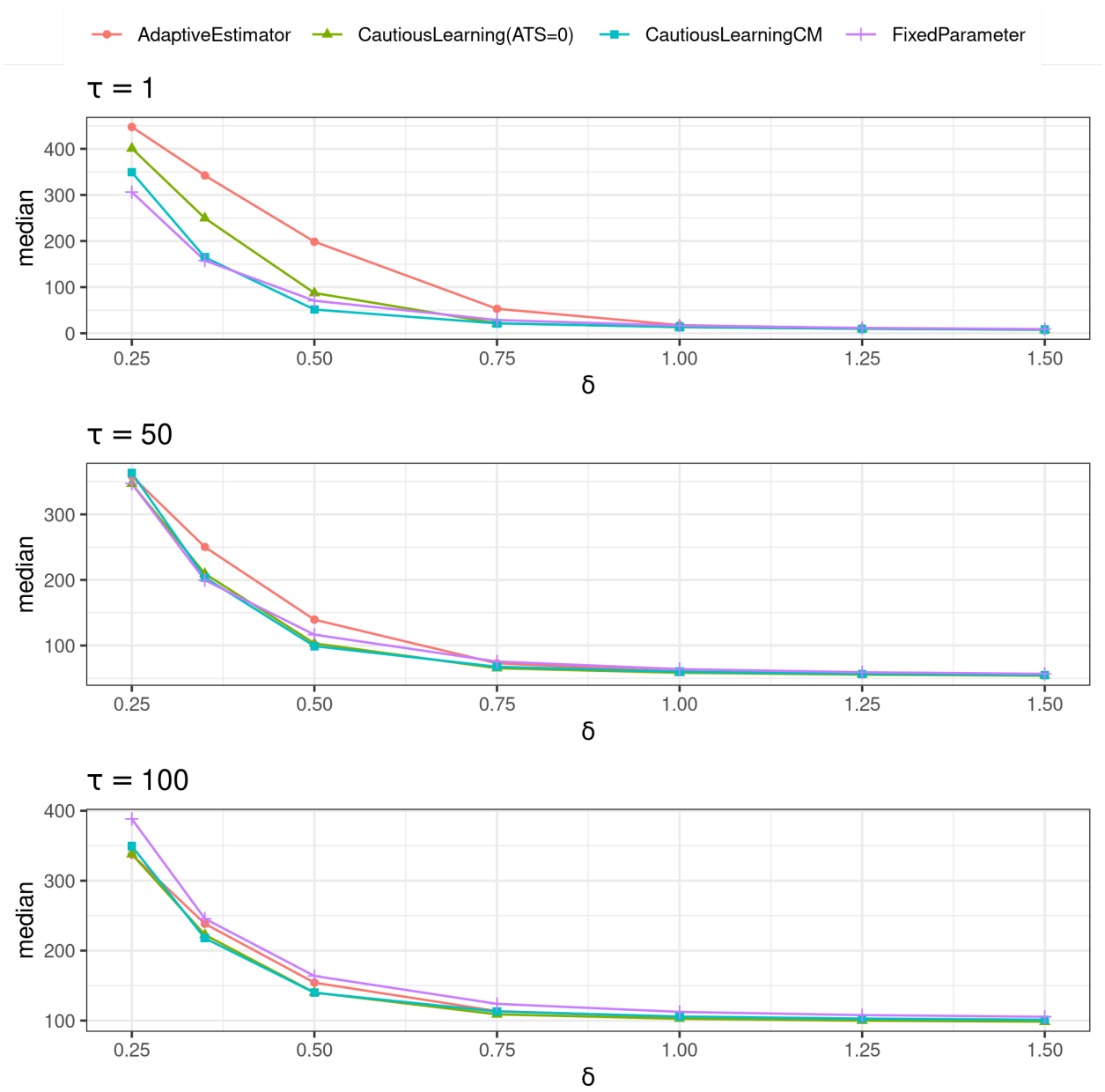


Figure 10: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.15$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

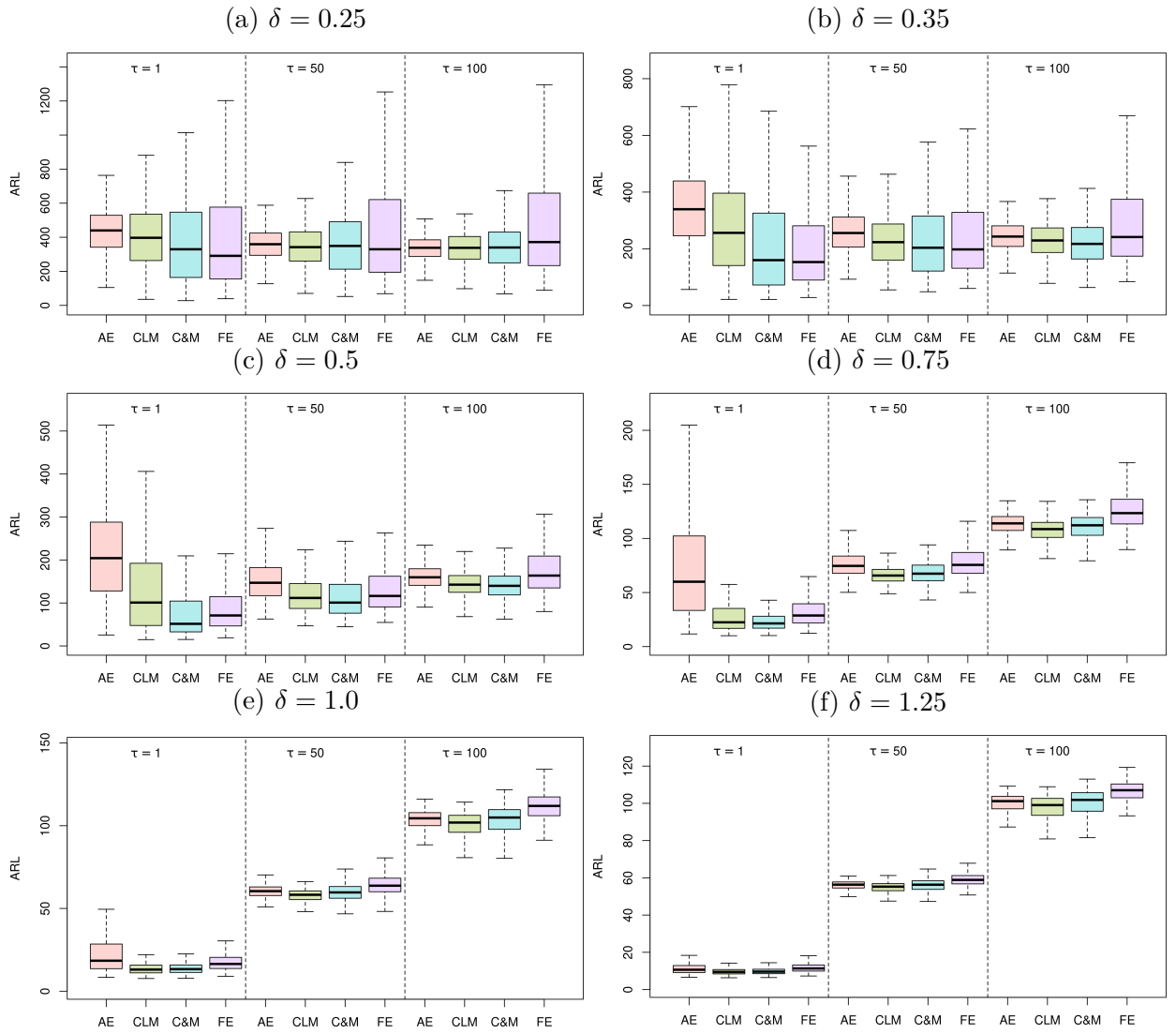


Figure 11: OC performance of the EWMA ($\lambda = 0.175$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

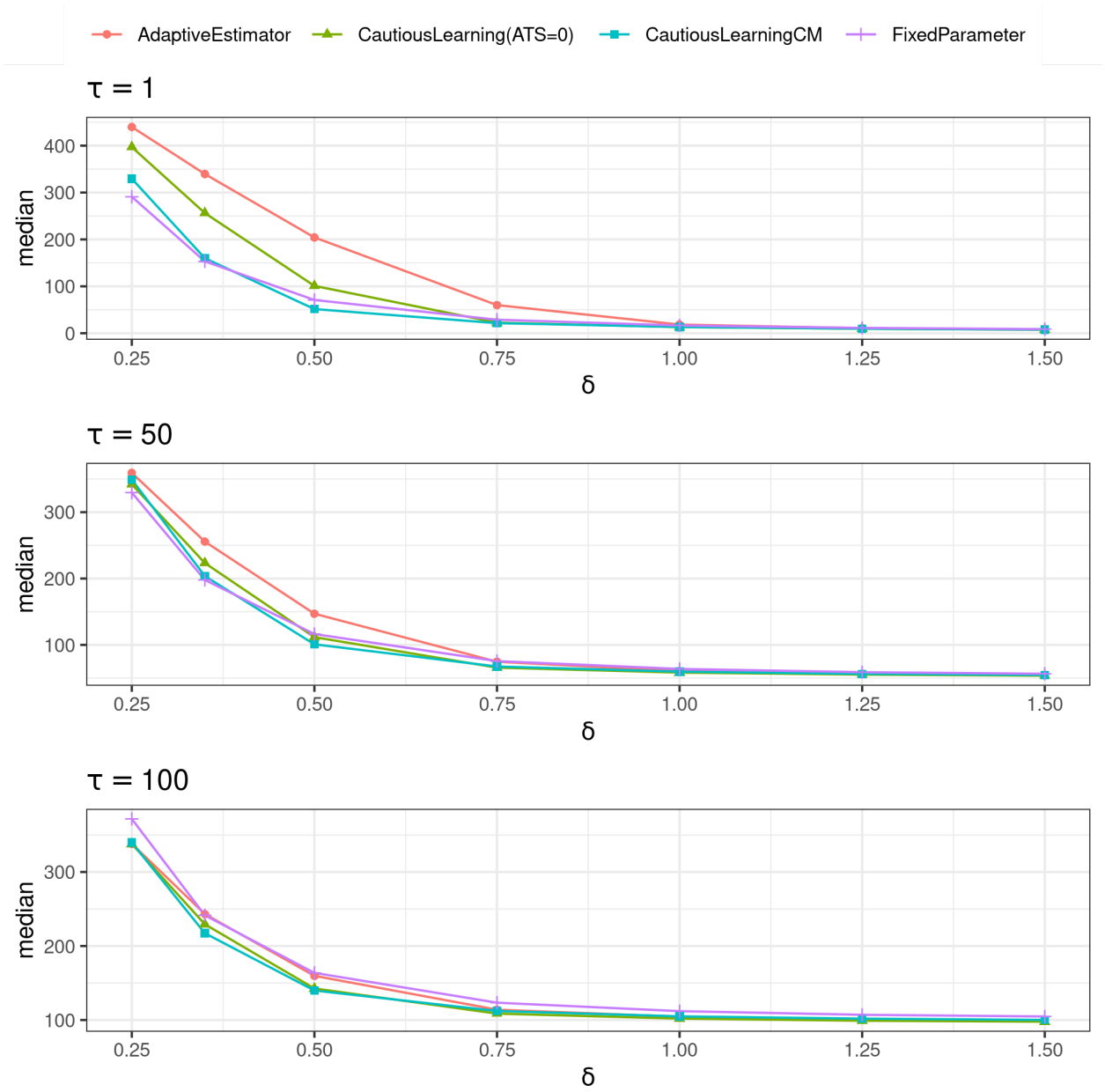


Figure 12: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.175$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

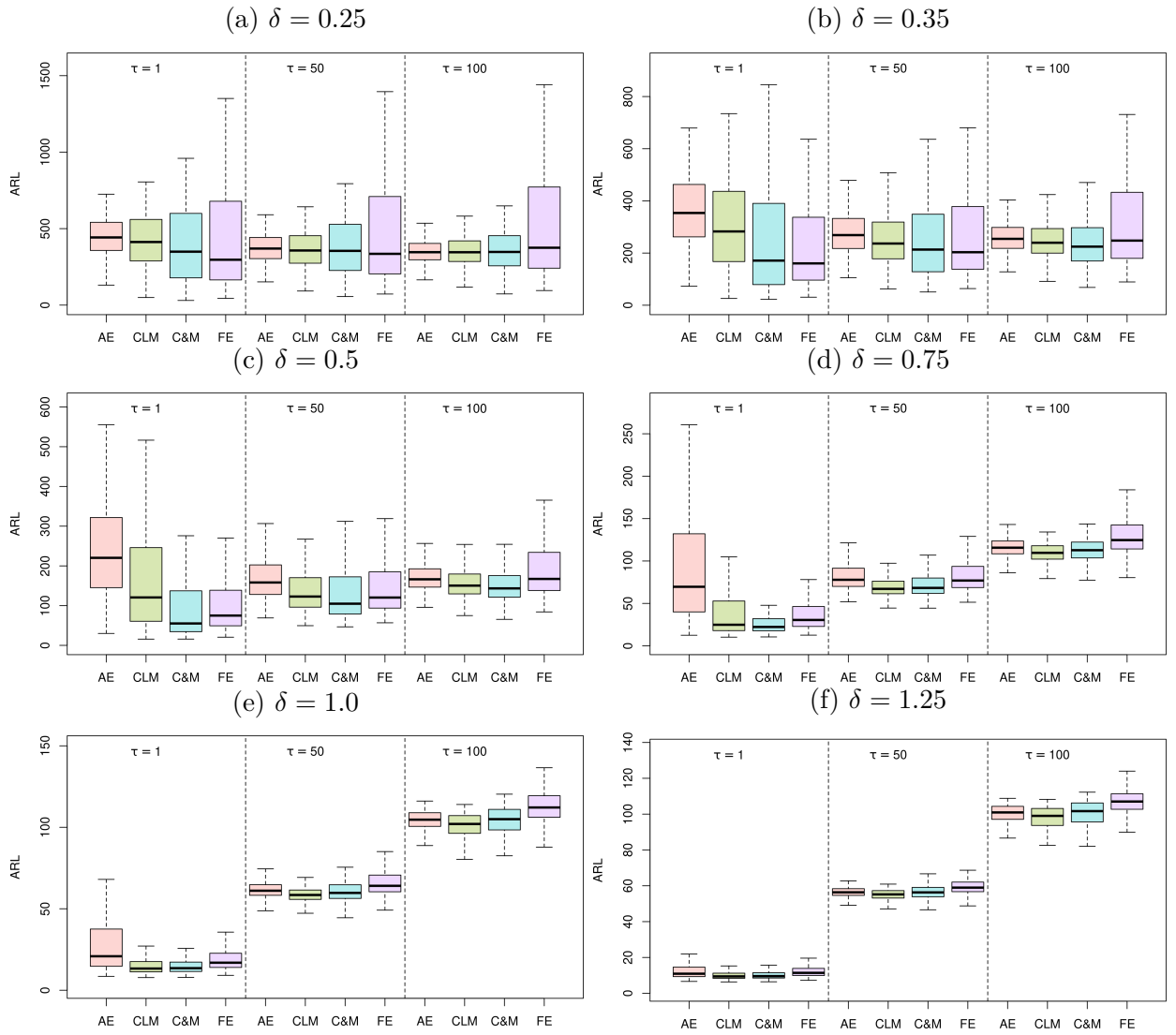


Figure 13: OC performance of the EWMA ($\lambda = 0.2$) control chart under fixed (FE), adaptive (AE), and cautious learning (CL) parameter updates when $\theta = 4$. Control charts satisfy the GICP condition with $\beta = 0.1$. Boxplots are based on the 200 simulated conditional ARLs.

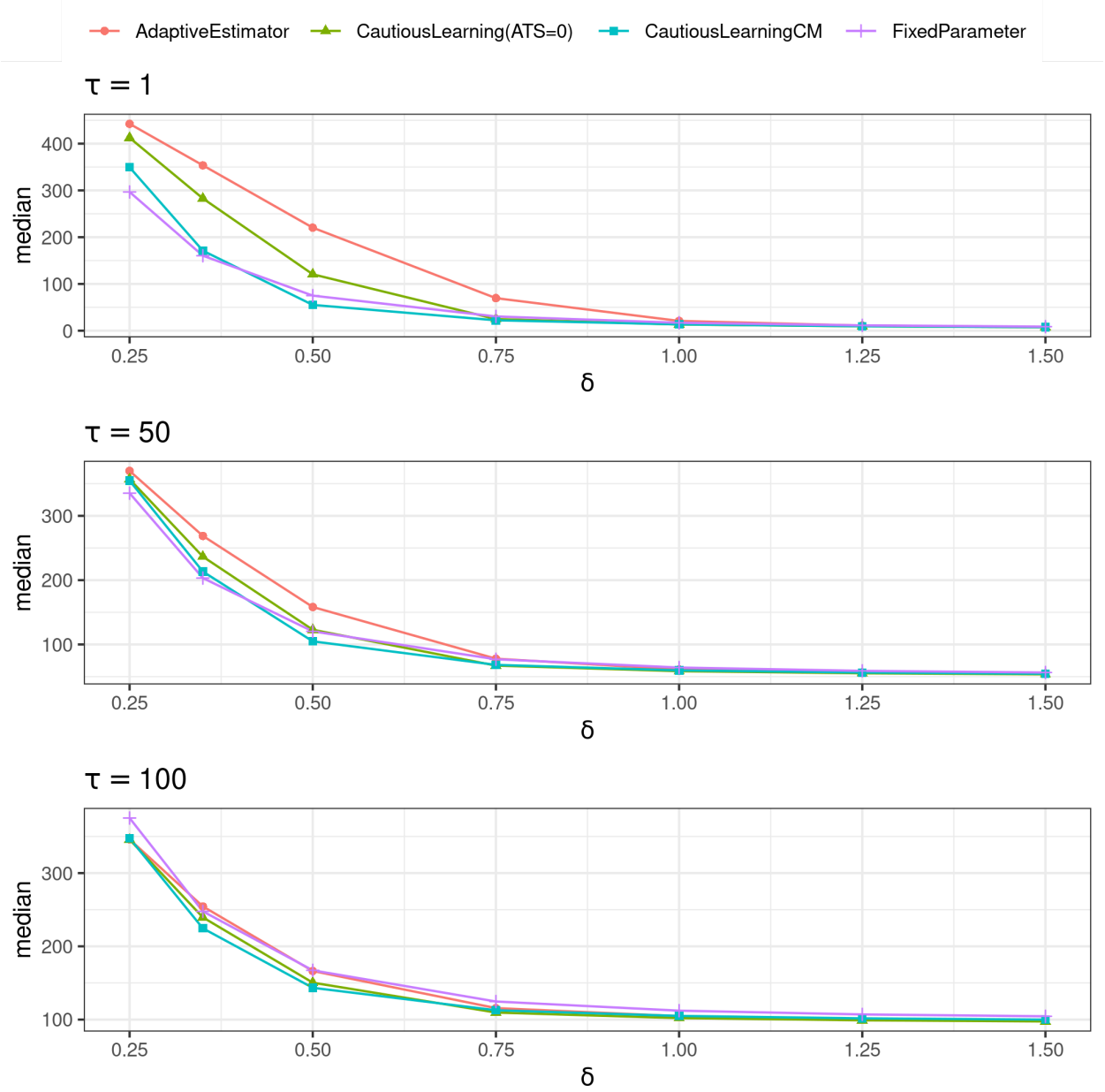


Figure 14: Median of the OC conditional ARL of the EWMA-type control chart under fixed (FE), adaptive (AE), cautious learning (CL) parameter updates for $\theta = 4$ and $\lambda = 0.2$. Control charts satisfy the GICP condition with $\beta = 0.1$. Plots are based on the 200 simulated conditional ARLs.

2 Application to the ICU admissions data

Here, we implement the code that reproduces the analysis on the ICU dataset in Section 5 of the paper.

First, we load the necessary packages and the environment variables that are provided in the source code.

```
using DrWatson
#@quickactivate "CautiousLearning"
quickactivate("$(homedir())/Documents/git/SPC/CautiousLearning")
```

```
using Distributions, Random
using Parameters
using SharedArrays
using DataFrames, CSV, Dates
using Plots, StatsBase, StatsPlots, LaTeXStrings, RCall
using StatisticalProcessControl

include(srcdir("generate_data.jl"))
include(srcdir("update_parameter.jl"))
include(srcdir("simulate_runs.jl"))
include(srcdir("cfg.jl"))
```

First, we load the dataset which is located in the `data/ICUadmissions` folder, relative to the main directory of the project. We consider the data concerning the New York City area in 2020.

```
fold = "ICUadmissions"
dat = DataFrame(CSV.File(datadir(fold,
"New_York_Forward_COVID-19_Daily_Hospitalization_Summary_by_Region.csv")))
nydat = filter(row -> row.Region == "NEW YORK CITY", dat)
nydat.date .= Date.(nydat[:, 1], dateformat"mm/dd/yyyy")
nydat = filter(row -> year(row.date) == 2020, nydat);
```

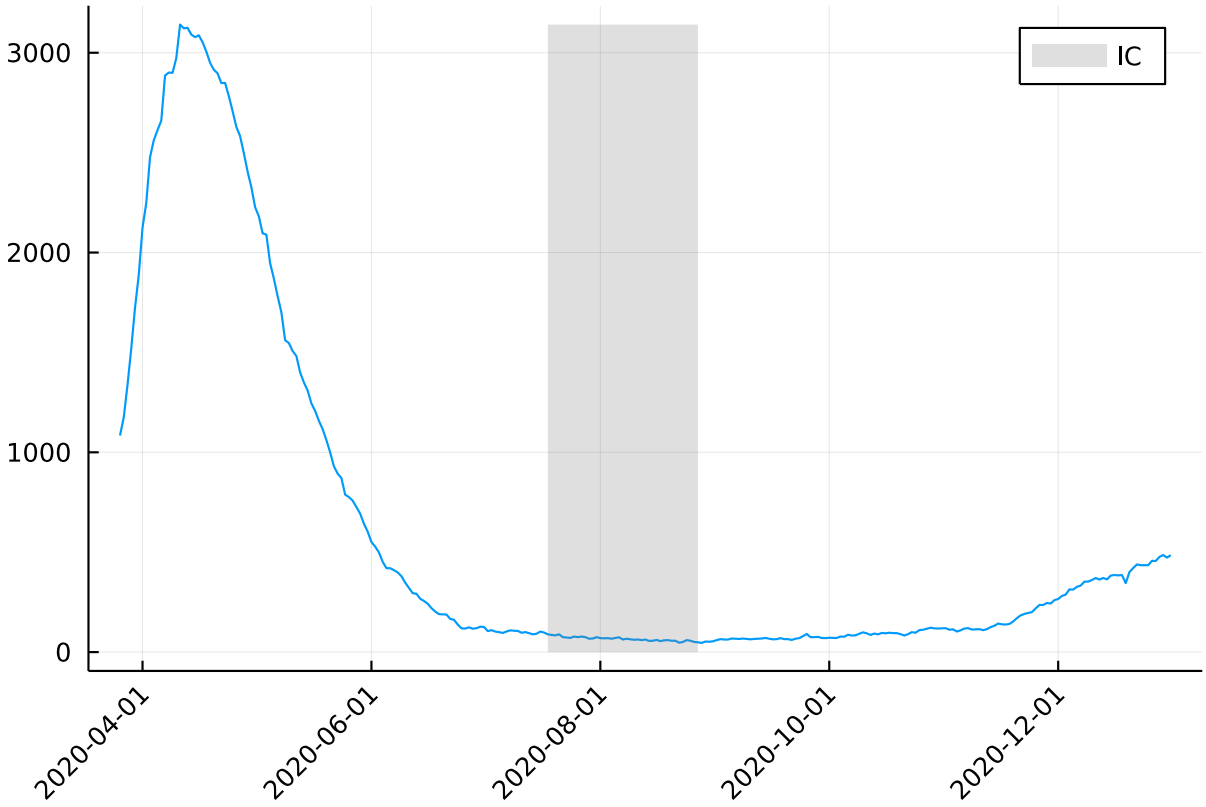
We obtain the daily ICU counts along with the corresponding dates, which start from March 2020.

```
using Plots.PlotMeasures
y2020 = nydat[:, 4]
days2020 = nydat[:, 5]
first(y2020, 10)
first(days2020, 10)
```

```
10-element Vector{Dates.Date}:
 2020-03-26
 2020-03-27
 2020-03-28
 2020-03-29
 2020-03-30
 2020-03-31
 2020-04-01
 2020-04-02
 2020-04-03
 2020-04-04
```

Then, we select the indices of the IC/OC datasets along with the initial sample size and prospective monitoring size. The following code reproduces Figure ?? in the paper.

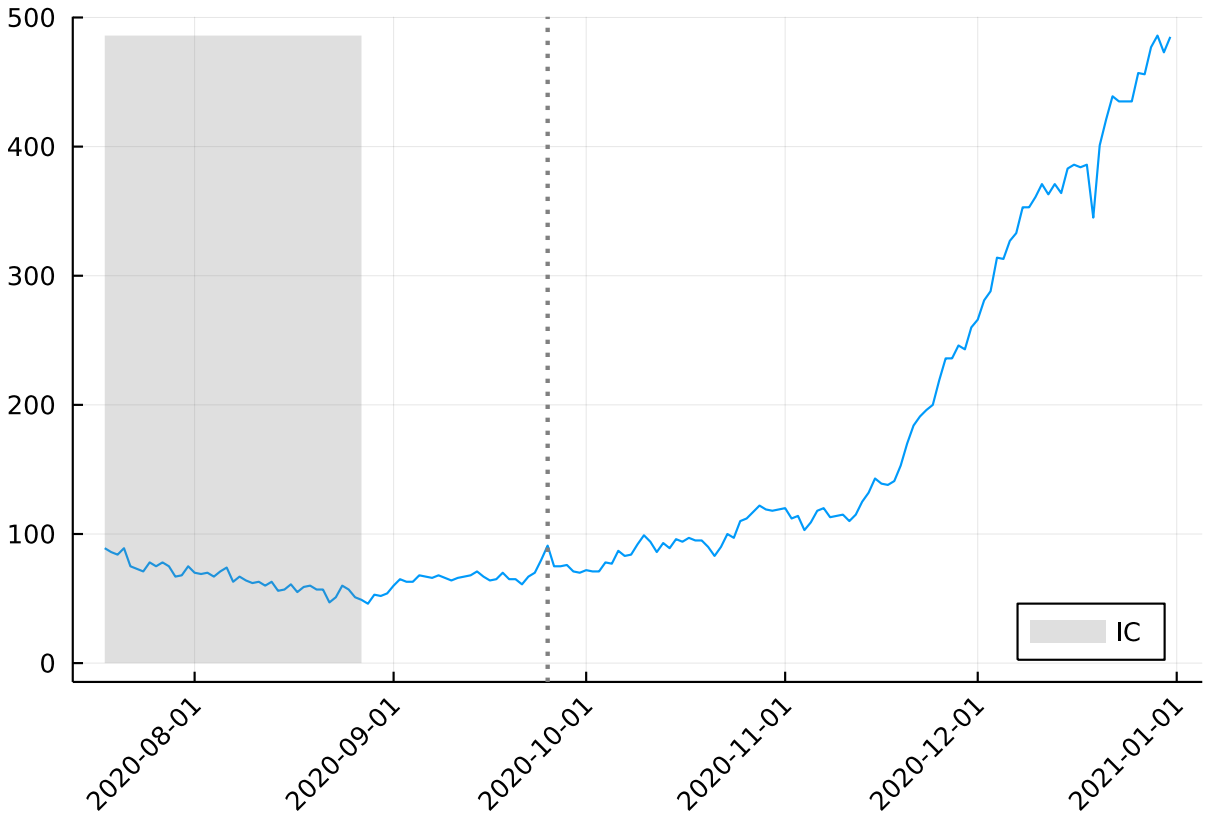
```
# ic_2020 = 115:175
ic_2020 = 115:155
oc_2020 = (ic_2020[end]+1):length(y2020)
n_ic = length(ic_2020)
n_oc = length(oc_2020)
pl = plot(days2020, y2020, label="", dpi=400, xrotation=45, bottom_margin=3mm)
plot!(pl, days2020[ic_2020], fill(0, n_ic), fillrange=fill(maximum(y2020), n_ic),
color=:gray, fillcolor = "gray", fillalpha=0.25, alpha=0.0, label="IC")
```



We then isolate the IC and OC data. This code reproduces Figure ?? in the paper.

```
y = y2020[[ic_2020; oc_2020]]
days = days2020[[ic_2020; oc_2020]]

ic_idx = 1:n_ic
oc_idx = (n_ic+1):(n_ic+n_oc)
yIC = y[ic_idx]
daysIC = days[ic_idx]
yOC = y[oc_idx]
daysOC = days[oc_idx]

println("In-control data: ", daysIC[[1, end]])
pl = plot(days, y, label="", dpi=400, xrotation=45, bottom_margin=3mm)
plot!(pl, days[1:n_ic], fill(0, n_ic), fillrange=fill(maximum(y), n_ic), color=:gray,
fillcolor = "gray", fillalpha=0.25, alpha=0.0, label="IC", legend=:bottomright)
τ = n_ic + 29
vline!([days[τ]], color=:gray, linestyle=:dot, linewidth=2,  label="", markersize=2.5)
display(pl)

println("Possible change-point location: ", days[τ])
```

```
In-control data: [Dates.Date("2020-07-18"), Dates.Date("2020-08-27")]
Possible change-point location: 2020-09-25
```



We can calculate the initial estimate $\hat{\theta}$ and set the desired nominal IC ARL of 500 alongside the GICP probability $\beta = 0.05$.

```
Arl0 = 500
beta = 0.05
thetaHat = mean(yIC)
println(thetaHat)
```

```
66.41463414634147
```

Next, we define a function that implements the control chart procedures using the various update mechanisms (AE, FE, CLM) that are defined in the source code.

```
function applyChart(ch, um, thetaHat, m, yprosp; seed = 123)
    # Random.seed!(seed)
    maxrl_i = length(yprosp)
    thetaHatVec = zeros(maxrl_i)
    thetaHatVec[1] = thetaHat
    di = 1
    diVec = Array{Int}(undef, maxrl_i)
    diVec[1] = di
    thetaHatCaut = thetaHat
    thetaHatCautVec = zeros(maxrl_i)
    thetaHatCautVec[1] = thetaHatCaut

    t_alarm = zeros(0)

    valueVec = zeros(maxrl_i)
    valueVec[1] = get_value(ch)
    i = 1
    while i < maxrl_i
        thetaHatCaut = thetaHatVec[i - di + 1]
        y = yprosp[i]
```

```
        ch = update_series(ch, chart_statistic(y, thetaHatCaut))
        thetaHat = update_parameter(thetaHat, y, i + m)
        i += 1
        valueVec[i] = get_value(ch)
        thetaHatVec[i] = thetaHat
        if check_update(ch, um)
            di = 1
        else
            di += 1
        end
        diVec[i] = di
        thetaHatCautVec[i] = thetaHatCaut
        if check_OC(ch)
            push!(t_alarm, i)
        end
    end

    return (t_alarm = t_alarm, dat = yprosp, chart_values = valueVec, limit_alarm =
get_limits(ch), limit_cautious = get_warning_limit(um), parameter_updates =
thetaHatCautVec, di = diVec)
end

applyChart (generic function with 1 method)
```

The function arguments are:

- `ch`: an `AbstractSeries` object, as defined in the `StatisticalProcessControl` package that is shipped with the source code.
- `um`: an `AbstractUpdate` object, as defined in the `StatisticalProcessControl` package that is shipped with the source code.
- `thetaHat`: the estimate of the IC mean.
- `m`: the initial sample size.
- `yprosp`: data onto which the control chart must be applied.
- `seed`: eventual seed that is passed to the GICP optimization for reproducibility.

When the above function is called sequence of count data, the returned values are

- `t_alarm`: the time of the first alarm.
- `dat`: the data onto which the control chart was applied.
- `chart_values`: the vector of values of the control chart.
- `limit_alarm`: the control chart limit.
- `limit_cautious`: if the control chart has a warning region, this contains the value.
- `parameter_updates`: the sequence of parameter values.
- `di`: the sequence of d_t's.

Furthermore, we define a function that applies the control chart to a prospective monitoring data after having corrected the control limit to satisfy the GICP condition.

```
function applyChartGICP(ch, um, yinit, yprosp, thetaHat, Arl0; beta::Union{Bool,
Float64} = 0.2, maxrl=1e04, verbose=true, seed=Int(rand(1:1e06)))
    m = length(yinit)
    if isa(um, CautiousLearning)
        Ats0 = get_ATS(um)
        if Ats0 != 0
            # Calculate limit if Ats0 != 0, otherwise use zero-restarting chart
            if verbose println("Calculating limits for target ATS...") end
            sa_ats = saControlLimits(ch, AdaptiveEstimator(), runSimulation, Ats0,
thetaHat, Poisson(thetaHat),
                                     m, verbose=false, Amin=0.1, maxiter=1e05,
                                     gamma=0.015, adjusted=true, seed=seed)
            um = CautiousLearning(L = sa_ats[:h], ATS = Ats0)
        else
            if verbose println("ATS = 0, skipping limit calculation.") end
        end
        # Estimate cautious learning limit
    end

    if beta == false
        chart = deepcopy(ch)
    else
        chart = adjust_chart_gicp(ch, um, yinit, thetaHat, runSimulation, m, Arl0,
beta=beta, verbose=verbose)
    end

    return applyChart(chart, um, thetaHat, m, yprosp)
end

applyChartGICP (generic function with 1 method)
```

The GICP control limit correction uses the `adjust_chart_gicp` function defined in the source code, which implements the control limit computation described in Subsection ??.

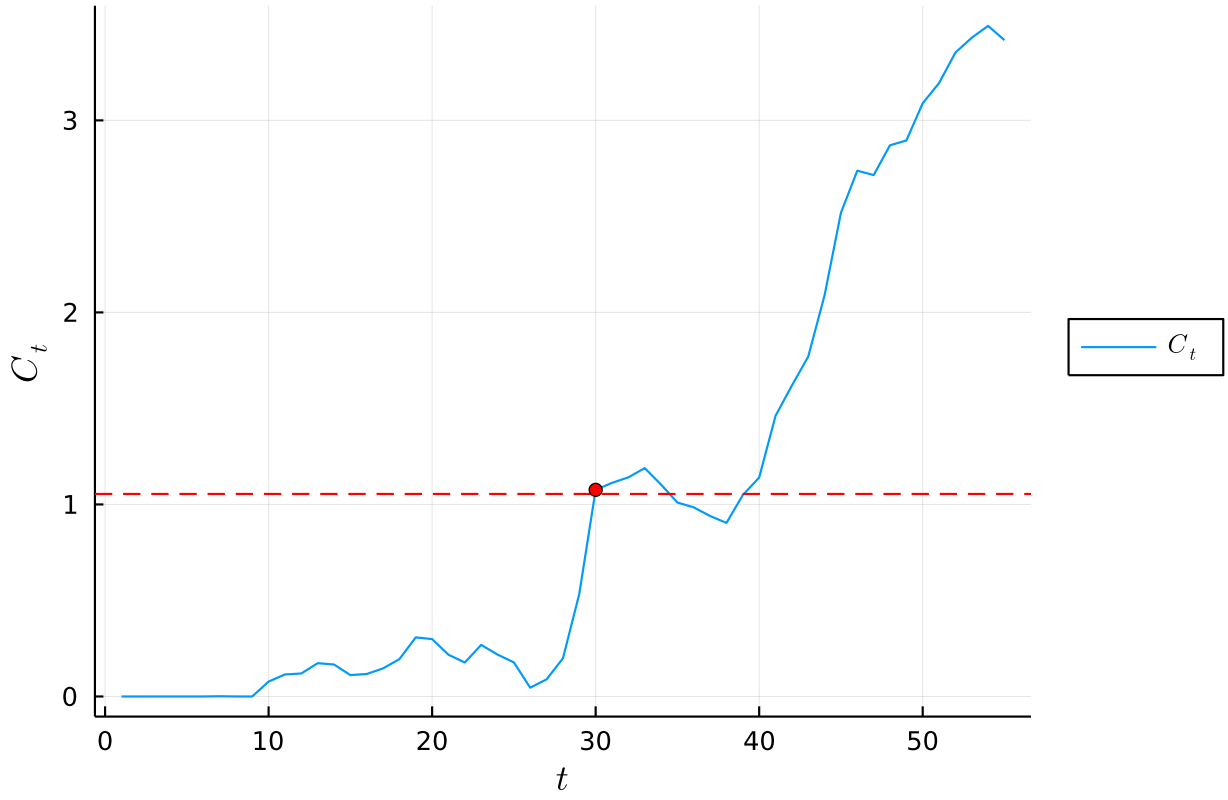
We finally apply the control charts using the methods and obtain the results. This code reproduces Figure ?? and Table ??.

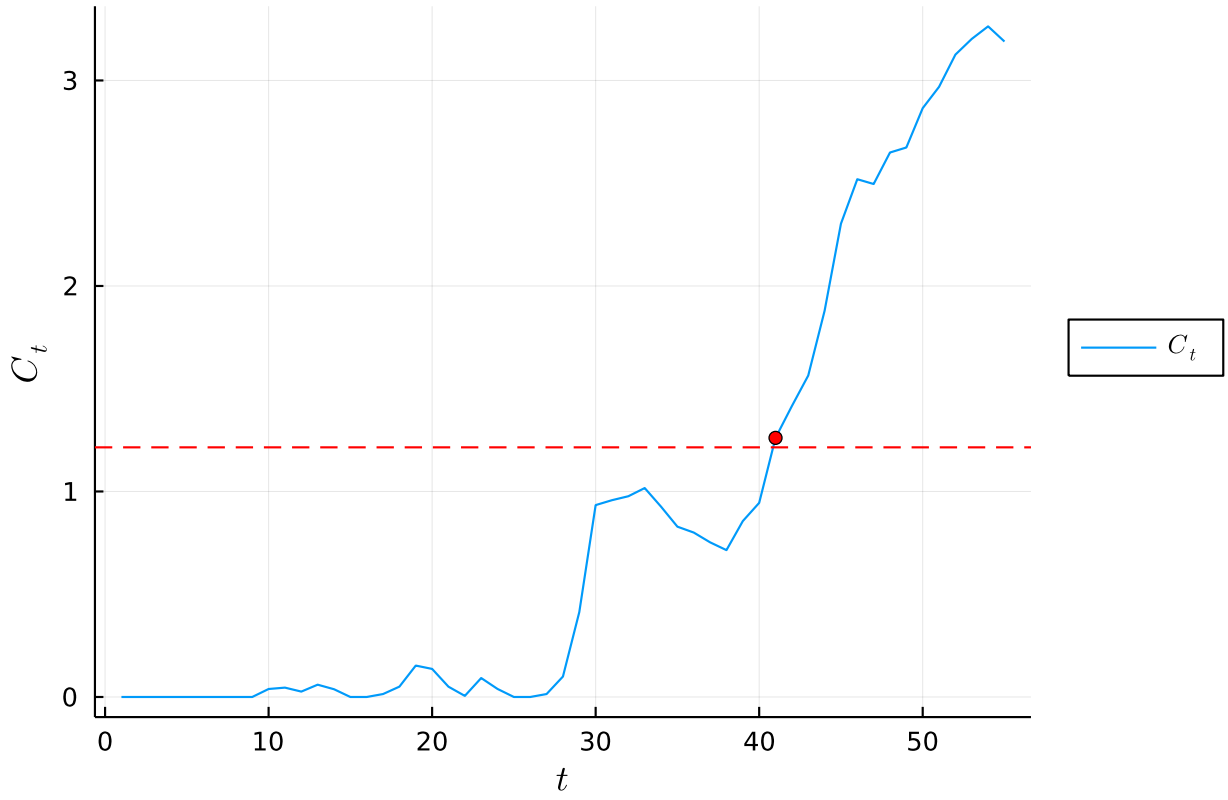
```
Random.seed!(2022-08-18)
D = Poisson
fname = plotsdir(fold, "alarms.jld2")

umVec = [CautiousLearning(ATS=0), FixedParameter(), AdaptiveEstimator()]
nms = ["CLM", "FE", "AE"]
perf = []
L = []
for i in eachindex(umVec)
    ch = signedEWMA(l=0.2, L = 1.0)
    um = umVec[i]
    res = applyChartGICP(ch, um, yIC, yOC, thetaHat, Arl0, beta=beta)
    append!(L, res[:limit_alarm])
    plotsave = plotsdir(fold, nms[i]*".png")
    pl = plot(res.chart_values[1:55], label=L"C_t", legend=:outerright, dpi=400)
    hline!([res.limit_alarm], style=:dash, colour="red", xlab=L"t", ylab=L"C_t",
label="")
    tau = Int(first(res.t_alarm))
    scatter!([tau], [res.chart_values[tau]], colour="red", label="")
    display(pl)
```

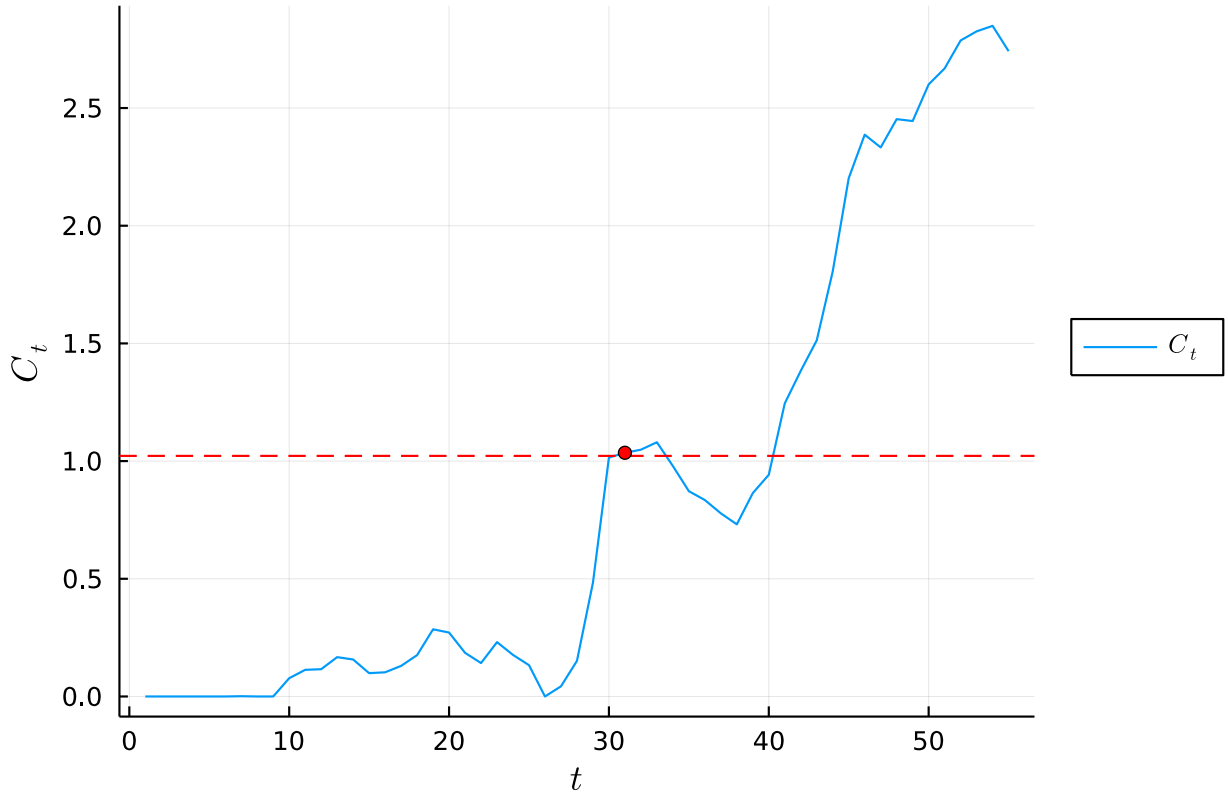
```
    append!(perf, tau)
    println(tau,"\t", daysOC[tau])
end

@rput nms
@rput perf
@rput L

tab = R"""
library(knitr)
library(kableExtra)
df = cbind(nms, perf, L)
tex_OC <- kable(df, format="latex", digits = 2, row.names=FALSE,
col.names=c("Estimator", "Alarm", "Limit"), escape=FALSE, align='c', linesep = "",
    caption = "Time to alarm of the one-sided EWMA control chart using the fixed (FE),
adaptive (AE) and the proposed cautious learning (CLM) update rules.", label="ICU OC
alarm")
""" |> rcopy

println(tab)

ATS = 0, skipping limit calculation.
Calculating GICP for extremum 1/1...
GICP done.
30      2020-09-26
Calculating GICP for extremum 1/1...
GICP done.
41      2020-10-07
Calculating GICP for extremum 1/1...
GICP done.
31      2020-09-27
```

Table 1: Time to alarm of the one-sided EWMA control chart using the fixed (FE), adaptive (AE) and the proposed cautious learning (CLM) update rules.

Estimator	Alarm	Limit
CLM	30	1.05469060798468
FE	41	1.2147825075143
AE	31	1.02214273823991


C_t
0
1
2
3
0
10
20
30
40
50
t
C_t


C_t
0
1
2
3
0
10
20
30
40
50
t
C_t


2.5
2.0
1.5
1.0
0.5
0.0
C_t
0
10
20
30
40
50
t
C_t