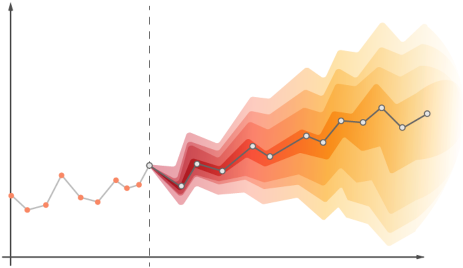
Multivariate Models

DS-5740 Advanced Statistics



Overview: Week 6

Overview | Week 6

Preliminaries

- None

Goals for the Week

- Recap models so far
- Cover vector autoregression (VAR) models
- Cover error-correction on VAR models

Recap

outcome (at time *t*)

sum of weights by predictor (at time *t*)

$$\underbrace{y_t}_{\text{outcome (at time } t)} = \underbrace{\beta_0}_{\text{intercept}} + \underbrace{\sum_k^n \beta_k x_{k,t}}_{\text{sum of weights by predictor (at time } t)} + \underbrace{\eta_t}_{\text{ARIMA errors}}$$

intercept

ARIMA errors

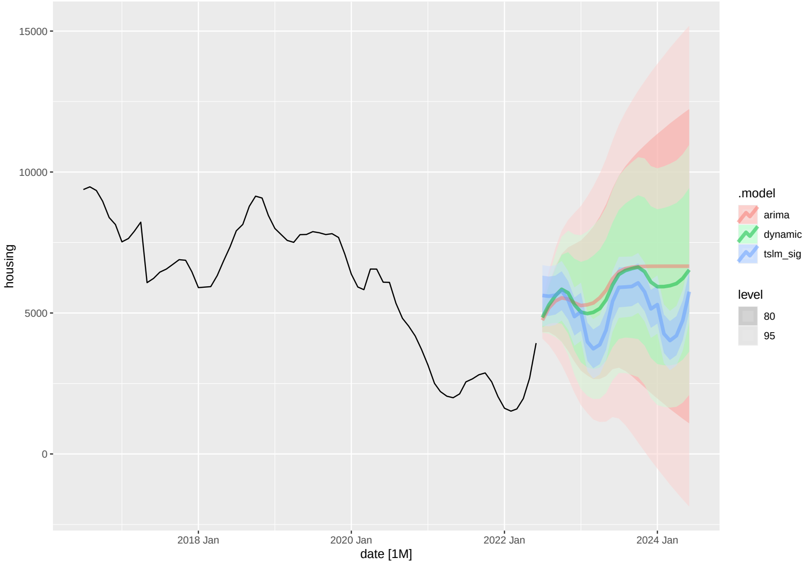
autoregressive

moving average

$$\eta_t = \underbrace{\phi_1 \eta_{t-1} + \ldots + \phi_p \eta_{t-p}}_{\text{autoregressive}} + \underbrace{\theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}}_{\text{moving average}} + \underbrace{\epsilon_t}_{\text{error}}$$

error

Multivariate Time Series | Recap



- `TSLM`:

- `TSLM`: general model for selecting predictors, requires new data values to be specified, does not recover dynamics well (i.e., autocorrelations) but possible to include lags (i.e., distributed lag model)

- `ETS`:

- `TSLM`: general model for selecting predictors, requires new data values to be specified, does not recover dynamics well (i.e., autocorrelations) but possible to include lags (i.e., distributed lag model)

- `ETS`: captures trend and seasonal data dynamics, differentially weights past time points, does not require new data to forecast

- `ARIMA`:

- `TSLM`: general model for selecting predictors, requires new data values to be specified, does not recover dynamics well (i.e., autocorrelations) but possible to include lags (i.e., distributed lag model)

- `ETS`: captures trend and seasonal data dynamics, differentially weights past time points, does not require new data to forecast

- `ARIMA`: captures dynamics well, requires stationarity (e.g., differencing), does not require new data to forecast

- `TSLM+ARIMA`:

- `TSLM`: general model for selecting predictors, requires new data values to be specified, does not recover dynamics well (i.e., autocorrelations) but possible to include lags (i.e., distributed lag model)

- `ETS`: captures trend and seasonal data dynamics, differentially weights past time points, does not require new data to forecast

- `ARIMA`: captures dynamics well, requires stationarity (e.g., differencing), does not require new data to forecast

- `TSLM+ARIMA`: captures dynamics while also allowing for predictors, requires stationarity (e.g., differencing), requires new data to forecast

Multivariate Time Series

What if we want to predict multiple variables?

What if we want to predict multiple variables?

Vector Autoregression

- Dynamics are often influenced by more than one variable

- While predicting one variable is useful, we often care about more than one variable

- Understanding the dynamics of multiple variables and their influence on one another can help us make better (more accurate) predictions (even for one variable!)

Autoregression

$$\underbrace{y_t}_{\text{outcome}} = \underbrace{c}_{\text{constant}} + \underbrace{\phi_1 y_{t-1} + \ldots + \phi_p y_{t-p}}_{\text{autoregressive}} + \underbrace{\epsilon_t}_{\text{error}}$$

- y_t = [3.4, 2.4, 2.3, 4.5, 4.2, 2.4]
- y_{t-1} = [NA, 3.4, 2.4, 2.3, 4.5, 4.2]

Vector Autoregression

- VAR(p)

$$y_{1,t} = c_1 + \phi_{11,1} y_{1,t-1} + \phi_{12,1} y_{2,t-1} + \epsilon_{1,t}$$

$$y_{2,t} = c_2 + \phi_{21,1} y_{1,t-1} + \phi_{22,1} y_{2,t-1} + \epsilon_{2,t}$$

- $\phi_{ok,p}$
  - o = variable being predicted (outcome)
  - k = predictor
  - p = lag

Remember stationary?


FUHGEDDABOUDIT
DEFOREST TUMBLR

Don't actually fuhgeddaboutit but with VAR the data don't *need* to be stationary at the outset (but they eventually do!)

Code in {fpp3}

```r
# Vector autoregression
fit_var <- housing_ts %>%
  model(
    lag_2 = VAR(
      vars(housing, unemployment, median_days,
           price_decreased, pending_listing) ~
        xreg(outlier, pandemic)
    )
  )

# Report fit
report(fit_var)
```

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

- column = o
- row = k, p

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

- In terms of $\phi_{ok,p}$, what is `1.446`?
- Assume `housing` = outcome 1 and `median_days` = outcome 2

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

- In terms of $\phi_{ok,p}$, what is `1.446`?
- Assume `housing` = outcome 1 and `median_days` = outcome 2
- $\phi_{11,1}$

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

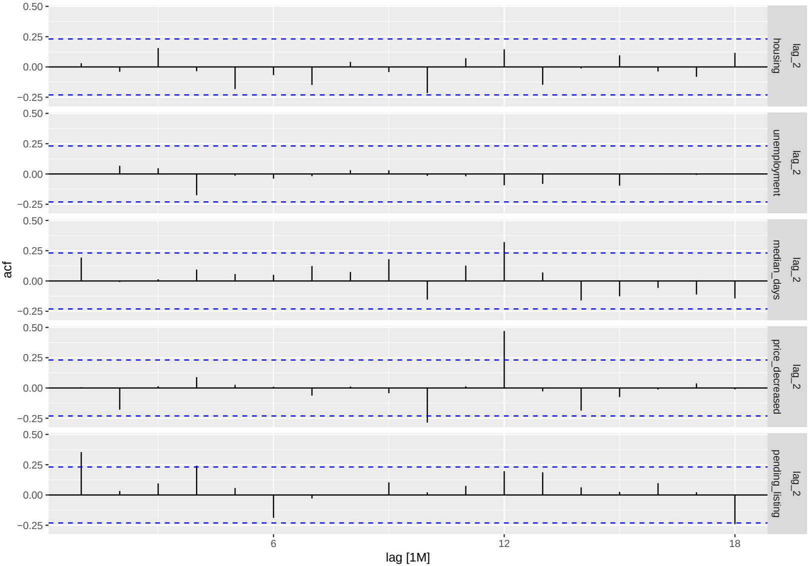
- In terms of $\phi_{ok,p}$, what is `-3.846`?
- Assume `housing` = outcome 1 and `median_days` = outcome 2

Under the hood (ϕ matrix)

	housing	median_days
lag(housing,1)	1.446	0.006
lag(housing,2)	-0.313	-0.003
lag(median_days,1)	-4.036	0.764
lag(median_days,2)	-3.846	-0.256

- In terms of $\phi_{ok,p}$, what is `-3.846`?
- Assume `housing` = outcome 1 and `median_days` = outcome 2
- $\phi_{12,2}$

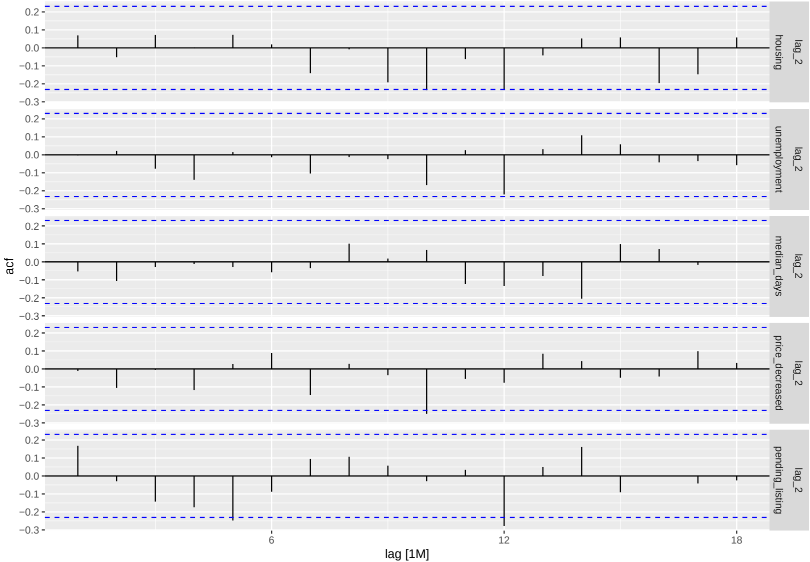
```r
# Autocorrelation of residuals
fit_var %>% augment() %>%
  ACF(.innov) %>% autoplot()
```



Re-fit with seasonal period = 12

```
# Vector autoregression
fit_var <- housing_ts %>%
  model(
    lag_2 = VAR(
      vars(housing, unemployment, median_days,
           price_decreased, pending_listing) ~
        xreg(outlier, pandemic, season(period = 12))
    )
  )
```

Multivariate Time Series | Vector Autoregression



Checking the lag in `VAR`

```r
# Fit VAR(2)
var_2 <- vars::VAR(
  y = housing_ts[,c(
    "housing", "unemployment", "median_days",
    "price_decreased", "pending_listing"
  )],
  exogen = housing_ts[,c("outlier", "pandemic")],
  type = "none", # same as {fpp3}'s `VAR`
  p = 2 # lag
)
serial.test(var_2, lags.pt = 10, type = "PT.adjusted")
```

```
    Portmanteau Test (adjusted)

data:  Residuals of VAR object var_2
Chi-squared = 250.55, df = 200, p-value = 0.008801
```

- $p < 0.05$: significant residual serial correlations
- $p \geq 0.05$: non-significant residual serial correlations

Checking the lag in VAR

```r
# Fit VAR(2) with season
var_2_season <- vars::VAR(
  y = housing_ts[,c(
    "housing", "unemployment", "median_days",
    "price_decreased", "pending_listing"
  )],
  exogen = housing_ts[,c("outlier", "pandemic")],
  type = "none", # same as {fpp3}'s `VAR`
  p = 2, # lag
  season = 12
)
serial.test(var_2_season, lags.pt = 10, type = "PT.adjusted")
```

```
	Portmanteau Test (adjusted)

data:  Residuals of VAR object var_2_season
Chi-squared = 282.17, df = 200, p-value = 0.0001151
```

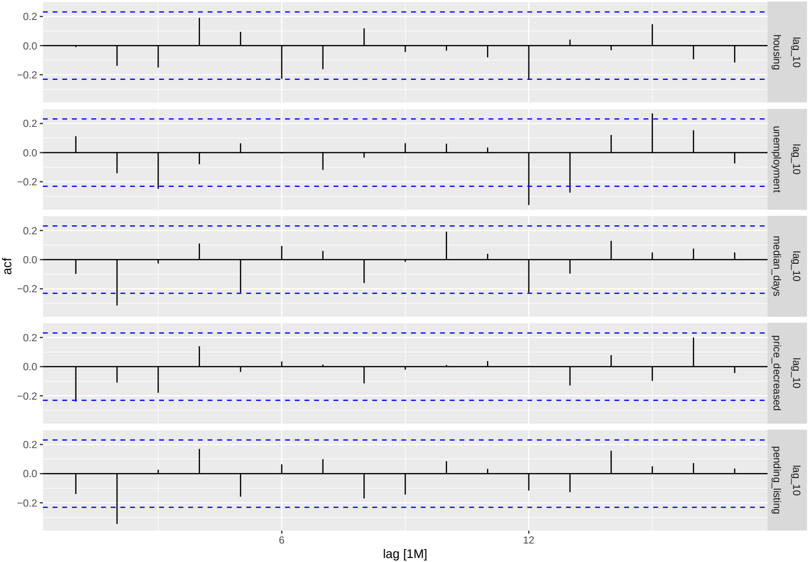
- $p < 0.05$: significant residual serial correlations
- $p \geq 0.05$: non-significant residual serial correlations

Specify the lag in `VAR`

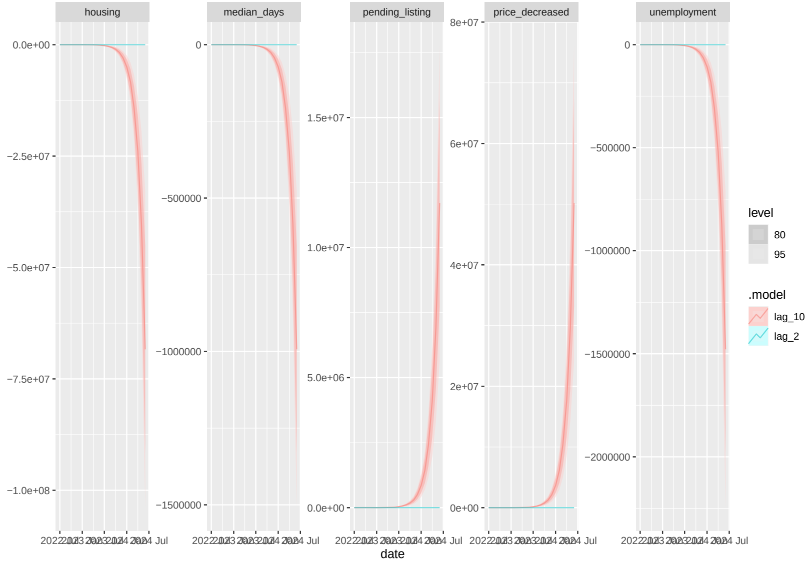
```r
# Vector autoregression
fit_var <- housing_ts %>%
  model(
    lag_2 = VAR(
      vars(housing, unemployment, median_days,
           price_decreased, pending_listing) ~
        xreg(outlier, pandemic)
    ),
    lag_10 = VAR(
      vars(housing, unemployment, median_days,
           price_decreased, pending_listing) ~
        xreg(outlier, pandemic) + AR(0:10)
    )
  )

# Report fit
glance(fit_var) %>% select(.model, AIC, AICc, BIC)
```

```
# A tibble: 2 x 4
  .model    AIC  AICc   BIC
  <chr>   <dbl> <dbl> <dbl>
1 lag_2   3489. 2575. 3680.
2 lag_10  2847. 2110. 3464.
```

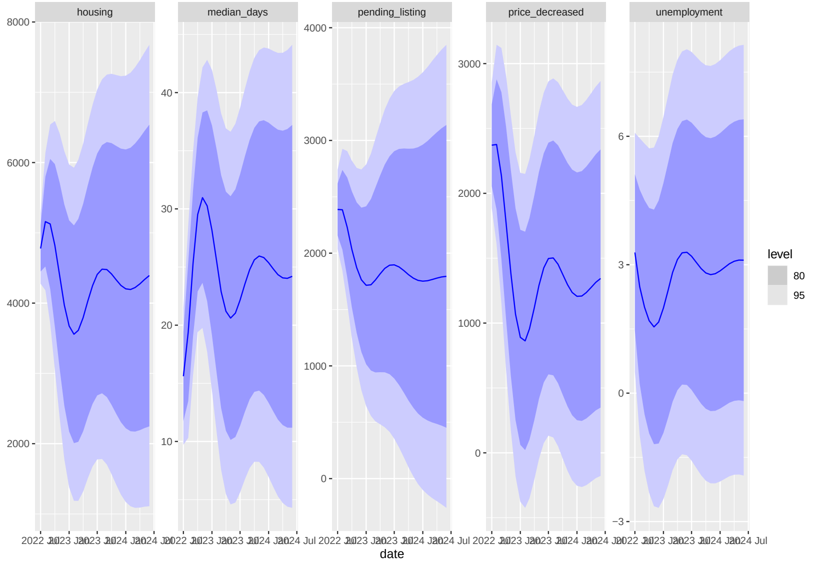
Multivariate Time Series | Vector Autoregression



Forecast


housing
median_days
pending_listing
price_decreased
unemployment
0.0e+00
−2.5e+07
−5.0e+07
−7.5e+07
−1.0e+08
0
−500000
−1000000
−1500000
0.0e+00
5.0e+06
1.0e+07
1.5e+07
0e+00
2e+07
4e+07
6e+07
8e+07
0
−500000
−1000000
−1500000
−2000000
date
level
80
95
.model
lag_10
lag_2

Forecast with VAR(2)

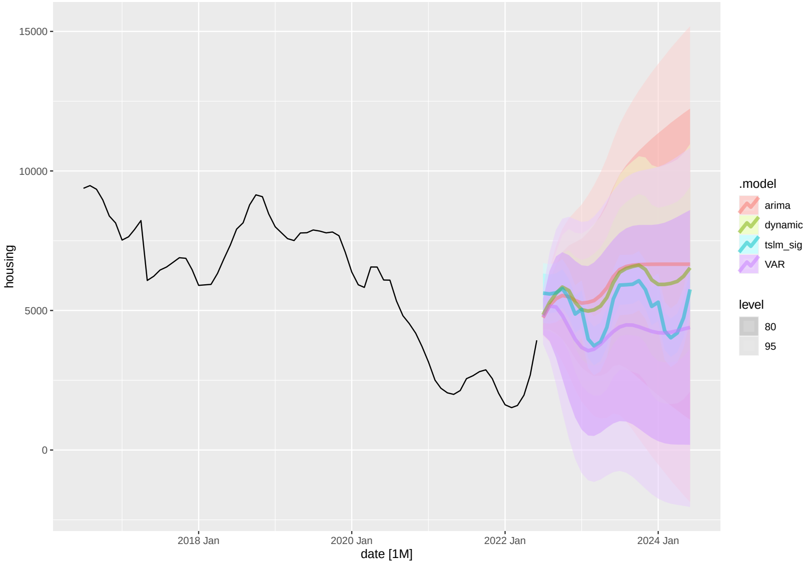


Switching over to {vars}

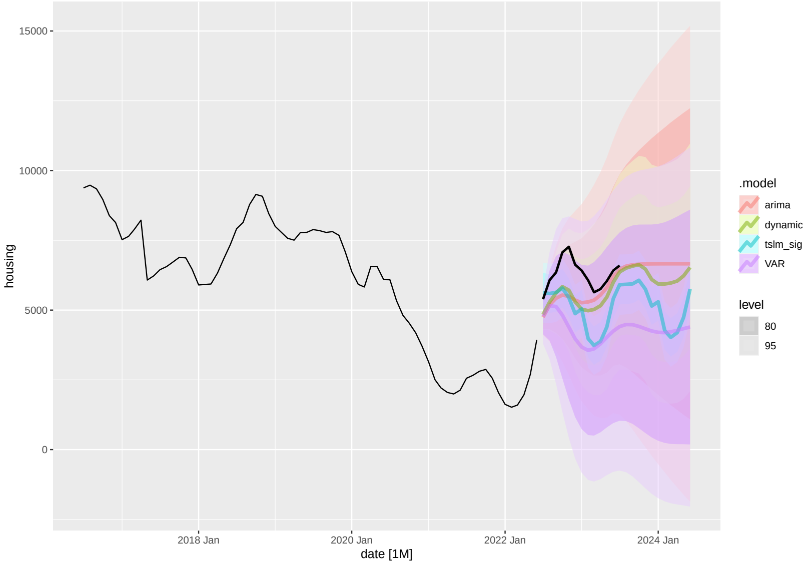
```r
# Load {vars}
library(vars)

# VAR model with {vars}
vars_var <- vars::VAR(
  y = housing_ts[,c(
    "housing", "unemployment", "median_days",
    "price_decreased", "pending_listing"
  )],
  exogen = housing_ts[,c("outlier", "pandemic")],
  type = "none", # same as {fpp3}'s `VAR`
  p = 2 # lag
)

# Make dummy variable matrix
dummat <- matrix(
  rep(0, 2 * 24), nrow = 24,
  dimnames = list(NULL, c("outlier", "pandemic"))
)

# Forecast
var_fc <- predict(vars_var, n.ahead = 24, dumvar = dummat)
```

```r
# Get forecast values
fc_housing <- var_fc$fcst$housing

# Set up forecast as {fpp3} does
var_fc <- data.frame(
  .model = "VAR",
  date = fc_var$date,
  housing = distributional::dist_normal(
    mean = fc_housing[,"fcst"],
    sd = fc_housing[,"CI"]
  ),
  .mean = fc_housing[,"fcst"],
  fc[fc$.model == "tslm_sig", -c(1:4)]
) %>% as_tsibble(index = date)

# Add "housing" to dimnames
dimnames(var_fc$housing) <- "housing"

# Add to original forecast
fc <- bind_rows(fc, var_fc)
```

Forecast `housing` with VAR(2)



It's been a year…


15000
10000
5000
0
housing
2018 Jan
2020 Jan
2022 Jan
2024 Jan
date [1M]
.model
arima
dynamic
tslm_sig
VAR
level
80
95

```r
# Point estimates
fc %>% accuracy(housing_valid)
```

```
# A tibble: 4 x 10
  .model    .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 VAR       Test  2011. 2120. 2011.  31.7  31.7   NaN   NaN 0.659
2 arima     Test   772.  934.  772.  11.9  11.9   NaN   NaN 0.780
3 dynamic   Test   854.  943.  854.  13.4  13.4   NaN   NaN 0.685
4 tslm_sig  Test  1262. 1426. 1298.  20.0  20.7   NaN   NaN 0.566
```

```r
# Distributional estimates
fc %>% accuracy(housing_valid, list(crps = CRPS))
```

```
# A tibble: 4 x 3
  .model    .type  crps
  <chr>     <chr> <dbl>
1 VAR       Test  1232.
2 arima     Test   656.
3 dynamic   Test   594.
4 tslm_sig  Test  1030.
```

Granger Causality

General Idea

- Causality can be inferred when properly considering probable causes

- Granger causality is whether the time series of some variable X has information that systematically predicts variable Y above and beyond Y's own time series

General Idea

- Causality can be inferred when properly considering probable causes

- Granger causality is whether the time series of some variable X has information that systematically predicts variable Y above and beyond Y's own time series
- Is this enough to determine *actual* causality?

Using VAR, Granger causality can be tested by:

- Estimating two models
  - Model 1: univariate autoregression (AR) of Y
  - Model 2: multiple regression of X of Y

Why multiple regression?

Using VAR, Granger causality can be tested by:

- Estimating two models
  - Model 1: univariate autoregression (AR) of *Y*
  - Model 2: multiple regression of *X* of *Y*

Why multiple regression?

- Hypotheses:
  - H_0: *X* does not predict *Y* above and beyond *Y*
  - H_A: *X* does predict *Y* above and beyond *Y*
- Compare prediction errors from the two models
  - If including past values of *X* significantly reduces prediction error, then Granger Causality (residuals significantly lower than without *X*)
  - If not, then no Granger Causality

Assumptions

- Data must be stationary (recall that this assumption is *not* required *at the outset!* for vector autoregression)
- Relationship between X and Y should be linear (also *not* required by vector autoregression)
- Normally distributed errors and no autocorrelations

Activity

Goal: Is there Granger causality from `median_days`?

1. Check for stationary of all variables (if not, then make them stationary)
2. Check for a linear relationship (use `GGally::ggpairs`)
3. Perform `causality()` on model with `cause = "median_days"`

```
$Granger

    Granger causality H0: median_days do not Granger-cause housing
    unemployment price_decreased pending_listing

data:  VAR object var_stationary
F-Test = 2.0533, df1 = 8, df2 = 275, p-value = 0.0406


$Instant

    H0: No instantaneous causality between: median_days and housing
    unemployment price_decreased pending_listing

data:  VAR object var_stationary
Chi-squared = 13.368, df = 4, p-value = 0.009611
```

Co-integration

- Assumption of `ARIMA` and `TSLM+ARIMA` models are that the data are stationary
- For `VAR`, data don't *need* to be stationary *but* at some point they do
- In general, we expect that (related) time series should be *moving* together
- *Co-integration* is used to test whether this expectation is true

- Co-integration occurs with two or more *non-stationary* time series
    - long-run equilibrium
    - move together such that a linear combination makes the time series stationary
    - underlying common stochastic trend (stationary)

Goal: Find a stable long-run relationship between non-stationary variables such that the resultant time series *is* stationary

Formal definition

> *An $(n \times 1)$ vector of variables y_t is said to be cointegrated if at least one non-zero n-element vector β_i exists such that $\beta_i' y_t$ is trend-stationary. β_i is called a cointegrating vector. If r such linearly independent vectors β_i ($i = 1, \ldots, r$ exist, we say that y_t is cointegrated with cointegrating rank r. We then define the $(n \times r)$ matrix of cointegrating vectors $\beta = (\beta_1, \ldots, \beta_r$. The r elements of the vector $\beta' y_t$ are trend-stationary, and β is called the cointegrating matrix.*

– *Analysis of Integrated and Cointegrated Time Series with R* (p. 79)

Johansen Procedure

- Use *canonical correlations* to determine if whether there is enough multicollinearity to represent the relationships in a reduced space (r)

- **canonical correlation**: the correlation of the linear combination(s) between two sets of variables (i.e., past values of the variable itself and other variables)
- Based on a likelihood ratio of the H_0, eigenvalues are tested against zero (critical values are provided by functions in R and Python)

Use {urca} package

```r
# Cointegration
co_test <- ca.jo(
  # variables
  x = housing_ts[,c(
    "housing", "unemployment", "median_days",
    "price_decreased", "pending_listing"
  )],
  type = "trace", # tends to be more conservative
  K = 2, # lag -- same as your VAR model
  spec = "longrun", # generally use "longrun"
  ecdet = "trend", # trend-stationary
  # exogeneous dummy variables
  dumvar = housing_ts[,c("outlier", "pandemic")]
)

# Summary
co_summ <- summary(co_test)
```

- Determines whether variables can be combined in a linear way that makes them stationary

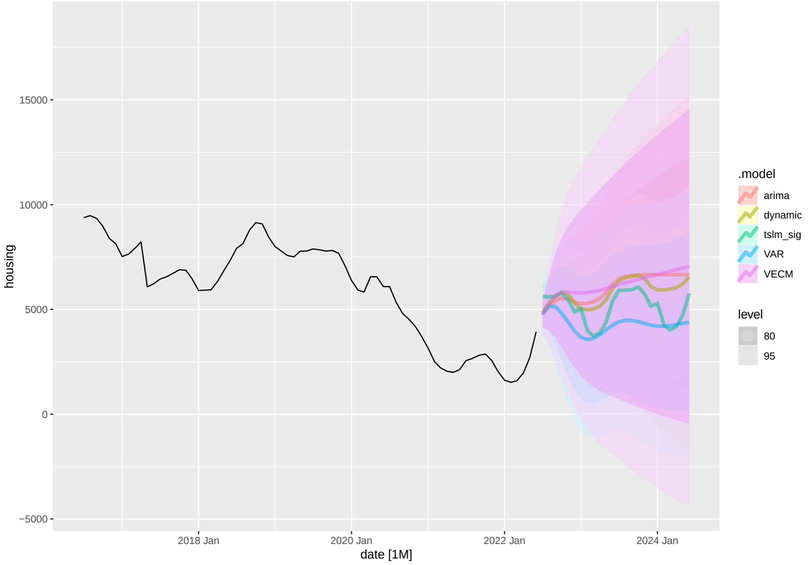
```
Eigenvalues

[1] 0.62263 0.43188 0.22021 0.07314 0.02275 0.00000


Critical Values

          10pct  5pct  1pct       test
r <= 4 | 10.49 12.25 16.26   1.610698
r <= 3 | 22.76 25.32 30.45   6.927180
r <= 2 | 39.06 42.44 48.45  24.337982
r <= 1 | 59.14 62.99 70.05  63.917368
r = 0  | 83.20 87.31 96.58 132.133493
```

Vector-Error-Correction Model

```r
# Convert VECM to VAR
vecm <- vars::vec2var(co_test, r = 1)

# Make dummy variable matrix
dummat <- matrix(
  rep(0, 2 * 24), nrow = 24
)
colnames(dummat) <- c("outlier", "pandemic")

# Forecast
vecm_fc <- predict(vecm, n.ahead = 24, dumvar = dummat)
```

Convert to `tsibble`

```r
# Get forecast values
fc_housing <- vecm_fc$fcst$housing

# Set up forecast as {fpp3} does
fc_vecm <- data.frame(
  .model = "VECM",
  date = var_fc$date,
  housing = distributional::dist_normal(
    mean = fc_housing[,"fcst"],
    sd = fc_housing[,"CI"]
  ),
  .mean = fc_housing[,"fcst"],
  fc[fc$.model == "tslm_sig", -c(1:4)]
) %>% as_tsibble(index = date)

# Add "housing" to dimnames
dimnames(fc_vecm$housing) <- "housing"

# Add to original forecast
fc <- bind_rows(fc, fc_vecm)
```



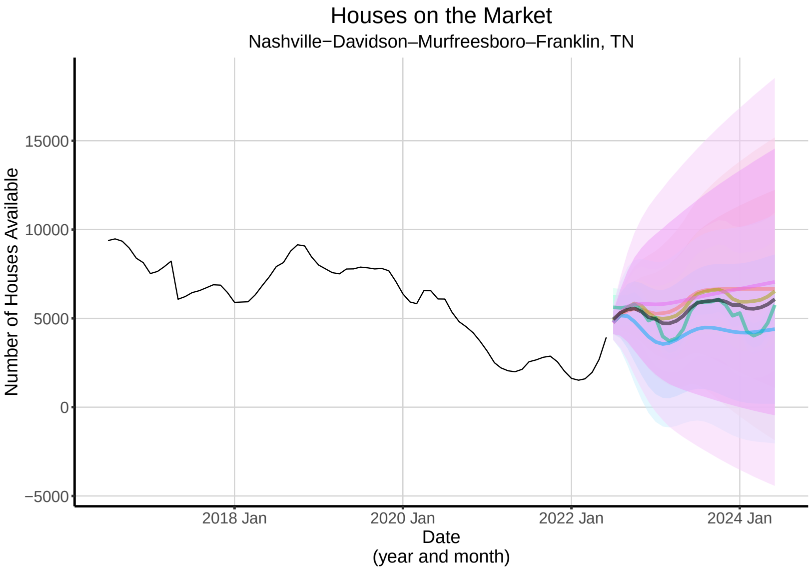
15000
10000
5000
0
−5000
housing
2018 Jan
2020 Jan
2022 Jan
2024 Jan
date [1M]
.model
arima
dynamic
tslm_sig
VAR
VECM
level
80
95

Best of All Estimates?

Why choose one forecast?

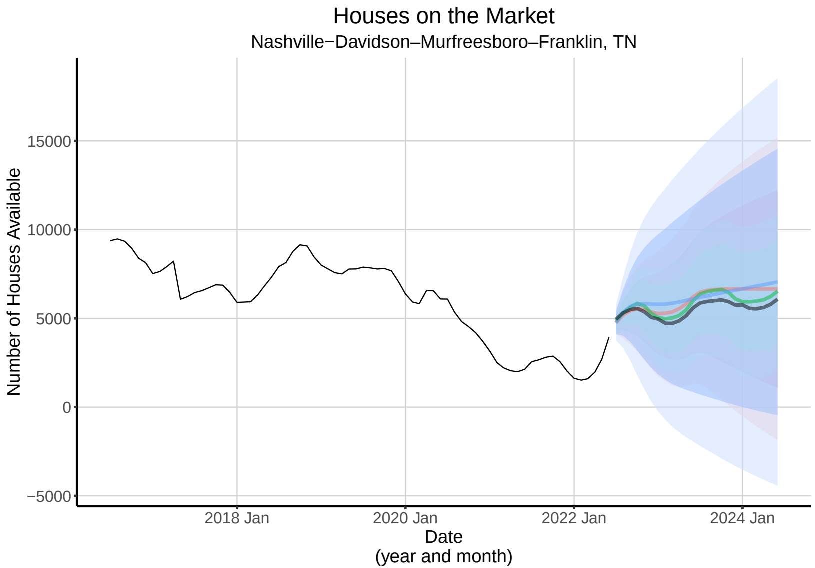
Averaging forecasts from different models can be more accurate than a single forecast

```r
# Create average of all models
average_fc <- fc %>%
  mutate(date_factor = factor(date)) %>%
  group_by(date_factor) %>%
  summarize(all_average = as.numeric(mean(.mean))) %>%
  select(all_average) %>%
  as_tsibble(key = NULL, index = date)
```


Houses on the Market
Nashville−Davidson–Murfreesboro–Franklin, TN
Number of Houses Available
15000
10000
5000
0
−5000
2018 Jan
2020 Jan
2022 Jan
2024 Jan
Date
(year and month)

Let's remove a few forecasts…

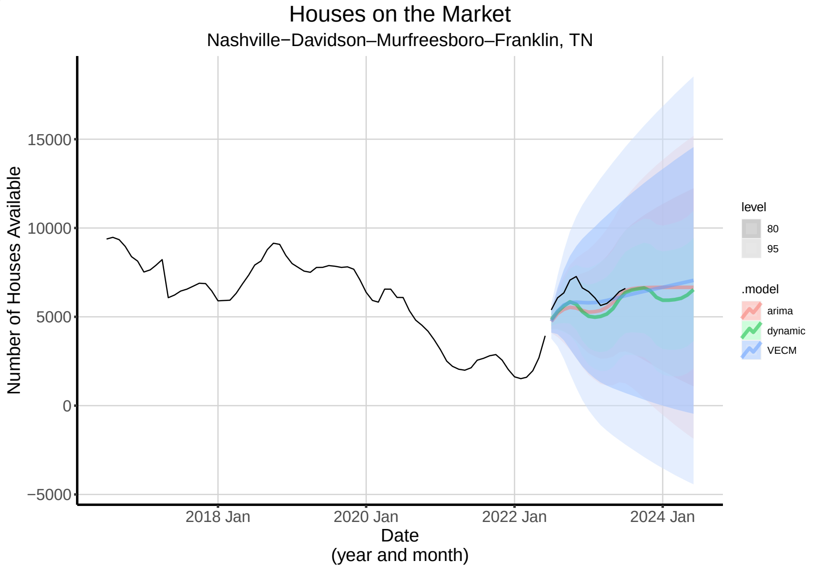
- We know VAR isn't stationary, so we can throw that forecast out
- We know plain TLSM is bad, so we can throw that out too


Houses on the Market
Nashville−Davidson–Murfreesboro–Franklin, TN
Number of Houses Available
15000
10000
5000
0
−5000
2018 Jan
2020 Jan
2022 Jan
2024 Jan
Date
(year and month)

It's been a year…

```r
# Load data
housing_validation <- read.csv("../data/housing_validation.csv")

# Convert date
housing_validation$date <- yearmonth(housing_validation$date)

# Convert to `tsibble`
housing_valid <- housing_validation %>%
  as_tsibble(index = date)
```

Multivariate Time Series | Next Two Years



Preference?

```r
# Point estimates
fc %>%
  filter(.model != "tslm_sig" & .model != "VAR") %>%
  accuracy(housing_valid)
```

```
# A tibble: 3 x 10
  .model  .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 VECM    Test   529.  712.  587.  8.02  9.05   NaN   NaN 0.776
2 arima   Test   772.  934.  772. 11.9  11.9    NaN   NaN 0.780
3 dynamic Test   854.  943.  854. 13.4  13.4    NaN   NaN 0.685
```

```r
# Distributional estimates
fc %>%
  filter(.model != "tslm_sig" & .model != "VAR") %>%
  accuracy(housing_valid, list(crps = CRPS))
```

```
# A tibble: 3 x 3
  .model  .type  crps
  <chr>   <chr> <dbl>
1 VECM    Test   764.
2 arima   Test   656.
3 dynamic Test   594.
```

Additional Resources

- Vector Autoregression
- Co-integration
- https://www.econometrics-with-r.org/
- Note on Why *Not* Use Granger Causality with VECM