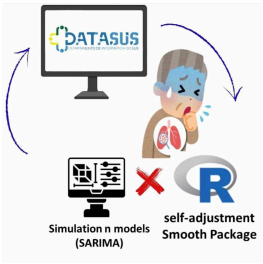
Dec 22, 2024

Algorithm with time series simulation versus auto-adjustment in R for predicting tuberculosis cases in Goiás.



DOI
https://dx.doi.org/10.17504/protocols.io.n92ldrw1og5b/v1

Laura Raniere Borges do Anjos[1], Leandro do Prado Assunção[1]

[1]IEEC

Leandro do Prado Assunção
IEEC

DOI: https://dx.doi.org/10.17504/protocols.io.n92ldrw1og5b/v1

Protocol Citation: Laura Raniere Borges do Anjos, Leandro do Prado Assunção 2024. Algorithm with time series simulation versus auto-adjustment in R for predicting tuberculosis cases in Goiás.. **protocols.io** https://dx.doi.org/10.17504/protocols.io.n92ldrw1og5b/v1

Protocol status: Working
We use this protocol and it's working

Created: December 21, 2024

Last Modified: December 22, 2024

Protocol Integer ID: 116783

Keywords: Tuberculosis, Goiás, Time Series, Auto-adjustment algorithm, Prediction, predicting tuberculosis case, predicting tb case, tb epidemic, estimation of an epidemiological trend, tuberculosis cases in goiá, tb in goiá, controlling tb, tuberculosis, time series model, cases of tb, epidemiological trend, mycobacterium tuberculosis, time series model with better parameter, planning of public health action, tb case, tb in the region, tb, time series simulation versus auto, time series simulation, prediction, time series, adjustment model, uncontrolled spread of the disease, public health action, raising significant public health concern, significant public health concern, planning, disease

Abstract

It is estimated that, globally, one in four people is infected with *Mycobacterium tuberculosis*, and approximately 5–10% of these individuals will develop tuberculosis (Tb) over their lifetime. In Goiás, in 2023, 1,499 cases of Tb were reported, representing a 9.42% increase compared to the previous year. This scenario highlights the uncontrolled spread of the disease in Goiás and worldwide, raising significant public health concerns. Predicting Tb cases is an essential tool for controlling the Tb epidemic. We developed an algorithm that constructs a time series model with better parameters than the auto-adjustment model of an R Studio library for prediction. This enabled the estimation of an epidemiological trend for Tb in Goiás. These findings can significantly contribute to the planning of public health actions and decision-making aimed at controlling Tb in the region.

Time series simulation algorithm versus auto-adjustment

1 ### Loading the libraries

```
library(dplyr); library(openxlsx); library(DataExplorer)
library(tidyr); library(ggplot2); library(lubridate)
library(magrittr); library(patchwork); library(cowplot)
library(tseries); library(Kendall); library(TTR);
library(forecast); library(smooth); library(sarima);
library(seasonal); library(scales); library(stringr)
```

2 ### Loading all data

```
diretorio = "C:/Users/Path"
nomes_arquivos = list.files(path = diretorio)
anos = c(2001:2023)
for(i in 1:length(anos)){
assign(paste0("dat_", anos[i]), read.csv(nomes_arquivos[i],
skip = 3, nrows = 12, fileEncoding = "ISO-8859-1", sep = ";"))}
```

3 ### Function to add the age range variable to the database

```
trasnform_table = function(dat, ano){
 faixa_etaria = c("< 1 year", "1 - 4 years", "5 - 9 years",
                  "10 - 14 years", "15 - 19 years", "20 - 39 years",
                  "40 - 59 yeaas", "60 - 64 years", "65 - 69 years",
                  "70 - 79 years", "> 80 years", "Total")
 mes = tolower(month.name)
 meses = str_sub(mes, 1, 3)
 anos = c(2001:2023)
 dat_org = data.frame(
  Faixa = c(),
  Mes =  c(),
  Ano =  c(),
  Casos =  c()
 )
 for(i in 1:length(faixa_etaria)){
  dat_org1 = data.frame(
   Faixa = rep(faixa_etaria[i], 12),
   Mes = meses,
   Ano = rep(ano, 12),
   Casos = unlist(dat[i,2:13])
```

```
    )
    dat_org = rbind(dat_org, dat_org1)
  }
  return(dat_org)
}
```

4

```
### Joining all tables

lista_tabela = c("dat_2001", "dat_2002", "dat_2003", "dat_2004", "dat_2005",
                 "dat_2006", "dat_2007", "dat_2008", "dat_2009", "dat_2010",
                 "dat_2011", "dat_2012", "dat_2013", "dat_2014", "dat_2015",
                 "dat_2016", "dat_2017", "dat_2018", "dat_2019", "dat_2020",
                 "dat_2021", "dat_2022", "dat_2023")
dat_final = data.frame( Faixa = c(), Mes =  c(), Ano =  c(), Casos =  c())
for(i in 1:length(lista_tabela)){
dat_fin = trasnform_table(get(lista_tabela[i]), anos[i])
dat_final = rbind(dat_final, dat_fin)}
```

5

```
### Filtered only total cases of tuberculosis in Goias

dat_total = dat_final %>% filter(Faixa == "Total")
```

6

```
### Creating a dataframe with grouped by cases by year

dat_total$Casos = as.numeric(dat_total$Casos)
dat_ano = aggregate(Casos ~ Ano, dat_total, sum)
```

7

```
### Exporting the data

openxlsx::write.xlsx(dat_ano, "Quantidade_casos_ano_Tuberculose_Goias.xlsx")
```

8

```
### Splitting data into training and testing

dat_total_22 = dat_total %>% filter(!Ano %in% c(2023))
dat_total_23 = dat_total %>% filter(Ano %in% c(2023))
dat_total_22$Casos = as.numeric(dat_total_22$Casos)
```

9

```
### Time series with data up to 2022

Z2 = ts(dat_total_22[,4], start = 2001, freq=12)
```

10
```
### Time series graph up to 2022

plot(Z2, ylab = "S(Z)", xlab = "Time (months)",
    main = "Tuberculosis cases - Goiás (2001 - 2022)",
    ylim = c(0,(max(dat_total_22$Casos))),
    xlim = c(2001, 2022), type='o')
```

11
```
### Hypothesis testing to test stationarity, trend and seasonality

adf.test(Z2)
MannKendall(Z2)
ano1 = seq(1,12,1)
tempo = c(rep(ano1, 22))
kruskal.test(as.numeric(Z2), tempo)
b2=decompose(Z2)
plot(b2)
```

12
```
### Checking the self-relation of the series

autocorrelations(Z2)
plot(autocorrelations(Z2))
Acf(Z2, "Correlation")
acf(Z2, plot = TRUE, type = "correlation")
```

13
```
### Best model simulation function

simulacao_series = function(Z1, P, D, Q){
 result = data.frame(Models = c(), Parameters = c(),
                AIC = c(), BIC = c())
 for(p in 0:P) {
  for(d in 0:D) {
   for(q in 0:Q) {
    modelo = tryCatch({
     Arima(Z1, order = c(p, d, q), seasonal = c(p, d, q))
    }, error = function(e) {
     NULL
    })
    if (!is.null(modelo)) {
     dat = data.frame(P = p, D = d, Q = q,
                 Criterio_AIC = AIC(modelo),
                 Criterio_BIC = BIC(modelo))
     result = rbind(result, dat)
    }
   }
```

```
    }
  }
  result1 = result %>% arrange(-Criterio_AIC) %>%
  mutate(Parameters = as.factor(paste0(P, D, Q)))
  result2 = aggregate(cbind(Criterio_AIC, Criterio_BIC) ~ Parameters, result1, mean)
  return(result2)
}
```

14 ### Simulation with parameters P = 3, Q = 2 and D = 3

```
result1 = simulacao_series(Z2, 3, 2, 3)
```

15 ### Sorting the data by the best model

```
result1 = result1 %>% arrange(Criterio_AIC, Criterio_BIC)
```

16 ### Exporting the data

```
openxlsx::write.xlsx(result1, "Modelos_Testados.xlsx")
```

17 ### Using the best model

```
model2 = Arima(Z2, order = c(3, 1, 1), seasonal = c(3, 1, 1))
```

18 ### Self-tuning model using the smooth library

```
model3 = auto.ssarima(Z2)
```

19 ### Generating 2023 predictions

```
previsoes_model2 = data.frame(forecast(model2, h=12))
previsoes_model3 = data.frame(forecast(model3, h=12))
```

20 ### Creating a dataframe with point and interval estimates

```
dat_total_pred_mse = dat_total_23 %>% mutate(
                Pred_model2 = round(previsoes_model2$Point.Forecast),
                LI_model2 = round(previsoes_model2$Lo.95),
                LS_model2 = round(previsoes_model2$Hi.95),
                Pred_model3 = round(previsoes_model3$Point.Forecast),
                LI_model3 = round(previsoes_model3$Lo.0.95),
                LS_model3 = round(previsoes_model3$Hi.0.95)
)
```

21 ### Exporting the data

```
openxlsx::write.xlsx(dat_total_pred_mse, "tabela_pred_modelos.xlsx")
```

22
```
### Applying the Box-pierce test to test the autocorrelation of residuals

Box.test(model2$residuals)
Box.test(model3$residuals)
```

23
```
### 2023 Forecast Chart

plot(forecast(model2, 12),ylab = "S(Z)", xlab = "Time (months)",
    main = "Prediction model 2 of Tuberculosis cases - 2021 e 2023",
    ylim = c(0,200), xlim = c(2001, 2023), type='o')

plot(forecast(model3, 12),ylab = "S(Z)", xlab = "Time (months)",
    main = "Prediction model 3 of Tuberculosis cases - 2021 e 2023",
    ylim = c(0,200), xlim = c(2001, 2023), type='o')
```

24
```
### Creating a dataframe with the actual data from 2023 and
### the two predictions from the best model from simulation and self-tuning

dat_total_23_pred = dat_total_23 %>% mutate(
                        Pred_model2 = round(previsoes_model2$Point.Forecast),
                        Pred_model3 = round(previsoes_model3$Point.Forecast))
```

25
```
### Transforming the cases variable into numeric

dat_total_23_pred$Casos = as.numeric(dat_total_23_pred$Casos)
```

26
```
### Calculate Mean Squared Error (MSE)

mse_model2 = mean((dat_total_23_pred$Casos - dat_total_23_pred$Pred_model2)^2)
mse_model3 = mean((dat_total_23_pred$Casos - dat_total_23_pred$Pred_model3)^2)
```

27
```
### Calculate the Root Mean Square Error (RMSD)

rmsd_model2 = sqrt(mse_model2)
rmsd_model3 = sqrt(mse_model3)
```

28
```
### MSE and RMSE results of the two models

cat("MSE_model2:", mse_model2, "\n")
cat("MSE_model3:", mse_model3, "\n")
cat("RMSD_model2:", rmsd_model2, "\n")
cat("RMSD_model3:", rmsd_model3, "\n")
```

29
```
### Creating the series with all data (2001 - 2023)

Z1 = ts(dat_total[,4], start = 2001, freq=12)
```

30 ### Time series graph with all data (2001 - 2023)

```
plot(Z1,
    ylab = "Cases of tuberculosis",
    xlab = "Time (months)",
    main = "Tuberculosis cases - Goiás (2001 - 2023)",
    ylim = c(0, max(dat_total$Casos)),
    xlim = c(2001, 2023),
    type = 'o',
    pch = 16,
    col = "black",
    lwd = 2,
    cex = 1,
    bg = "white",
    xaxt = "n",
    yaxt = "n"

axis(1, at = seq(2001, 2023, by = 1), labels = seq(2001, 2023, by = 1), las = 2)
axis(2, las = 1)
grid(col = "gray", lty = "dotted")
```

31 ### ### Hypothesis testing to test stationarity, trend and seasonality

```
adf.test(Z1)
MannKendall(Z1)
ano1 = seq(1,12,1)
tempo = c(rep(ano1, 23))
kruskal.test(as.numeric(Z1), tempo)
b1=decompose(Z1)
plot(b1)
```

32 ### Checking the self-relation of the series

```
autocorrelations(Z1)
plot(autocorrelations(Z1))
Acf(Z1, "Correlation")
acf(Z1, plot = TRUE, type = "correlation")
```

33 ### SARIMA models for all data considering the parameters
of the best simulation model

```
model1 = Arima(Z1, order = c(3, 1, 1), seasonal = c(3, 1, 1))
```

34 ### Generating forecasts for 2024, 2025 and 2026

```
previsoes = data.frame(forecast(model1, h=36))
```

35 ### Exporting the data

```
openxlsx::write.xlsx(previsoes, "predicoes_2024_ate_2026.xlsx", rowNames = TRUE)
```

36 ### Applying the Box-pierce test to test the autocorrelation of residuals

```
Box.test(model1$residuals)
```

37 ### Table with all point and interval estimates for the years 2024, 2025 and 2026

```
dat_pred = data.frame(
  mes = rep(seq(1:12),3),
  ano = c(rep(2024,12), rep(2025,12), rep(2026,12)),
  quantidade = round(previsoes$Point.Forecast)
)
```

38 ### Prediction graphs

```
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2) + 0.1)  s
plot(forecast(model1, 36),
     ylab = "Cases of tuberculosis",
     xlab = "Time (months)",
     main = "Prediction of Tuberculosis cases - 2025 e 2026",
     ylim = c(0, 200),
     xlim = c(2001, 2026),
     type = 'o',
     pch = 16,
     col = "black",
     lwd = 1,
     cex = 1,
     bg = "white",
     xaxt = "n",
     yaxt = "n"
)
axis(1, at = seq(2001, 2026, by = 1), labels = seq(2001, 2026, by = 1), las = 2)
axis(2, las = 1)
grid(col = "gray", lty = "dotted")
forecast_data = forecast(model1, 36)
lines(forecast_data$mean, col = "blue", lwd = 2)
lines(forecast_data$lower[, 2], col = "red", lty = 2)
lines(forecast_data$upper[, 2], col = "red", lty = 2)
```

39 ### Chart detailing the years 2024, 2025 and 2026 by month

```
plot(dat_pred$mes[dat_pred$ano == 2024], dat_pred$quantidade[dat_pred$ano ==
2024],
    type = 'o',
    xlab = "Month",
    ylab = "Cases of tuberculosis",
    xlim = c(1, 12),
    ylim = c(min(dat_pred$quantidade), max(dat_pred$quantidade)),
    col = "blue",
    pch = 16,
    main = "Tuberculosis Cases by Month (2024-2026)",
    xaxt = "n",
    lty = 1
)
axis(1, at = 1:12, labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))
pch_values = c(16, 17, 18)
col_values = c("black", "red", "blue")
lines(dat_pred$mes[dat_pred$ano == 2024], dat_pred$quantidade[dat_pred$ano ==
2024],
     type = 'o', col = "black", pch = 16, lty = 1)
lines(dat_pred$mes[dat_pred$ano == 2025], dat_pred$quantidade[dat_pred$ano ==
2025],
     type = 'o', col = "red", pch = 17, lty = 1)
lines(dat_pred$mes[dat_pred$ano == 2026], dat_pred$quantidade[dat_pred$ano ==
2026],
     type = 'o', col = "blue", pch = 18, lty = 1)
legend("topright",
      legend = as.character(2024:2026),
      col = col_values,
      pch = pch_values,
      lty = rep(1, 3),
      cex = 0.8,
      title = "Years")
par(mfrow = c(1, 1))
```

40

```
### Plot graphs separated by year and month

df1 = dat_total %>% mutate(Meses = rep(seq(1:12),23))
df = df1 %>% select(mes = Meses,
                    ano = Ano,
                    quantidade = Casos)
```

41

```
### Set the figure layout to 5×5 (23 graphs) and Loop to create the graph for each year
```

```
par(mfrow = c(5, 5), mar = c(4, 4, 2, 1))
for (ano in 2001:2023) {
  plot(df$mes[df$ano == ano], df$quantidade[df$ano == ano],
       type = 'o',
       xlab = "Months",
       ylab = "Cases of tuberculosis",
       xlim = c(1, 12),
       ylim = c(min(df$quantidade), max(df$quantidade)),
       col = "blue",
       pch = 16,
       main = paste("Tuberculosis -", ano),
       xaxt = "n"
  )
  axis(1, at = 1:12, labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))
}
par(mfrow = c(1, 1))
```

Protocol references

Svetunkov I (2024). _smooth: Forecasting Using State Space Models_. R package version 4.0.2, https://CRAN.R-project.org/package=smooth.
Wickham H, François R, Henry L, Müller K, Vaughan D (2023)._dplyr: A Grammar of Data Manipulation_. R package version 1.1.4, https://CRAN.R-project.org/package=dplyr.
Schauberger P, Walker A (2024). _openxlsx: Read, Write and Edit xlsx Files_. R package version 4.2.7,https://CRAN.R-project.org/package=openxlsx.
Cui B (2024). _DataExplorer: Automate Data Exploration and Treatment_. R package version 0.8.3, https://CRAN.R-project.org/package=DataExplorer.
Wickham H, Vaughan D, Girlich M (2024). _tidyr: Tidy Messy Data_. R package version 1.3.1,https://CRAN.R-project.org/package=tidyr.
H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL **https://www.jstatsoft.org/v40/i03/**.
Bache S, Wickham H (2022)._magrittr: A Forward-Pipe Operator for R_. R package version 2.0.3, https://CRAN.R-project.org/package=magrittr.
Pedersen T (2024). _patchwork: The Composer of Plots_. R package version 1.3.0, <https://CRAN.R-project.org/package=patchwork>.
Wilke C (2024). _cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'_. R package version 1.1.3, https://CRAN.R-project.org/package=cowplot.
Trapletti A, Hornik K (2024). _tseries: Time Series Analysis and Computational Finance_. R package version 0.10-57, https://CRAN.R-project.org/package=tseries.
McLeod A (2022). _Kendall: Kendall Rank Correlation and Mann-Kendall Trend Test_. R package version 2.2.1, https://CRAN.R-project.org/package=Kendall.
Ulrich J (2023). _TTR: Technical Trading Rules_. R package version 0.24.4, https://CRAN.R-project.org/package=TTR.
Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2024). _forecast: Forecasting functions for time series and
linear models_. R package version 8.23.0,**https://pkg.robjhyndman.com/forecast/**.
Hyndman RJ, Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R." _Journal of Statistical Software_, *27*(3), 1-22. doi:10.18637/jss.v027.i03 https://doi.org/10.18637/jss.v027.i03.
Boshnakov GN, Halliday J (2024). _sarima: Simulation and Prediction with Seasonal ARIMA Models_. R package version 0.9.3, https://CRAN.R-project.org/package=sarima.
Sax C, Eddelbuettel D (2018). "Seasonal Adjustment by X-13ARIMA-SEATS in R." _Journal of Statistical
Software_, *87*(11), 1-17. doi:10.18637/jss.v087.i11 https://doi.org/10.18637/jss.v087.i11.
Wickham H, Pedersen T, Seidel D (2023). _scales: Scale Functions for Visualization_. R package version 1.3.0, https://CRAN.R-project.org/package=scales.
Wickham H (2023). _stringr: Simple, Consistent Wrappers for Common String Operations_. R package version 1.5.1, https://CRAN.R-project.org/package=stringr.

Acknowledgements

Acknowledgements to DataSUS and the Notification of Diseases Information System (SINAN) for providing the publicly available and unrestricted database on tuberculosis cases in Goiás, which was essential for conducting this study. Access to this information enabled the development of an algorithm for building a prediction model using time series, which can provide estimates of tuberculosis cases in this region and contribute to the planning of healthcare actions aimed at controlling this disease.