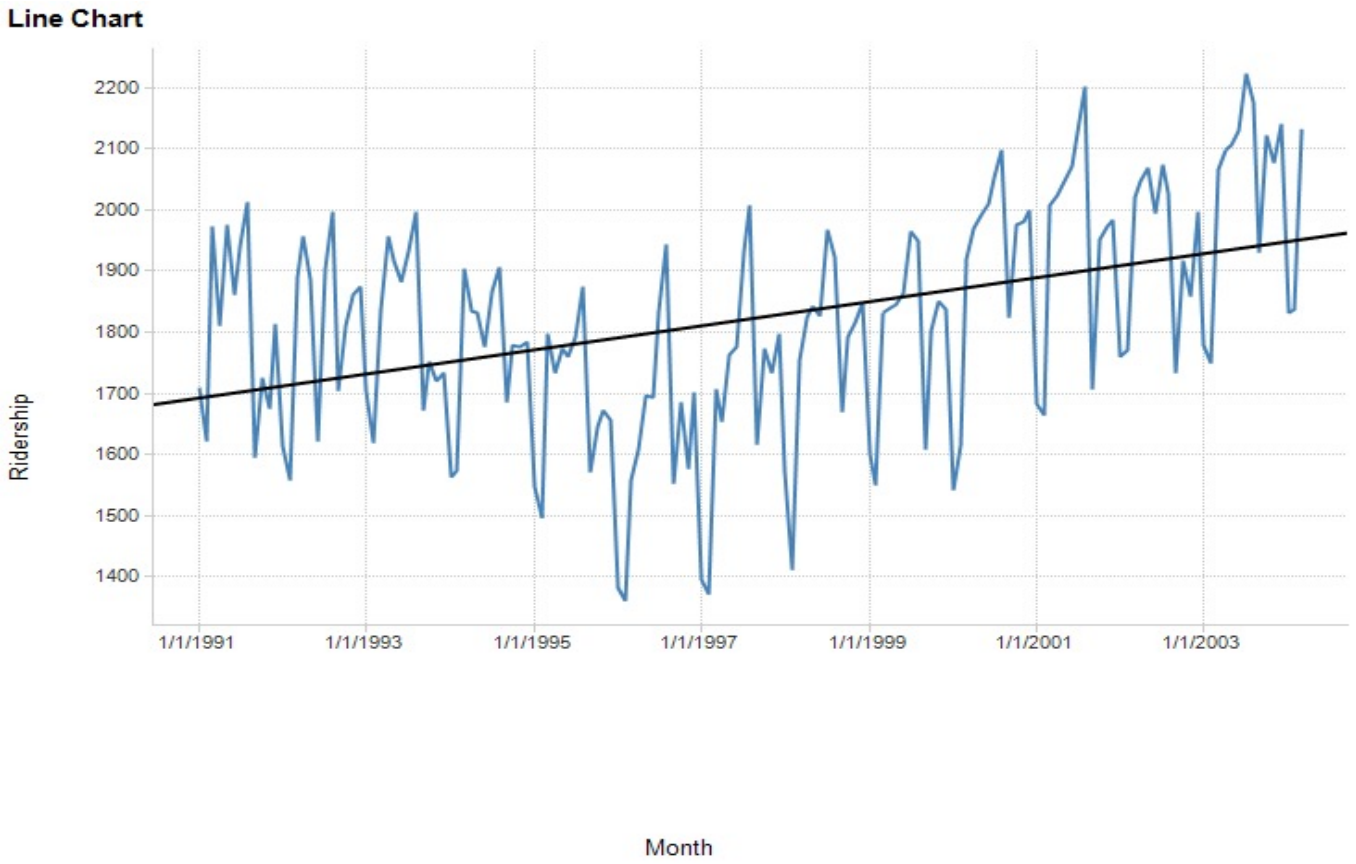
Regression Based Forecasting

Main ideas

- Fit linear trend, time as predictor
- Modify & use also for non-linear trends
  - Exponential
  - Polynomial
- Can also capture seasonality

Linear fit to Amtrak ridership data

(Doesn't fit too well – more later)



The regression model

Ridership Y is a function of time (t) and noise (error = e)

$Y_i = B_0 + B_1 * t + e$

Thus we model 3 of the 4 components:

- Level (B_0)
- Trend* (B_1)
- Noise (e)

*Our trend model is linear, which we can see from the graph is not a good fit (more later)

function `ts` converts Amtrak .csv data into time series object

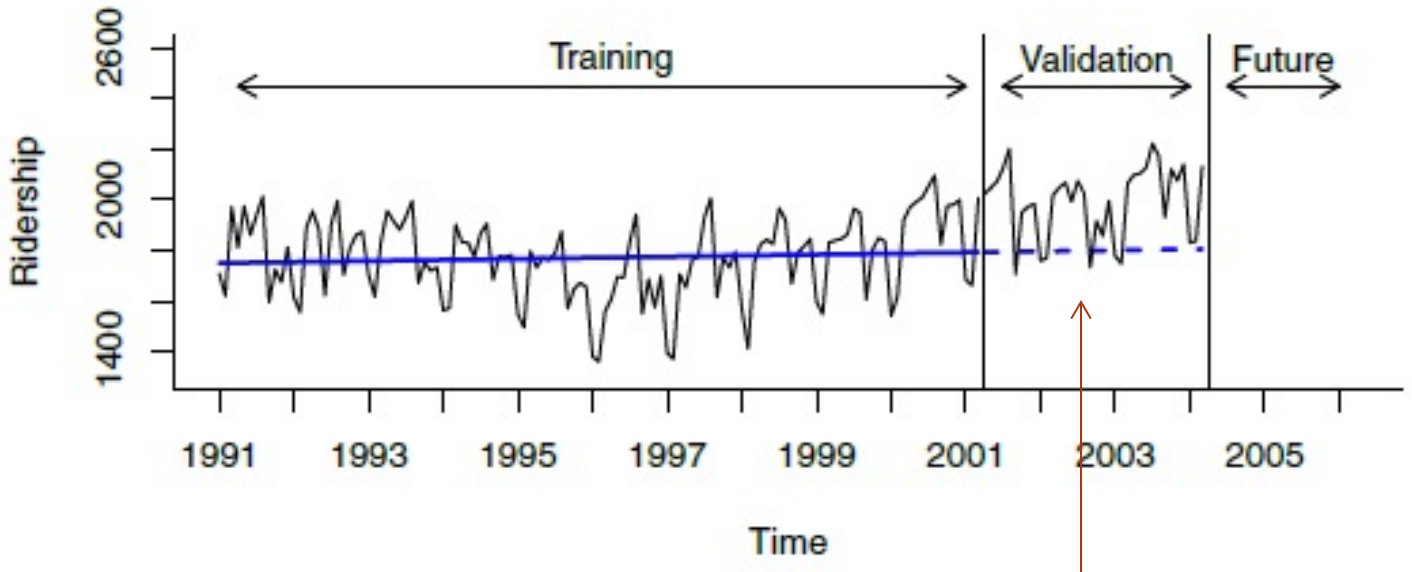
Although the original data have the time points, `ts` uses only the ridership data, and recreates the time points itself with `start`, `end`, and `freq`

```
library(forecast)
Amtrak.data <- read.csv("Amtrak.csv")

# create time series
ridership.ts <- ts(Amtrak.data$Ridership, start = c(1991,1),
     end = c(2004,3), freq = 12)

# produce linear trend model
ridership.lm <- tslm(ridership.ts ~ trend)

# plot the series
plot(ridership.ts, xlab = "Time", ylab = "Ridership", ylim =
     c(1300,2300), bty = "l")
lines(ridership.lm$fitted, lwd = 2)
```

`trend` is predictor variable created on the fly from the time series data

`tslm` fits linear model to data with time series components

```
# fit linear trend model to training set and create
forecasts
train.lm <- tslm(train.ts ~ trend)
train.lm.pred <- forecast(train.lm, h = nValid, level = 0)
```



Trend based on training data underestimates validation period

Exponential Trend

Appropriate model when increase/decrease in series over time is multiplicative

e.g. t_1 is x% more than t_0, t_2 is x% more than t_1…

Replace Y with log(Y) then fit linear regression

$$log(Y_i) = B_0 + B_1 t + e$$

Exponential trend - forecast errors

Note that performance measures in standard linear regression software are not in original units

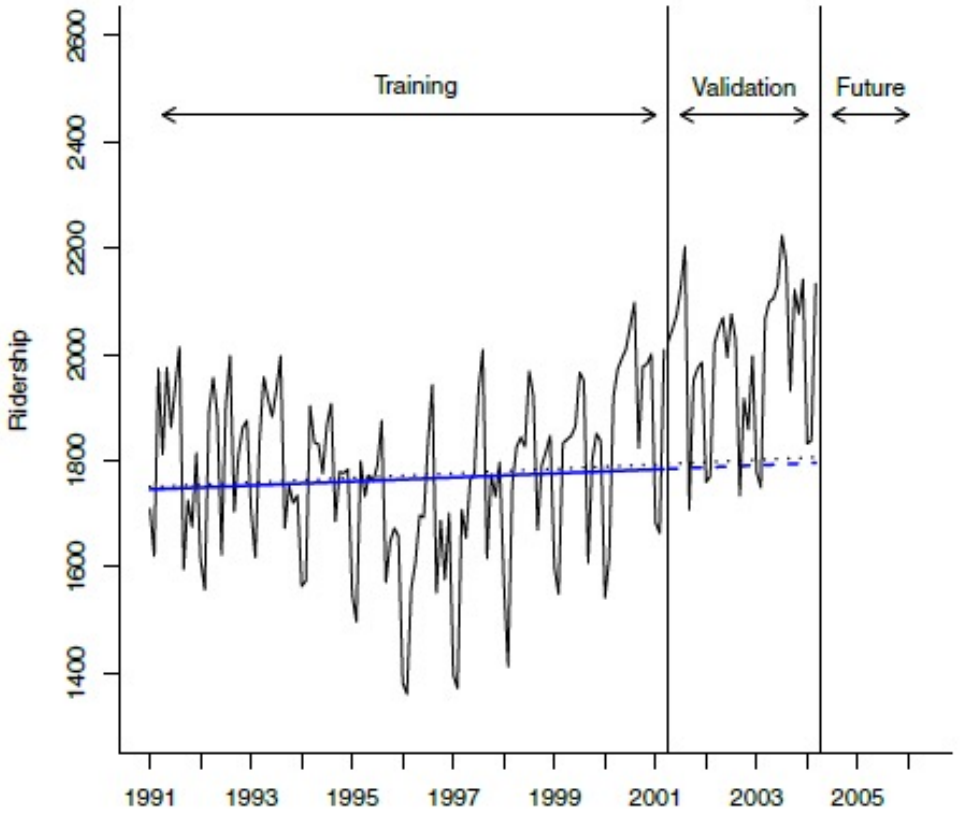
Model forecasts will be in the form log(*Y*)

Return to original units by taking exponent of model forecasts

Calculate standard deviation of these forecast errors to get RMSE

```
# fit exponential trend using tslm() with argument
# lambda = 0

train.lm.expo.trend <- tslm(train.ts ~ trend, lambda = 0)
train.lm.expo.trend.pred <- forecast(train.lm.expo.trend,
     h = nValid, level = 0)
```

Exponential trend (dotted line) very similar to linear trend (solid line)



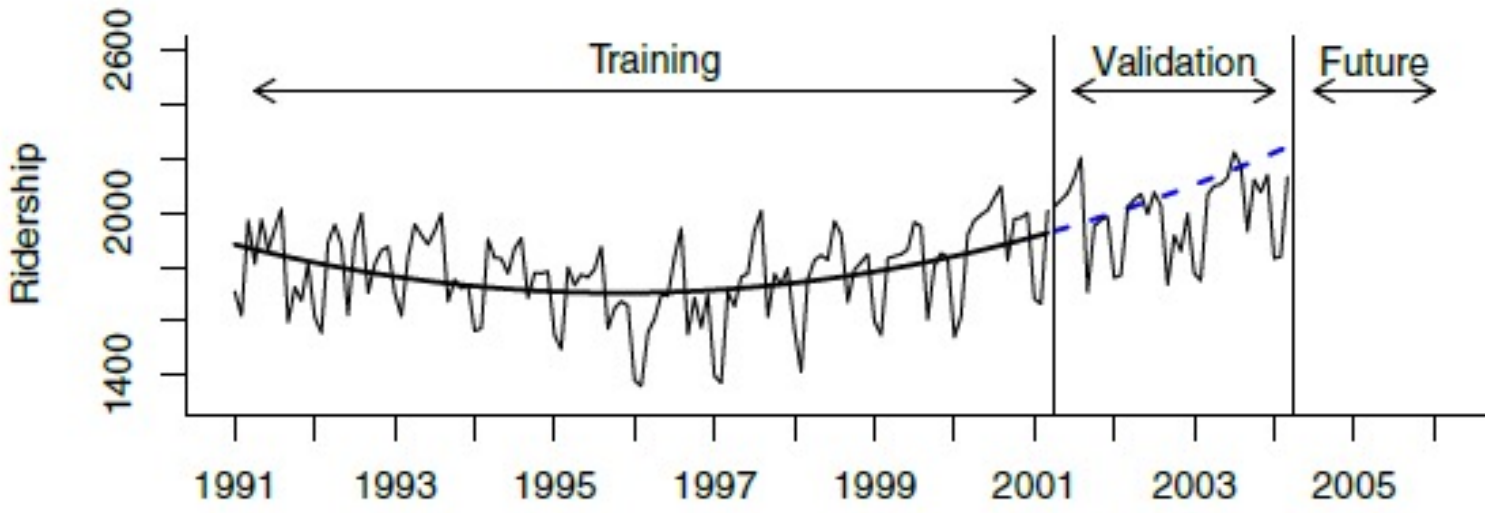
Polynomial Trend

Add additional predictors as appropriate

For example, for quadratic relationship add a t^2 predictor

Fit linear regression using both t and t^2

```
# fit quadratic trend using function I(), which treats an
# object "as is".

train.lm.poly.trend <- tslm(train.ts ~ trend + I(trend^2))
summary(train.lm.poly.trend)
train.lm.poly.trend.pred <- forecast(train.lm.poly.trend,
     h = nValid, level = 0)
```



- Better job capturing the trend, though it over forecasts in validation period.
- Next: we'll try capturing seasonality.

Handling Seasonality

- Seasonality is any recurring cyclical pattern of consistently higher or lower values (daily, weekly, monthly, quarterly, etc.)
- Handle in regression by adding categorical variable for season, e.g.

Month	Ridership	Season
Jan-91	1709	Jan
Feb-91	1621	Feb
Mar-91	1973	March
Apr-91	1812	April

11, not 12, to avoid multicollinearity

```
# include season as a predictor in tslm(). Here it creates 11
# dummies, one for each month except for first season, January
train.lm.season <- tslm(train.ts ~ season)
summary(train.lm.season)
```

Final model, Amtrak data

Incorporates trend and seasonality

13 predictors

- 11 monthly dummies
- t
- t^2

```
train.lm.trend.season <- tslm(train.ts ~ trend +
     I(trend^2) + season)
```

Autocorrelation

Unlike cross-sectional data, time-series values are typically correlated with nearby values ("autocorrelation")

Ordinary regression does not account for this

Computing autocorrelation

Create “lagged” series

Copy of the original series, offset by one or more timer periods

Compute correlation between original series and lagged series

- Lag-1, lag-2, etc.

Autocorrelation

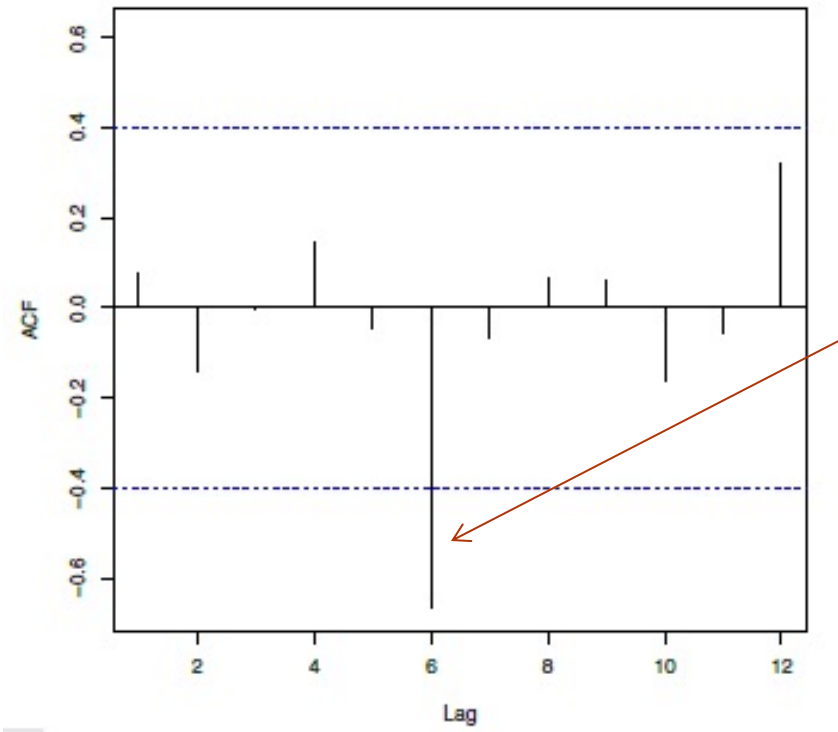
Positive autocorrelation at lag-1 = stickiness

Strong autocorrelation (positive or negative) at a lag > 1 indicates seasonal (cyclical) pattern

Autocorrelation in residuals indicates the model has not fully captured the seasonality in the data

Compute & display autocorrelation for different lags, over 24 months:

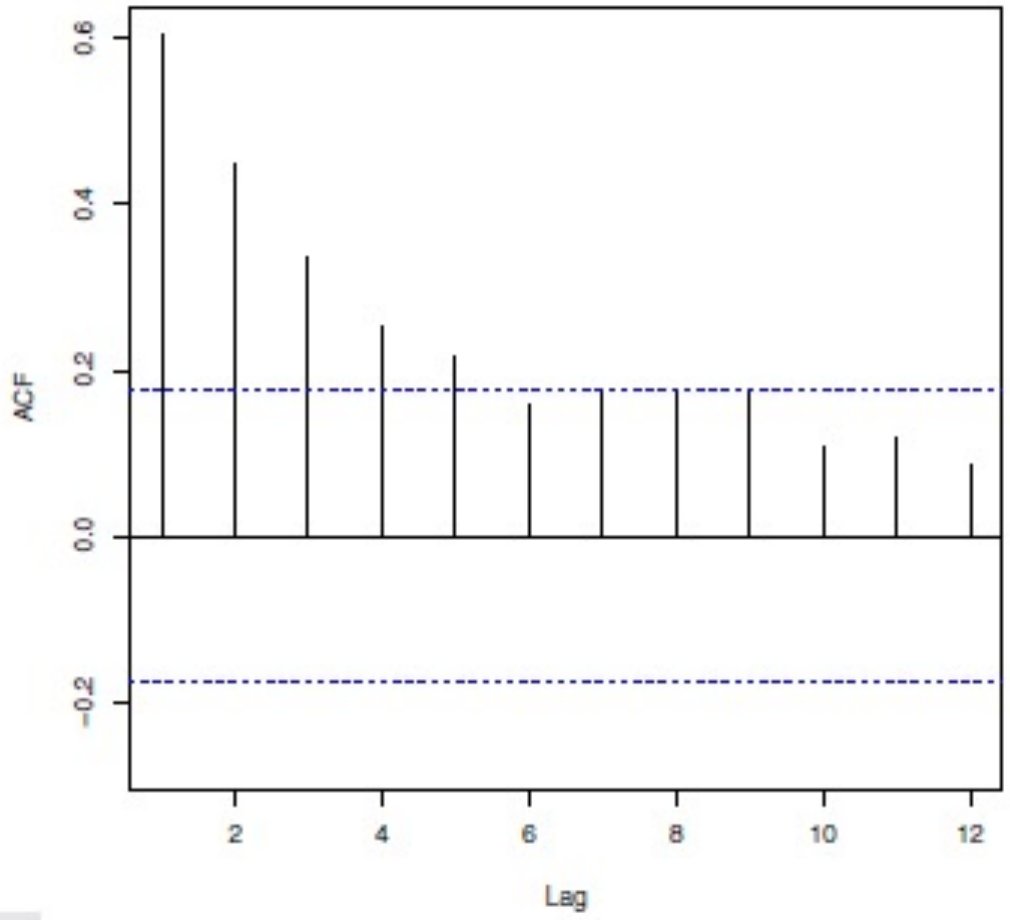
```
ridership.24.ts <- window(train.ts, start = c(1991, 1),
     end = c(1991, 24))
Acf(ridership.24.ts, lag.max = 12, main = "")
```



Strong negative correlation at 6 months shows seasonal pattern (high summer traffic, low winter)

The dotted lines are confidence bounds for judging statistical significance

It is useful to examine autocorrelation for the residuals:



Strong autocorrelation from lag 1 on, but lag 6 no longer dominates.

Note: If you have correlation at lag 1, it will naturally propagate to lag 2, 3, etc., tapering off

Incorporating autocorrelation into models

Use a forecasting method to forecast *k*-steps ahead

Fit AR (autoregressive) model to residuals

Incorporate residual forecasts

Improved $F_{t+k} = F_{t+k} + E_{t+k}$

Choose order of the AR model

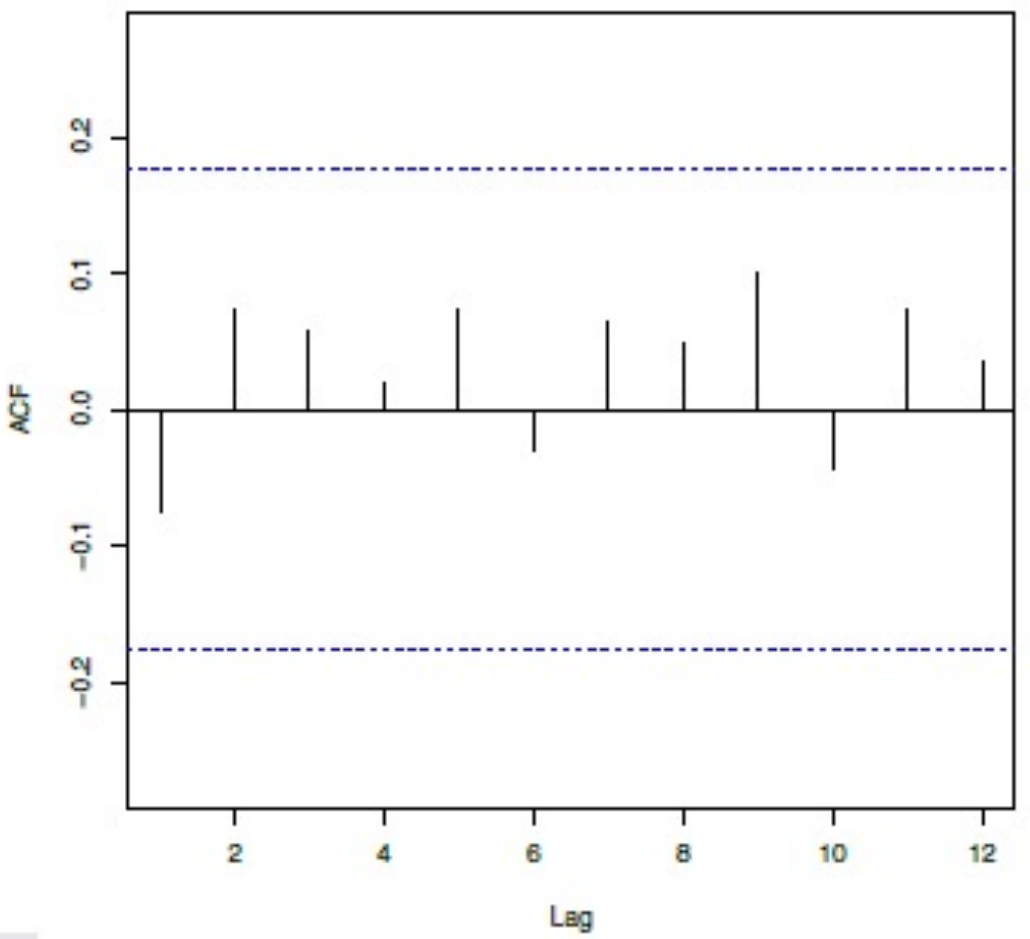
If autocorrelation exists at Lag-1, a Lag-1 model should be sufficient to capture lags at other periods as well

$$E_t = B_0 + B_1 E_{t-1} + e$$

Where E_t is residual (forecast error) at time t

Plot autocorrelation of “residuals of residuals”

Autocorrelation is mostly gone – AR(1) has adequately captured the autocorrelation in the data:



Random walks

- Before forecasting, consider “is the time series predictable?”
- Or is it a random walk?
- Do a statistical hypothesis test that slope = 1 in an AR(1) model (i.e. that the forecast for a period is the most recently-observed value)
- If hypothesis cannot be rejected, series is statistically equivalent to a random walk (i.e. we have not shown that it is predictable).

Summary – Regression Based Forecasting

- Can use linear regression for exponential models (use logs) and polynomials (exponentiation)
- For seasonality, use categorical variable (make dummies)
- Incorporate autocorrelation by modeling it, then using those error forecasts in the main model