LABORATORY OF APPLIED MATHEMATICAL PROGRAMMING AND STATISTICS

LAMPS | PUC RIO

JuMP-Dev 2024

Time series modeling via JuMP

Professor: Davi Valladão

Students:

PhD: André Ramos, Marina Dietze, Tomás Gutierrez, Matheus Alves

MSc: Luiz Fernando, Matheus Nogueira

UG: Luisa Compasso, João Couto

lampsco

stonelab

Objective and contributions

- The main objective of this project is to implement traditional time series models in Julia, harnessing the power of JuMP to provide access to state-of-the-art solvers, explores different modeling choices and extend existing methodologies via novel optimization techniques.

- Contributions:
  - **SARIMAX.jl**
  - **UnobservedComponetsGAS.jl**
  - **StateSpaceLearning.jl**

Our packages

- **SARIMAX.jl**
  - Implements the auto SARIMA model via modern optimization techniques.

- **UnobservedComponetsGAS.jl**
  - Estimate a conditional time-varying dynamic for a given probabilistic distribution parameters.

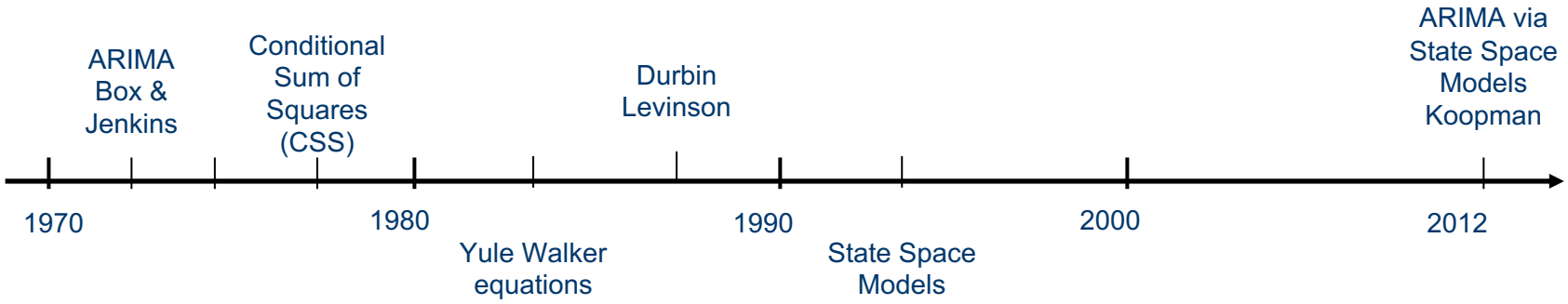
- **StateSpaceLearning.jl**
  - Estimate a structural time series model via a high-dimensional regression.

SARIMAX.jl

- Given ARMA(p,q)

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t\,, \qquad \varepsilon_t \sim N(0, \sigma^2)$$

- Parameter estimation: existing approaches
- Maximum likelihood via Kalman filtering



SARIMAX.jl

- Under the optimization lens:

$$\min_{c,\boldsymbol{\Phi},\boldsymbol{\Theta},\varepsilon} \sum_{t=1}^{T} \varepsilon_t^2$$

$$s.t: \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

SARIMAX.jl

- Under the optimization lens:

$$\min_{c,\boldsymbol{\Phi},\boldsymbol{\Theta},\varepsilon} \sum_{t=1}^{T} \varepsilon_t^2$$

$$s.t: \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

- JuMP.jl formulation

```
m = Model(Ipopt.Optimizer)
@variable(m, c)
@variable(m, θ[1:q])
@variable(m, φ[1:p])
@variable(m, ε[1:T])
@constraint(m, [t = (max(p, q) + 1):T], ε[t] == y[t] - (c + sum(φ[j]*y[t-j]
                                for j in 1:p) + sum(θ[j]*ε[t-j] for j in 1:q)))
@objective(m, Min, sum((ε[t])^2 for t in max(p, q) + 1:T))
optimize!(m)
```

SARIMAX.jl

- Package usage

```
model = SARIMA(trainingSet, 1, 0, 1;
               seasonality=12, P=0, D=1, Q=2, allowMean=false)
fit!(model; objectiveFunction="mse")
forecast = predict!(model, stepsAhead)
scenarios = simulate(model, stepsAhead, 200)
```

- Controled experiment: 1000 random ARMA stationary processes

Percentage of best estimation metrics						
Model	**AIC**	**BIC**	**AICc**	**MSE**	**Log Likel.**	**Time**
Statsmodels, pmdarima	0.9	0.9	0.9	0.8	0.7	0.0
StateSpaceModels.jl	1.0	1.0	1.0	0.8	0.9	20.7
Rforecast	0.9	0.9	0.9	1.2	1.2	64.1
SARIMAX.jl (MSE)	**89.1**	**89.1**	**89.1**	**89.1**	**89.1**	**15.1**
Tied models	8.0	8.0	8.0	8.0	8.0	0.0

Percentage of erros during estimation	
Model	**(%)**
Statsmodels	0.0
Pmdarima	0.0
StateSpaceModels.jl	9.8
Rforecast	0.6
SARIMAX.jl (MSE)	**0.0**

Possible extensions

$$\underset{c,\phi_t,\theta_t\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$

Add explanatory variables

$$\underset{c,\Phi,\Theta,\beta,\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{m} \beta_i X'_{ti} + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in \{1, \cdots, T\}$$

Change objective function

$$\underset{c,\phi_t,\theta_t\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} |\epsilon_t|$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$

Add regularization

$$\underset{c,\phi_t,\theta_t,\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2 + \lambda \left(\sum_{i=1}^{N} |\Psi_i| \right)$$

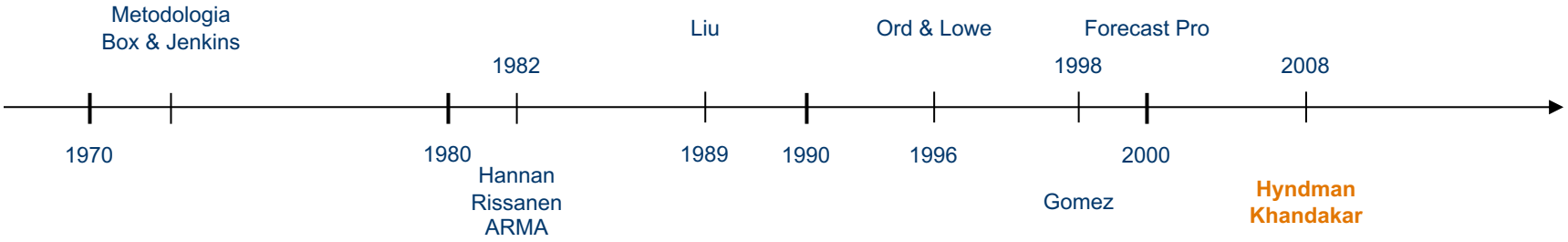
$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$

$$\Psi = \{\phi, \theta\}$$

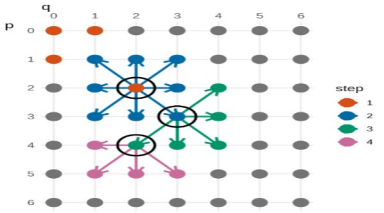
Possible extensions

$$\underset{c,\phi_t,\theta_t\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$



Specification: Auto-ARIMA (Hyndman-Khandakar)

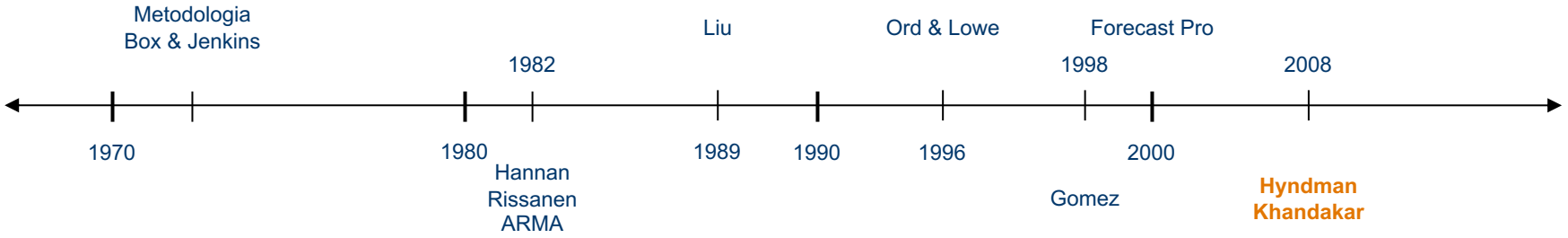


Optimal Specification/Estimation: MINLP

Possible extensions

$$\underset{c,\phi_t,\theta_t\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$



Optimal Specification/Estimation: MINLP

$$\underset{c,\phi_t,\theta_t\epsilon_t}{\text{minimize}} \quad \sum_{t=1}^{T} \epsilon_t^2$$

$$\text{subject to} \quad y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \quad \forall t \in 1, \cdots, T$$

$$||\Psi||_0 \leq K$$

Advantages

- State-of-the-art optimization algorithms through modern solvers
  - Gurobi;
  - Ipopt;
  - SCIP.
- Possibility to use global solvers, like
  - Alpine;
  - Eago;
- Access to a wide range of optimization resources, including:
  - Flexibility to add necessary restrictions;
  - Regularization terms in the objective function;
  - Initialization techniques, e.g. warm-start;
  - Mixed Integer Programming (MIP);
  - Robust optimization techniques.

UnobservedComponentsGAS.jl

- Estimates a conditional time-varying dynamic for a given probabilistic distribution parameters

$$
\begin{aligned}
y_t &\sim p(y_t \mid \mu_{t|t-1}; \boldsymbol{Y_{t-1}}, \sigma, \theta) = N(\mu_{t|t-1}, \sigma) \\
\mu_{t+1|t} &= m_{t+t|t} \\
m_{t+1|t} &= m_{t|t-1} + b_{t|t-1} + \kappa_m s_t \\
b_{t+1|t} &= b_{t|t-1} + \kappa_b s_t
\end{aligned}
$$

- Optimization problem

$$
\max_{\substack{\boldsymbol{\mu},\boldsymbol{m},\boldsymbol{b},\kappa_m,\kappa_b,\sigma, \\ \boldsymbol{s}(\boldsymbol{\mu},\sigma,\boldsymbol{y})}} \sum_{t=1}^{T} \left(-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(y_t - \mu_{t|t-1})^2}{2\sigma^2} \right)
$$

$$
\begin{aligned}
s.t.: \quad & \mu_{t+1|t} = m_{t+t|t} && \forall t = 1, \dots, T \\
& m_{t+1|t} = m_{t|t-1} + b_{t|t-1} + \kappa_m s_t && \forall t = 1, \dots, T \\
& b_{t+1|t} = b_{t|t-1} + \kappa_b s_t && \forall t = 1, \dots, T \\
& \sigma > 0 \\
& \mathrm{LB} \leq \kappa_m \leq UB \\
& \mathrm{LB} \leq \kappa_b \leq UB
\end{aligned}
$$

UnobservedComponentsGAS.jl

- Currently available distributions:
  - Gaussian
  - LogNormal
  - t-Location-Scale

- More continuous and discrete distributions will be implemented.

- Currently available components structures:
  - Level: random walk, random walk with slope, $AR(1)$
  - Seasonality with trigonometric functions: deterministic and stochastic
  - Autoregressive of order p
  - Explanatory variables

UnobservedComponentsGAS.jl

- Package usage

```
dist = UnobservedComponentsGAS.NormalDistribution()
d    = 0.0

params      = [true, false]
level       = ["random walk slope", ""]
seasonality = ["deterministic 12", ""]
ar          = [missing, missing]
gas_model   = UnobservedComponentsGAS.GASModel(dist, params, d, level,
                                               seasonality, ar)

fitted_model = UnobservedComponentsGAS.fit(gas_model, y)
forec        = UnobservedComponentsGAS.predict(gas_model, fitted_model,   y,
                                               steps_ahead,  num_scenarious)
```

StateSpaceLearning.jl

- The Basic Structural Model

$$
\begin{aligned}
y_t &= \mu_t + \gamma_t + \varepsilon_t, & \varepsilon_t &\sim N(0, \sigma_\varepsilon^2) \\
\mu_t &= \mu_{t-1} + \nu_t + \xi_t, & \xi_t &\sim N(0, \sigma_\xi^2) \\
\nu_t &= \nu_{t-1} + \zeta_t, & \zeta_t &\sim N(0, \sigma_\zeta^2) \\
\gamma_t &= -\sum_{j=1}^{s-1} \gamma_{t-j} + \omega_t, & \omega_t &\sim N(0, \sigma_\omega^2)
\end{aligned}
$$

- Can be written as a high-dimensional regression:

$$
\mathrm{y_t} = \mu_0 + t\nu_0 + \sum_{\tau=1}^{t-1}(t-\tau+1)\zeta_\tau + \sum_{\tau=1}^{t-1}\xi_\tau + \gamma_{\tau-s}M_t + \sum_{j=1}^{M_t}\left(\omega_{t-js} - \omega_{t-js-1}\right) + \varepsilon_t^*
$$

StateSpaceLearning.jl

- For an illustrative example, consider the local level model:

$$y_t = \mu_t + \epsilon_t$$
$$\mu_t = \mu_{t-1} + \eta_t$$

- By iterating the level component we obtain:

$$y_t = \mu_1 + \sum_{\tau=2}^{t} \eta_\tau + \epsilon_t$$

- This equation can be written in a vectorized form:

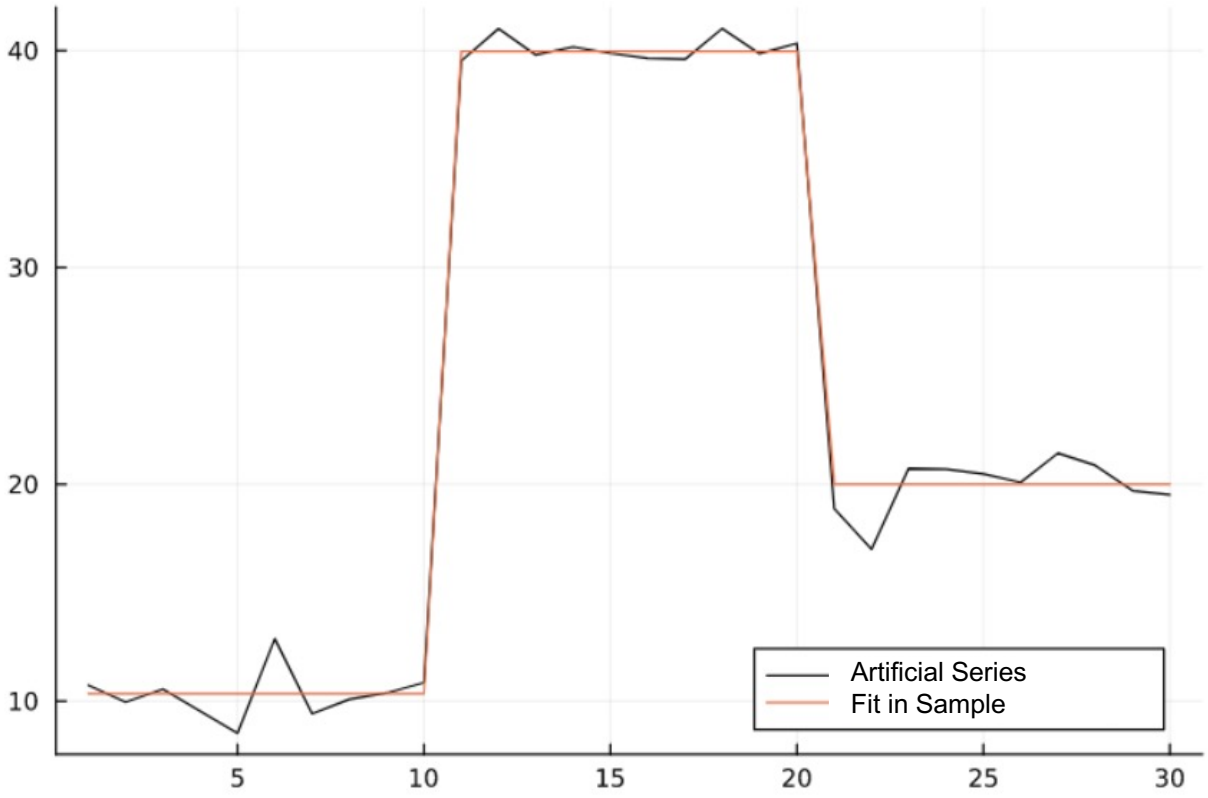
$$\boldsymbol{y} = \mu_1 + \boldsymbol{X\eta} + \epsilon,$$

- Where y = (y1, . . . , yT), η = (η2, . . . , ηT) and:

$$X = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 0 \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

StateSpaceLearning.jl

- Result of a Lasso estimation for the local level model in regression form :



StateSpaceLearning.jl

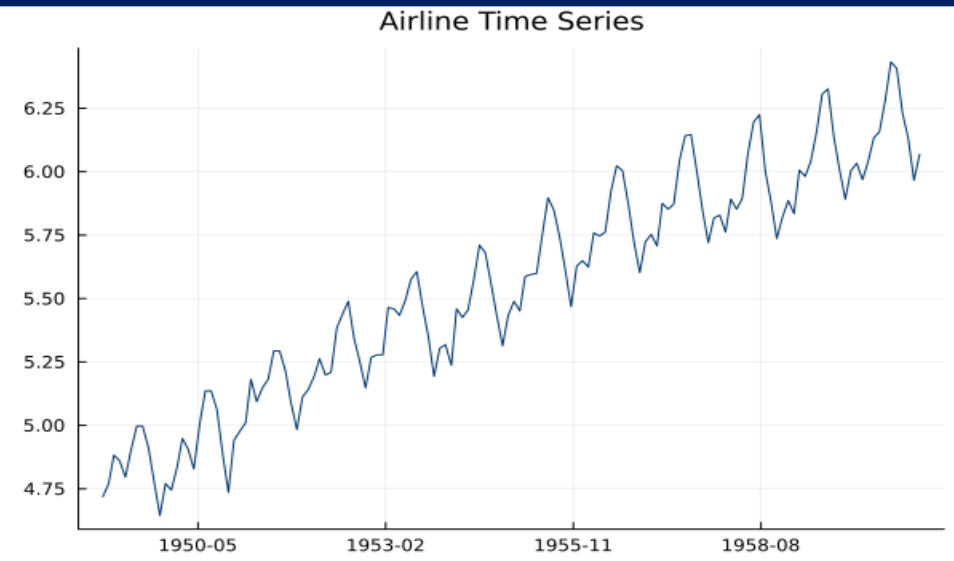
- In principle we fit a regularized regression (“AdaLasso”) using “coordinate descent” algorithm.
- To explore other regularization techniques such as the norm L0, we will rely on JuMP to solve the optimization model.

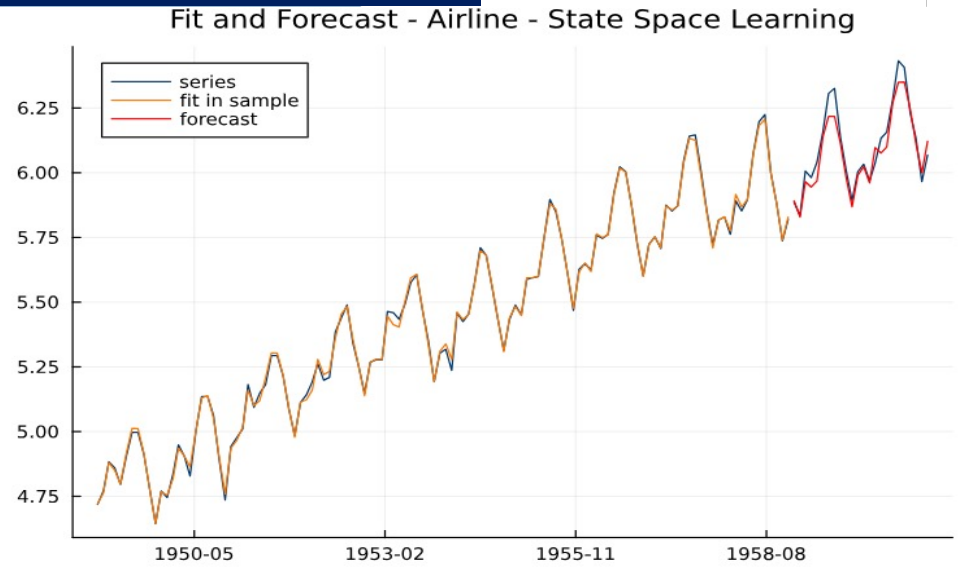
$$\min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|$$

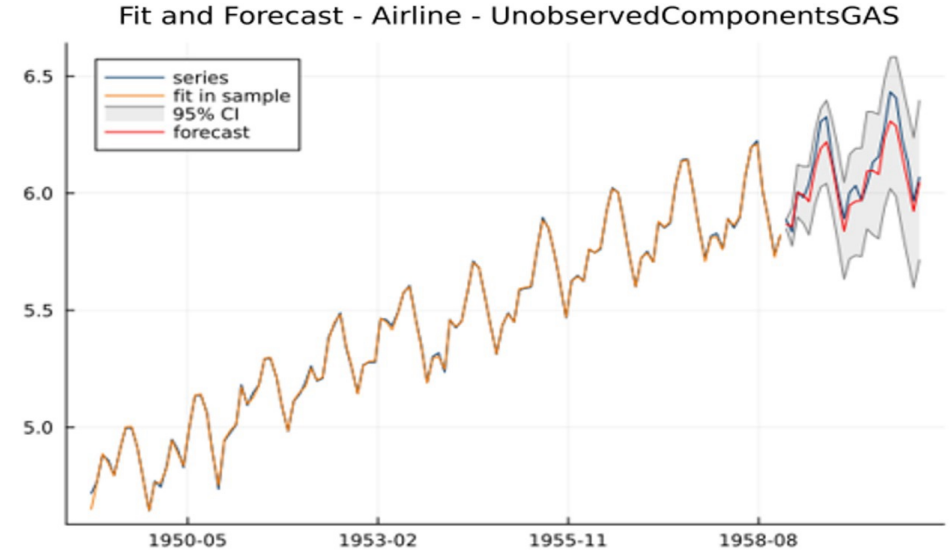
Package usage:

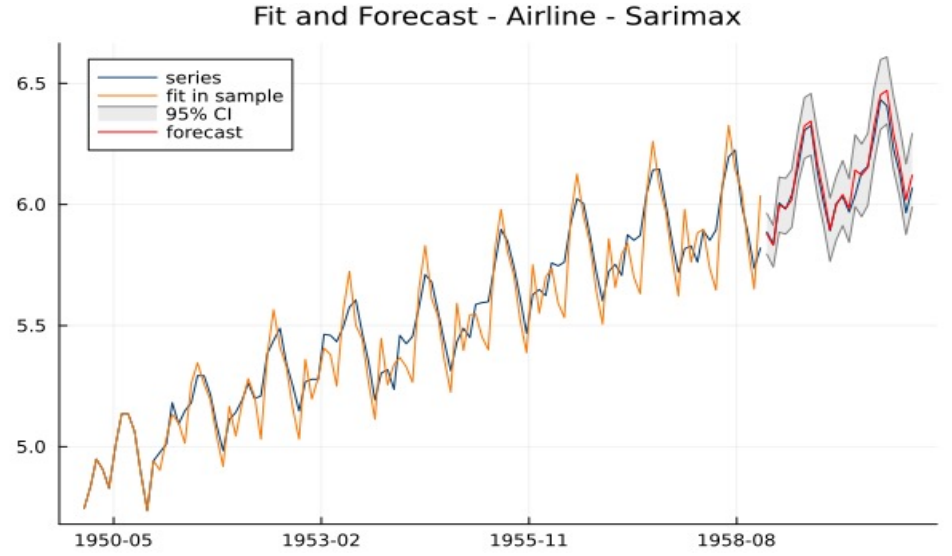
```
output = StateSpaceLearning.fit_model(y)
prediction = StateSpaceLearning.forecast(output, steps_ahead)
```

Comparison


Airline Time Series
6.25
6.00
5.75
5.50
5.25
5.00
4.75
1950-05
1953-02
1955-11
1958-08


Fit and Forecast - Airline - State Space Learning
series
fit in sample
forecast
6.25
6.00
5.75
5.50
5.25
5.00
4.75
1950-05
1953-02
1955-11
1958-08


Fit and Forecast - Airline - UnobservedComponentsGAS
series
fit in sample
95% CI
forecast
6.5
6.0
5.5
5.0
1950-05
1953-02
1955-11
1958-08


Fit and Forecast - Airline - Sarimax
series
fit in sample
95% CI
forecast
6.5
6.0
5.5
5.0
1950-05
1953-02
1955-11
1958-08

Numerical Results

- Results for out of sample forecasting accuracy for 48000 monthly M4 time series competition

MASE								
Models	Mean				Median			
	Short	Medium	Long	Total	Short	Medium	Long	Total
autoARIMA Python	0.63	0.969	1.24	0.947	0.488	0.719	0.909	0.745
StateSpaceModels.jl	0.662	1.031	1.34	1.011	0.504	0.741	0.929	0.765
SARIMAX.jl	0.632	0.963	1.229	0.942	0.478	0.708	0.891	0.733
UnobserverdComponentsGAS.jl	1.717	4.161	6.738	4.205	0.679	0.883	1.106	0.924
StateSpaceLearning.jl	0.654	0.96	1.194	0.936	0.501	0.695	0.845	0.72

Thank You!!