

Time Series Modeling using R

Dr. Kapil Kumar

Associate Professor, Department of Statistics
Chaudhary Charan Singh University, Meerut, India
kapilstats@gmail.com

One-week online Faculty Development Programme on
*Advanced Statistical Software, Methodologies, and Data Visualization for
Research Excellence (23–27 June 2025)*

Department of Mathematics & Statistics, Manipal University Jaipur, in
collaboration with the Indian Association for Reliability and Statistics (IARS)

26 June 2025 · National Statistics Day 2025 (online)

Plan of the session

1. Getting set up in R
2. What makes data a time series
3. Warm-up: a simulated sales series
4. Case study: Air India domestic passengers
5. Modelling workflow and take-aways

By the end you can build a `ts` object, decompose it, test for a unit root, identify and fit a seasonal ARIMA model, check its residuals, and produce forecasts with prediction intervals.

PART 1

Getting set up in R

Packages used in this session

- **forecast:** `ma()`, `auto.arima()`, `checkresiduals()`, `forecast()`
- **tseries:** `adf.test()` for unit roots
- **zoo:** `rollmean()`, `na.approx()` for ordered data
- **ggplot2:** publication-quality graphics

```
# install once
# install.packages(c("forecast", "tseries",
#                    "zoo", "ggplot2"))
library(forecast)
library(tseries)
library(zoo)
library(ggplot2)
col_main <- "#A9471F"   # plot colours
col_alt  <- "#2F5D80"
```

Working folder, reading and writing data

- Keep the script and a data/ folder together (an RStudio project); avoid hard-coded paths such as D:/OneDrive/...
- `read.csv()` reads a comma-separated file into a data frame
- `write.csv()` saves a cleaned copy for reuse

```
# getwd() shows where R reads and writes
air <- read.csv("data/airindia_domestic_DGCA.csv")
dim(air)

## [1] 196    4

head(air[, 1:3], 3)

##   year month passengers
## 1 2009     1     33763
## 2 2009     2     41170
## 3 2009     3     50376

write.csv(air, "data/air_copy.csv",
          row.names = FALSE)
```

PART 2

What makes data a time series

Time series: definition and notation

- A **time series** is a sequence of observations $y_1, y_2, \dots, y_n$ recorded at regular time points; the order of the observations carries information.
- The **frequency** m is the number of observations per seasonal cycle: $m = 12$ for monthly, $m = 4$ for quarterly, $m = 7$ for daily data with a weekly cycle.
- Observations are **dependent**: today's value is informative about tomorrow's.

$$\text{Autocorrelation at lag } k: \quad \rho_k = \frac{\text{Cov}(y_t, y_{t-k})}{\text{Var}(y_t)}, \quad r_k = \frac{\sum_{t=k+1}^n (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^n (y_t - \bar{y})^2}$$

Examples: daily stock prices, monthly rainfall, quarterly GDP, monthly airline passengers.

Stationarity

A series is (weakly) stationary if, for all t ,

$$\begin{aligned} E(y_t) &= \mu, & \text{Var}(y_t) &= \sigma^2, \\ \text{Cov}(y_t, y_{t-k}) &= \gamma_k \text{ (depends on } k \text{ only).} \end{aligned}$$

Trends, changing variance and seasonality all break stationarity.

Why it matters ARMA models describe *stationary* behaviour. A non-stationary series is first transformed (logs, differences) until it is stationary, then modelled.

Common fix First difference: $\nabla y_t = y_t - y_{t-1}$. Seasonal difference: $\nabla_{12} y_t = y_t - y_{t-12}$.

Components of a time series

- **Trend** T_t : long-run rise or fall
- **Seasonal** S_t : pattern repeating every m periods (festivals, holidays, weather)
- **Cycle**: rises and falls without a fixed period (business cycles)
- **Irregular** E_t : what is left over

Additive: $y_t = T_t + S_t + E_t$

Multiplicative: $y_t = T_t \times S_t \times E_t$

Use the multiplicative form (or model $\log y_t$) when the seasonal swing grows with the level.

Uses understanding structure, detecting anomalies, and forecasting future values for planning.

PART 3

Warm-up: a simulated sales series

Simulating nine years of monthly sales

$$y_t = 100 + 10t + \varepsilon_t$$
$$\varepsilon_t \sim N(0, 50^2), \quad t = 1, \dots, 108$$

A linear trend plus noise, with no seasonality. `ts()` attaches the time index: frequency = 12 makes it monthly.

```
set.seed(123)
y <- 100 + 10 * (1:108) + rnorm(108, mean = 0, sd = 50)
sales <- ts(y, start = c(2016, 1), frequency = 12)
window(sales, end = c(2016, 6))
```

##		Jan	Feb	Mar	Apr	May	Jun
##	2016	81.98	108.49	207.94	143.53	156.46	245.75



Smoothing with a moving average

Centred MA of order 5:

$$\hat{T}_t = \frac{1}{5} \sum_{j=-2}^2 y_{t+j}$$

Averaging neighbouring months removes short-term noise and reveals the trend. For even orders `ma()` uses a centred $2 \times m$ average.

```
sales_ma <- ma(sales, order = 5)
cols <- c(Sales = "grey60", "MA(5)" = col_main)
autoplot(sales, series = "Sales") +
  autolayer(sales_ma, series = "MA(5)") +
  scale_colour_manual(values = cols) +
  labs(x = NULL, y = "Sales", colour = NULL)
```



Testing for a unit root: the ADF test

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^p \delta_i \Delta y_{t-i} + e_t$$

$H_0 : \gamma = 0$ (unit root) vs $H_1 : \gamma < 0$

`adf.test()` includes a constant and a trend, so rejecting H_0 means *stationary around a linear trend*.

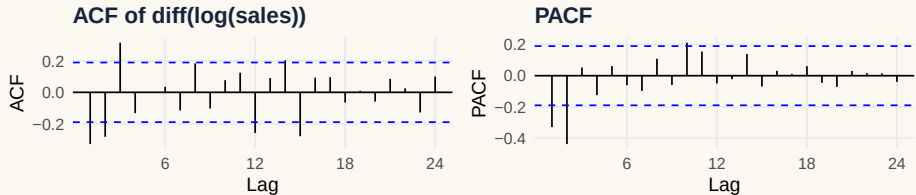
```
adf.test(sales)
```

```
##  
## Augmented Dickey-Fuller Test  
##  
## data: sales  
## Dickey-Fuller = -4, Lag order = 4, p-value =  
## 0.01  
## alternative hypothesis: stationary
```

Reading it $p = 0.014 < 0.05$: no unit root. The series is trend-stationary, which matches how we simulated it.

Differencing, then ACF and PACF

```
d1 <- diff(log(sales))      # growth rate: removes the trend
ggAcf(d1) + ggtitle("ACF of diff(log(sales))") -> p1
ggPacf(d1) + ggtitle("PACF") -> p2
gridExtra::grid.arrange(p1, p2, ncol = 2)
```



Bars inside the $\pm 1.96/\sqrt{n}$ band are consistent with zero autocorrelation. The negative lag-1 spike is the footprint of over-differencing a trend-stationary series.

PART 4

Case study: Air India domestic passengers

The data

- Monthly **passengers carried** by Air India on scheduled domestic services, January 2009 to December 2024
- Source: Directorate General of Civil Aviation (DGCA), *Monthly Traffic and Operating Statistics*
- Jan–Apr 2025 is held back to check the forecasts

Data note for this version Values were re-extracted from the DGCA annual tables. The figures for 2015 and for 2020 used in the live session have been corrected. April 2020 is recorded as zero: scheduled domestic flights were suspended from 25 March to 24 May 2020.

Reading the data and building a date

```
air <- read.csv("data/airindia_domestic_DGCA.csv")
air$date <- as.Date(paste(air$year, air$month, 1, sep = "-"))
sum(is.na(air$passengers))           # any missing values?

## [1] 0

subset(air, year == 2020 & month <= 5, select = c(date, passengers))

##           date passengers
## 133 2020-01-01    1285138
## 134 2020-02-01    1283574
## 135 2020-03-01     722474
## 136 2020-04-01         0
## 137 2020-05-01     41595
```

Missing is not the same as zero

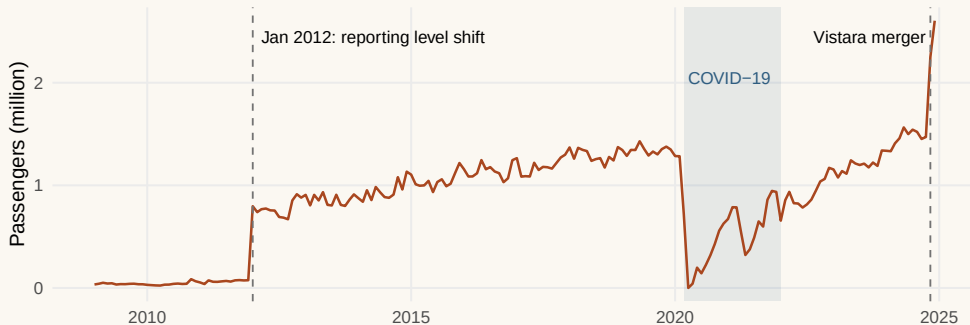
- `zoo::na.approx()` fills gaps by linear interpolation. It is reasonable when a value was *not recorded* but activity carried on.
- In April 2020 no scheduled domestic flights operated, so the true value is 0. Interpolating would invent about 382 thousand passengers.
- Decide from *domain knowledge*, then document the decision.

```
x <- c(722474, NA, 41595) # Mar-May 2020
na.approx(x) # what interpolation gives

## [1] 722474 382034 41595
```

Here No values are missing;
April 2020 is kept as 0.

The full series: three structural breaks



2009–2011 values are about 0.04 million a month in the DGCA tables, then about 0.8 million from January 2012. We therefore model **January 2012 onwards**, where the series is measured consistently.

Creating the time series object

```
full <- ts(air$passengers[air$year <= 2024], start = c(2009, 1), frequency = 12)
y <- window(full, start = c(2012, 1))      # analysis window
y <- y / 1e6                               # millions, for readable output
round(window(y, end = c(2013, 12)), 2)    # first two years
```

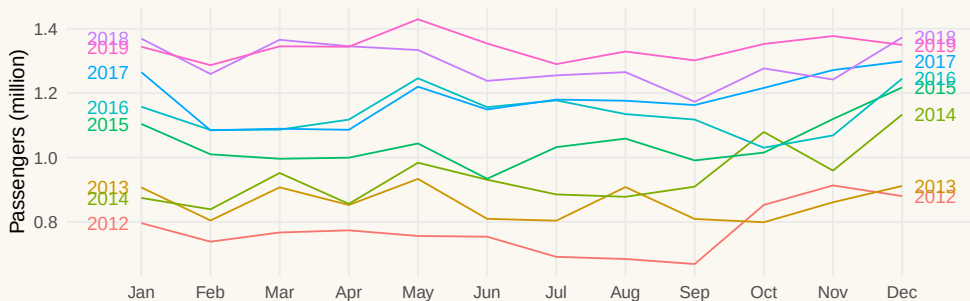
```
##           Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2012 0.80 0.74 0.77 0.77 0.76 0.75 0.69 0.68 0.67 0.85 0.91 0.88
## 2013 0.91 0.80 0.91 0.85 0.93 0.81 0.80 0.91 0.81 0.80 0.86 0.91
```

```
frequency(y); start(y); end(y)
```

```
## [1] 12
## [1] 2012    1
## [1] 2024   12
```

Seasonality: which months are busy?

```
yc <- window(y, end = c(2019, 12))           # pre-COVID years only
ggseasonplot(yc, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = NULL, x = NULL, y = "Passengers (million)") + theme(legend.position = "none")
```



A 12-month moving average shows the trend

```
trend12 <- ma(y, order = 12) # centred 2x12 MA
cols <- c(Monthly = "grey60", "2x12 MA" = col_main)
autoplot(y, series = "Monthly") +
  autolayer(trend12, series = "2x12 MA") +
  scale_colour_manual(values = cols) +
  labs(x = NULL, y = "Million", colour = NULL)
```



Centred 2×12 moving average:

$$\hat{T}_t = \frac{1}{12} \sum_{j=-5}^5 y_{t+j} + \frac{1}{24} (y_{t-6} + y_{t+6})$$

The half weights at the two ends balance an even-length window around month t , so each calendar month gets equal weight and seasonality is removed.

Classical additive decomposition

```
dec <- decompose(y, type = "additive")  
autoplot(dec) + labs(title = NULL, x = NULL)
```



Reading the decomposition

```
seas <- round(dec$figure * 1000) # thousands
names(seas) <- month.abb
seas

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 81 40 37 -67 -21 -77 -71 -39 -61 17 45 116
```

- Peaks in **December** (116 thousand above trend) and **January**: the winter holiday season
- Troughs in **June–July** and **April**
- The trend carries the COVID collapse and the recovery
- The remainder is large in 2020–21: no fixed seasonal pattern fits a lockdown

Is the series stationary?

```
adf.test(y)$p.value      # levels
## [1] 0.9465

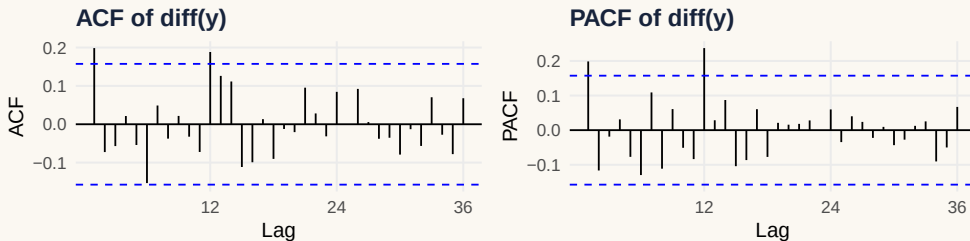
dy <- diff(y)            # first differences
adf.test(dy)$p.value
## [1] 0.01
```

Conclusion Levels: $p = 0.95$, a unit root cannot be rejected. First differences: $p \leq 0.01$, stationary. So $d = 1$.



Correlograms of the differenced series

```
p1 <- ggAcf(dy, lag.max = 36) + ggtitle("ACF of diff(y)")  
p2 <- ggPacf(dy, lag.max = 36) + ggtitle("PACF of diff(y)")  
gridExtra::grid.arrange(p1, p2, ncol = 2)
```



Spikes at lag 1 and at lag 12 suggest a non-seasonal MA(1) term and a seasonal term; the model search on the next slides confirms this.

Seasonal ARIMA in one line

$$\text{ARIMA}(p, d, q)(P, D, Q)_m : \quad \phi(B) \Phi(B^m) (1 - B)^d (1 - B^m)^D y_t = c + \theta(B) \Theta(B^m) \varepsilon_t$$

- B is the backshift operator: $By_t = y_{t-1}$
- ϕ, Φ : AR polynomials (orders p, P)
- θ, Θ : MA polynomials (orders q, Q)
- d, D : number of ordinary and seasonal differences

Identification rules of thumb ACF
cuts off after lag $q \Rightarrow \text{MA}(q)$.
PACF cuts off after lag $p \Rightarrow \text{AR}(p)$.
Spikes at lags 12, 24 $\Rightarrow$ seasonal terms.

Automatic model selection

```
fit <- auto.arima(y)
fit

## Series: y
## ARIMA(0,1,1)(0,0,1)[12]
##
## Coefficients:
##          ma1    sma1
##       0.247  0.216
## s.e.  0.080  0.082
##
## sigma^2 = 0.016:  log likelihood = 101.2
## AIC=-196.5   AICc=-196.3   BIC=-187.3
```

- `auto.arima()` picks d by KPSS tests, then searches p, q, P, Q to minimise AICc (Hyndman–Khandakar algorithm)

Selected: ARIMA(0, 1, 1)(0, 0, 1)₁₂

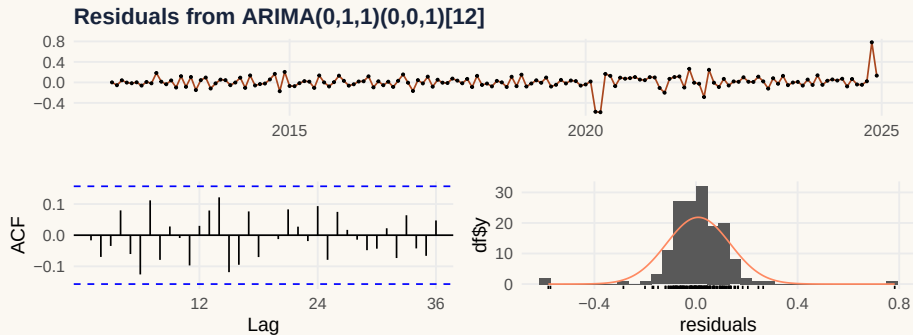
$$(1 - B)y_t = (1 + \theta_1 B)(1 + \Theta_1 B^{12})\varepsilon_t$$

$$\hat{\theta}_1 = 0.247, \quad \hat{\Theta}_1 = 0.216$$

A random walk with MA(1) and seasonal MA(1) errors and no drift, so the forecasts stay close to the last observed level.

Residual diagnostics

```
checkresiduals(fit)
```



Ljung-Box test for white-noise residuals

$$Q = n(n+2) \sum_{k=1}^h \frac{r_k^2}{n-k} \sim \chi_{h-K}^2$$

r_k : residual autocorrelations; K : number of estimated ARMA coefficients. H_0 : residuals are white noise.

```
Box.test(residuals(fit), lag = 24,  
         type = "Ljung-Box",  
         fitdf = length(coef(fit)))
```

```
##  
## Box-Ljung test  
##  
## data: residuals(fit)  
## X-squared = 23, df = 22, p-value = 0.4
```

Reading it $p = 0.4 > 0.05$: no remaining autocorrelation. But the residual spread is far from constant (COVID months), so read the intervals with care.

Forecasting the next 12 months

```
fc <- forecast(fit, h = 12, level = c(80, 95))
autoplot(window(y, start = 2018)) +
  autolayer(fc, colour = col_main) + labs(title = NULL, x = NULL, y = "Passengers (million)")
```



How good was the forecast? Checking against 2025

```
actual <- air$passengers[air$year == 2025] / 1e6
check <- data.frame(month = month.abb[1:4],
                    point = fc$mean[1:4],
                    lo95 = fc$lower[1:4, 2],
                    hi95 = fc$upper[1:4, 2],
                    actual = actual)
round(check[, -1], 2)
```

##	point	lo95	hi95	actual
## 1	2.63	2.39	2.88	2.42
## 2	2.64	2.24	3.03	2.41
## 3	2.65	2.15	3.16	2.30
## 4	2.66	2.07	3.25	2.18

- All four actual values fall **inside** the 95% intervals
- But every point forecast is **too high**: the model carries the post-merger December peak forward as the new level
- Mean absolute percentage error: 13.9%

What a careful analyst would try next

Handle the breaks explicitly Add intervention dummies for the lockdown months and a step dummy from November 2024, e.g.
`auto.arima(y, xreg = X).`

Compare models honestly Hold out the last 12 months, or use time-series cross-validation with `tsCV()`, and compare ARIMA with ETS (`ets()`) on out-of-sample error.

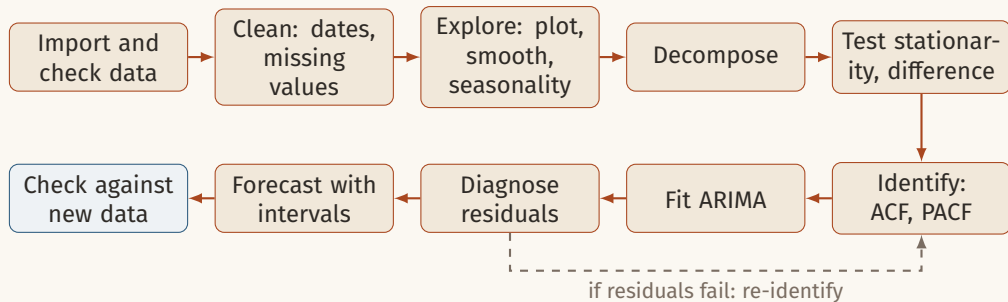
Stabilise the variance Model $\log y_t$ (or a Box-Cox transform) so that seasonal swings are proportional to the level.

Keep in mind A model fitted to the past cannot foresee a merger or a pandemic. Report forecasts with their intervals and their assumptions.

PART 5

Modelling workflow and take-aways

The workflow we followed



Key take-aways

1. **Look before you model.** Plots revealed a reporting break, a pandemic and a merger that no test would have named.
2. **Check the data against its source.** Two years of the original file did not match DGCA.
3. **Missing is not zero**, and zero is not missing: decide from domain knowledge.
4. **Stationarity first:** ADF on levels and on differences fixes d .
5. **Diagnose, then forecast:** white-noise residuals are necessary, not sufficient.
6. **Judge forecasts on data the model has not seen.**

References

- Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*, 5th Ed., Wiley, Hoboken.
- Cryer, J. D., and Chan, K.-S. (2008). *Time Series Analysis: With Applications in R*, 2nd Ed., Springer, New York.
- Hyndman, R. J., and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*, 2nd Ed., OTexts, Melbourne.
- Hyndman, R. J., and Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 27(3), 1–22.
- Ljung, G. M., and Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2), 297–303.
- Said, S. E., and Dickey, D. A. (1984). Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3), 599–607.
- Data: Directorate General of Civil Aviation, Government of India. *Monthly Traffic and Operating Statistics: Air India, Scheduled Domestic Services*, 2009–2025.

Thank you

Questions and discussion