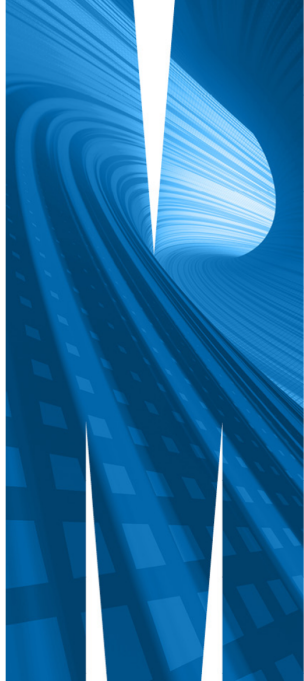


25 years of open source forecasting software

Rob J Hyndman

26 June 2025



Outline

1

R

2

Python

3

Julia

Outline

1 R

2 Python

3 Julia

Early R forecasting (c.2000)

ts package (now stats package):

- `HoltWinters()`: point forecasts only, with optional multiplicative seasonality (written by David Meyer).
- `arima()`: state space formulation of ARIMA models (written by Brian Ripley).
- `structTS()`: Basic structural models as per Harvey (written by Brian Ripley).

Early R forecasting (c.2000)

ts package (now stats package):

- `HoltWinters()`: point forecasts only, with optional multiplicative seasonality (written by David Meyer).
 - `arima()`: state space formulation of ARIMA models (written by Brian Ripley).
 - `structTS()`: Basic structural models as per Harvey (written by Brian Ripley).
- Each had a `predict()` method, but output was inconsistent.
 - `HoltWinters` did not produce prediction intervals.

forecast package for R

- Consistent output for existing methods by introducing new S3 generic `forecast()` and new S3 class `forecast`.
- Add new methods including `ets()`, `thetaf()`, `auto.arima()`.
- Modelling functions should be able to be swapped while leaving code unchanged.
- Easy plotting tools with new `plot.forecast()` method.
- Add forecasting tools such as `accuracy()` calculations.

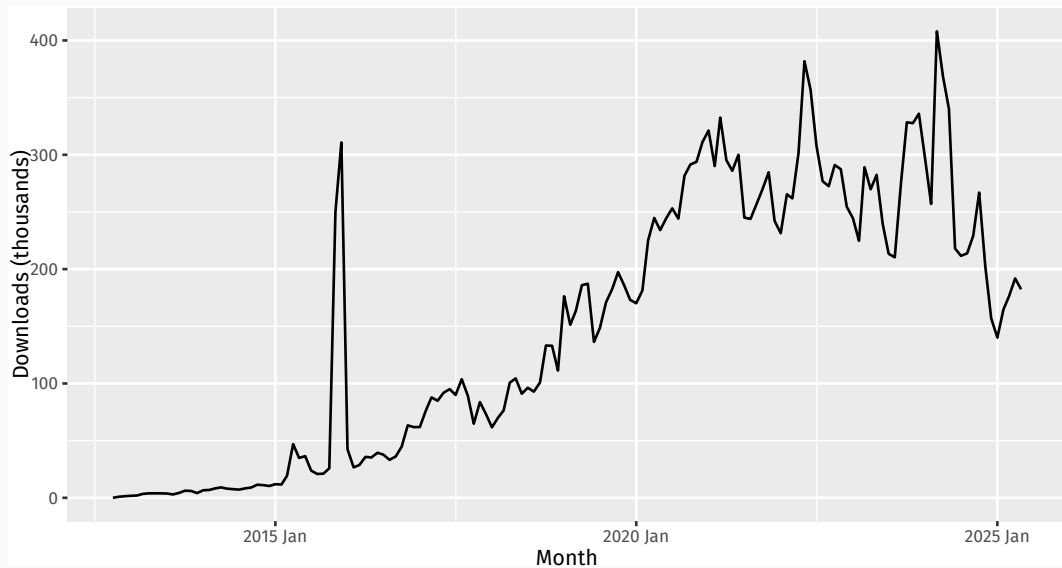
forecast package for R

Date	Event
Pre 2003	Collection of functions used for consulting projects
July/August 2003	ets() and thetaf() added
August 2006	v1.0 available on CRAN
May 2007	auto.arima() added
July 2008	JSS paper (Hyndman & Khandakar)
September 2009	v2.0 . Unbundled from Mcomp, fma & expsmooth
May 2010	arfima() added
Feb/March 2011	tslm(), stlf(), naive(), snaive() added
August 2011	v3.0 . Box Cox transformations added
December 2011	tbats() added

forecast package for R

Date	Event
April 2012	Package moved to github
November 2012	v4.0. <code>nnetar()</code> added
June 2013	Major speed-up of <code>ets()</code>
January 2014	v5.0. <code>tsoutliers()</code> and <code>tsclean()</code> added
May 2015	v6.0. Added several new plots
February 2016	v7.0. Added ggplot2 graphics & bias adjustment
March 2017	v8.0. Added <code>tsCV()</code> & <code>baggedETS()</code>
April 2018	v8.3. Added <code>mstl()</code> , and revised <code>auto.arima()</code>
April 2025	v8.24. Last update

forecast package for R



forecast package for R

- `auto.arima + forecast`
- `ets + forecast`
- `tbats + forecast`
- `bats + forecast`
- `arfima + forecast`
- `nnetar + forecast`
- `stlm + forecast`
- `meanf`
- `rwf, naive`
- `thetaf`
- `dshw, hw, holt, ses`
- `splinef`
- `croston`

All produce an object
of class `forecast`

forecast package for R

- `auto.arima + forecast`
- `ets + forecast`
- `tbats + forecast`
- `bats + forecast`
- `arfima + forecast`
- `nnetar + forecast`
- `stlm + forecast`
- `meanf`
- `rwf, naive`
- `thetaf`
- `dshw, hw, holt, ses`
- `splinef`
- `croston`

All produce an object
of class `forecast`

v9.0 will have new
model functions:

- `mean_model()`
- `rw_model()`
- `theta_model()`
- `spline_model()`
- `croston_model`

CRAN Task View Time Series

CRAN Task View: Time Series Analysis

Maintainer: Rob J Hyndman, Rebecca Killick

Contact: Rob.Hyndman at monash.edu

Version: 2025-05-17

URL: <https://CRAN.R-project.org/view=TimeSeries>

Source: <https://github.com/cran-task-views/TimeSeries/>

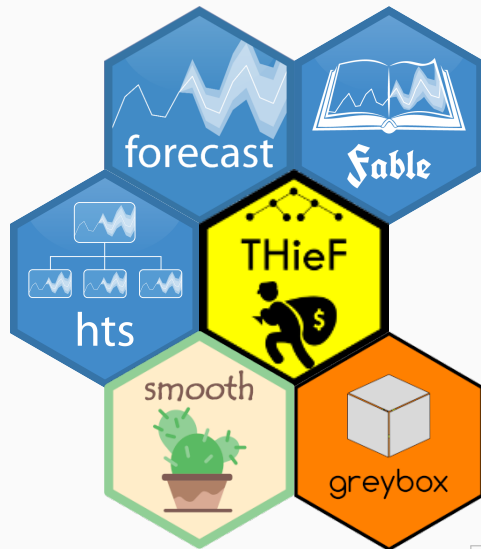
Contributions: Suggestions and improvements for this task view are very welcome and can be made through issues or pull requests on GitHub or via e-mail to the maintainer address. For further details see the [Contributing guide](#).

Citation: Rob J Hyndman, Rebecca Killick (2025). CRAN Task View: Time Series Analysis. Version 2025-05-17. URL <https://CRAN.R-project.org/view=TimeSeries>.

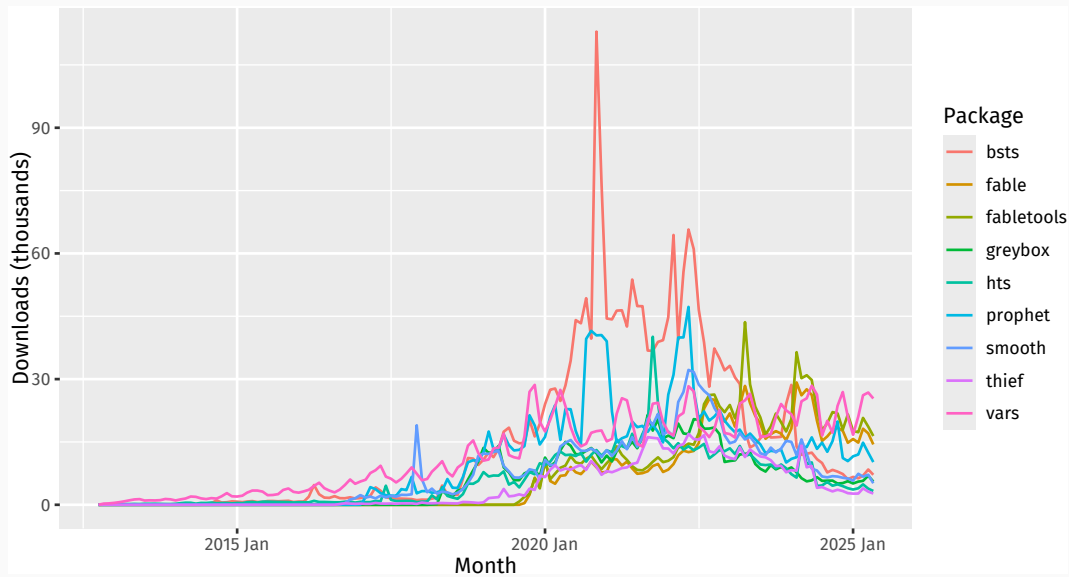
Installation: The packages from this task view can be installed automatically using the [ctv](#) package. For example, `ctv::install.views("TimeSeries", coreOnly = TRUE)` installs all the core packages or `ctv::update.views("TimeSeries")` installs all packages that are not yet installed and up-to-date. See the [CRAN Task View Initiative](#) for more details.

Top ten downloaded forecasting packages on CRAN

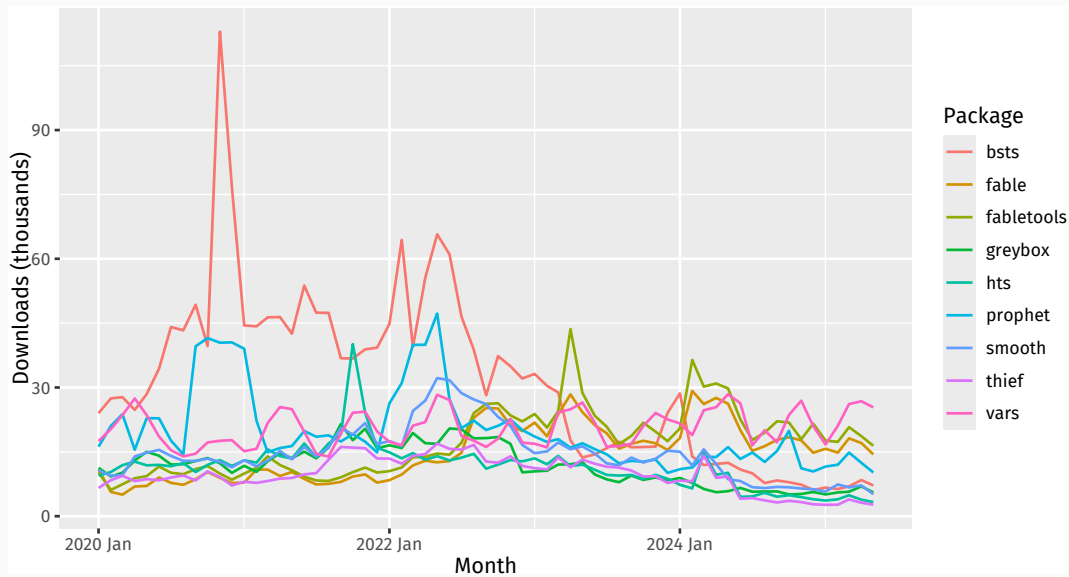
Package	Downloads ('000)
forecast	22783
bsts	2322
vars	1851
prophet	1578
smooth	1159
fabletools	1134
fable	982
greybox	910
hts	896
thief	673



Top ten downloaded forecasting packages on CRAN



Top ten downloaded forecasting packages on CRAN



Function	PIntervals	Automatic	Covariates
<code>stats::HoltWinters()</code>	No	No	No
<code>forecast::ets()</code>	Yes	Yes	No
<code>fable::ETS()</code>	Yes	Yes	No
<code>smooth::es()</code>	Yes	Yes	Yes

forecast::ets()

```
ets(AirPassengers)
```

```
ETS(M,Ad,M)
```

```
Call:
```

```
ets(y = AirPassengers)
```

```
Smoothing parameters:
```

```
alpha = 0.7096
```

```
beta  = 0.0204
```

```
gamma = 1e-04
```

```
phi   = 0.98
```

```
Initial states:
```

```
l = 120.9939
```

```
b = 1.7705
```

```
s = 0.8944 0.7993 0.9217 1.059 1.22 1.232
```

```
1.111 0.9786 0.9804 1.011 0.8869 0.9059
```

forecast::ets()

```
ets(AirPassengers) |> forecast(h = 10)
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan 1961	441.8	419.6	464.0	407.9	475.7
Feb 1961	434.1	407.2	461.1	392.9	475.3
Mar 1961	496.6	460.6	532.6	441.6	551.7
Apr 1961	483.2	443.6	522.9	422.6	543.8
May 1961	484.0	440.0	528.0	416.7	551.2
Jun 1961	551.0	496.3	605.7	467.4	634.7
Jul 1961	613.2	547.4	679.0	512.6	713.8
Aug 1961	609.4	539.2	679.5	502.1	716.6
Sep 1961	530.5	465.5	595.6	431.0	630.0
Oct 1961	463.0	402.8	523.2	371.0	555.1

fable::ETS()

```
as_tsibble(AirPassengers) |>  
  model(ets = ETS(value)) |>  
  report()
```

Series: value

Model: ETS(M,Ad,M)

Smoothing parameters:

alpha = 0.7096

beta = 0.02041

gamma = 0.0001005

phi = 0.98

Initial states:

```
l[0] b[0] s[0] s[-1] s[-2] s[-3] s[-4] s[-5] s[-6] s[-7] s[-8] s[-  
9] s[-10]  
121 1.771 0.8944 0.7993 0.9217 1.059 1.22 1.232 1.111 0.9786 0.9804 1.011 0.8869  
s[-11]  
0.9059
```

fable::ETS()

```
as_tsibble(AirPassengers) |>  
  model(ets = ETS(value)) |>  
  forecast(h = 10)
```

```
# A fable: 10 x 4 [1M]  
# Key:      .model [1]  
  .model    index      value .mean  
  <chr>      <mth>      <dist> <dbl>  
1 ets      1961 Jan  N(442, 299)  442.  
2 ets      1961 Feb  N(434, 442)  434.  
3 ets      1961 Mar  N(497, 789)  497.  
4 ets      1961 Apr  N(483, 956)  483.  
5 ets      1961 May  N(484, 1177) 484.  
6 ets      1961 Jun  N(551, 1821) 551.  
7 ets      1961 Jul  N(613, 2636) 613.  
8 ets      1961 Aug  N(609, 2994) 609.  
9 ets      1961 Sep  N(531, 2577) 531.  
10 ets     1961 Oct  N(463, 2205) 463.
```

smooth::es()

```
es(AirPassengers)
```

Time elapsed: 1.85 seconds

Model estimated using es() function: ETS(MMdM)

With optimal initialisation

Distribution assumed in the model: Normal

Loss function type: likelihood; Loss function value: 526.8

Persistence vector g:

alpha	beta	gamma
0.3536	0.0000	0.4560

Damping parameter: 0.9991

Sample size: 144

Number of estimated parameters: 18

Number of degrees of freedom: 126

Information criteria:

AIC	AICc	BIC	BICc
1090	1095	1143	1157

smooth::es()

```
es(AirPassengers) |> forecast(h = 10, interval = "parametric")
```

	Point forecast	Lower bound (2.5%)	Upper bound (97.5%)
Jan 1961	450.4	413.7	486.6
Feb 1961	426.3	389.2	463.8
Mar 1961	483.1	439.3	527.0
Apr 1961	506.8	459.5	555.7
May 1961	522.0	471.2	575.0
Jun 1961	596.6	535.7	660.7
Jul 1961	688.2	615.9	763.3
Aug 1961	681.8	606.5	758.4
Sep 1961	567.3	503.9	634.5
Oct 1961	505.1	445.0	567.7

Benchmarks

```
bench::mark(  
  forecast = ets(AirPassengers) |> forecast(h = 10),  
  fable = as_tsibble(AirPassengers) |> model(ETS(value)) |> forecast(h = 10),  
  smooth = es(AirPassengers) |> forecast(h = 10, interval = "parametric"),  
  check = FALSE  
)
```

expression	min	median	itr/sec	mem_alloc
forecast	673.96ms	673.96ms	1.48	43.6MB
fable	729.2ms	729.2ms	1.37	37.5MB
smooth	1.61s	1.61s	0.62	221.9MB

Function	PIntervals	Automatic	Covariates
<code>stats::arima()</code>	Yes	No	Yes
<code>forecast::Arima()</code>	Yes	No	Yes
<code>forecast::auto.arima()</code>	Yes	Yes	Yes
<code>fable::ARIMA()</code>	Yes	Yes	Yes
<code>smooth::ssarima()</code>	Yes	No	Yes
<code>smooth::auto.ssarima()</code>	Yes	Yes	Yes

forecast::auto.arima()

```
auto.arima(AirPassengers, lambda = 0)
```

Series: AirPassengers

ARIMA(0,1,1)(0,1,1)[12]

Box Cox transformation: lambda= 0

Coefficients:

	ma1	sma1
	-0.402	-0.557
s.e.	0.090	0.073

sigma^2 = 0.00137: log likelihood = 244.7

AIC=-483.4 AICc=-483.2 BIC=-474.8

forecast::auto.arima()

```
auto.arima(AirPassengers, lambda = 0) |> forecast(h = 10)
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan 1961	450.4	429.5	472.3	418.9	484.3
Feb 1961	425.7	402.8	449.9	391.2	463.3
Mar 1961	479.0	450.1	509.7	435.6	526.8
Apr 1961	492.4	459.9	527.2	443.5	546.7
May 1961	509.1	472.7	548.2	454.6	570.0
Jun 1961	583.3	538.9	631.5	516.8	658.5
Jul 1961	670.0	615.9	728.9	589.1	762.1
Aug 1961	667.1	610.4	729.1	582.3	764.2
Sep 1961	558.2	508.5	612.8	484.0	643.8
Oct 1961	497.2	451.0	548.1	428.3	577.2

fable::ARIMA()

```
as_tsibble(AirPassengers) |>  
  model(arima = ARIMA(log(value))) |>  
  report()
```

Series: value

Model: ARIMA(2,0,0)(0,1,1)[12] w/ drift

Transformation: log(value)

Coefficients:

	ar1	ar2	sma1	constant
	0.5754	0.2614	-0.5553	0.0193
s.e.	0.0843	0.0842	0.0771	0.0015

sigma^2 estimated as 0.001323: log likelihood=249.7

AIC=-489.3 AICc=-488.8 BIC=-474.9

fable::ARIMA()

```
as_tsibble(AirPassengers) |>  
  model(arima = ARIMA(log(value))) |>  
  forecast(h = 10)
```

```
# A fable: 10 x 4 [1M]  
# Key:      .model [1]  
  .model    index      value .mean  
  <chr>     <mth>      <dbl> <dbl>  
1 arima 1961 Jan t(N(6.1, 0.0013)) 453.  
2 arima 1961 Feb t(N(6.1, 0.0018)) 430.  
3 arima 1961 Mar t(N(6.2, 0.0022)) 486.  
4 arima 1961 Apr t(N(6.2, 0.0025)) 502.  
5 arima 1961 May t(N(6.3, 0.0028)) 522.  
6 arima 1961 Jun  t(N(6.4, 0.003)) 600.  
7 arima 1961 Jul  t(N(6.5, 0.0031)) 691.  
8 arima 1961 Aug  t(N(6.5, 0.0032)) 690.  
9 arima 1961 Sep  t(N(6.4, 0.0033)) 579.  
10 arima 1961 Oct t(N(6.2, 0.0034)) 516.
```

smooth::auto.ssarima()

```
auto.ssarima(log(AirPassengers))
```

Time elapsed: 4.37 seconds

Model estimated: SARIMA(0,1,3)[1](0,1,3)[12]

Matrix of MA terms:

	Lag 1	Lag 12
MA(1)	-0.4157	-0.7397
MA(2)	0.0313	0.1145
MA(3)	-0.1255	0.0747

Initial values were produced using backcasting.

Loss function type: likelihood; Loss function value: -287.9556

Error standard deviation: 0.0336

Sample size: 144

Number of estimated parameters: 7

Number of degrees of freedom: 137

Information criteria:

AIC	AICc	BIC	BICc
-561.9	-561.1	-541.1	-539.1

smooth::auto.ssarima()

```
auto.ssarima(log(AirPassengers)) |> forecast(h = 10)
```

	Point forecast	Lower bound (2.5%)	Upper bound (97.5%)
Jan 1961	6.104	6.039	6.169
Feb 1961	6.049	5.973	6.124
Mar 1961	6.181	6.096	6.266
Apr 1961	6.185	6.094	6.276
May 1961	6.224	6.128	6.321
Jun 1961	6.372	6.271	6.474
Jul 1961	6.506	6.399	6.612
Aug 1961	6.512	6.401	6.623
Sep 1961	6.324	6.209	6.440
Oct 1961	6.201	6.082	6.321

Benchmarks

```
bench::mark(  
  forecast = auto.arima(AirPassengers, lambda = 0, biasadj = TRUE) |>  
    forecast(h = 12),  
  fable = as_tsibble(AirPassengers) |> model(ARIMA(log(value))) |>  
    forecast(h = 12),  
  smooth = auto.ssarima(log(AirPassengers)) |>  
    forecast(h = 12, interval="parametric"),  
  check = FALSE  
)
```

expression	min	median	itr/sec	mem_alloc
forecast	1.92s	1.92s	0.52	402.39MB
fable	5.26s	5.26s	0.19	1.28GB
smooth	3.2s	3.2s	0.31	428.36MB

Outline

1

R

2

Python

3

Julia

Python packages

- pmdarima
- sktime
- **Nixtla:** statsforecast, hierarchicalforecast
- GluonTS
- Merlion
- statsmodels,
- pyhts
- Darts

Outline

1

R

2

Python

3

Julia

■ forecast