Applied Economic Forecasting

3. Evaluation of Basic Forecasting Models

1. Forecasting Time Series
2. Residual diagnostics
3. Prediction intervals

Section 1

Forecasting Time Series

Our Forecasting Workflow

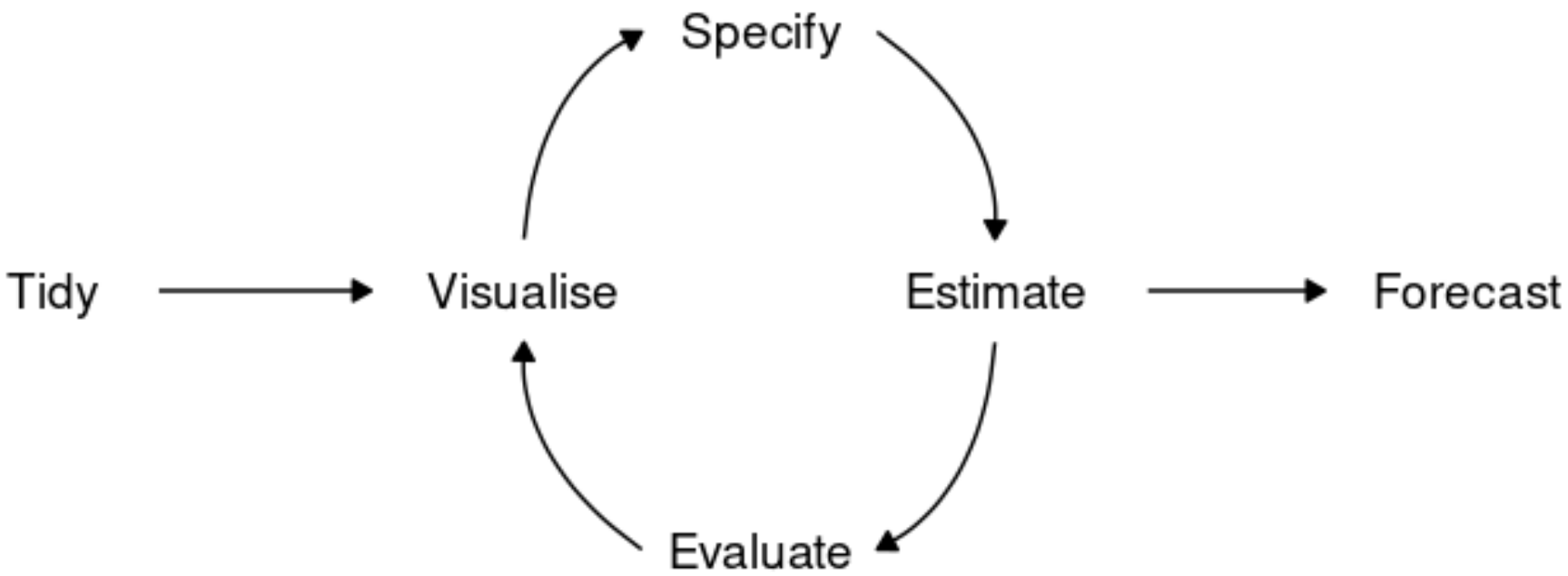


Figure 1: Basic Forceasting Workflow

Simplest forecasting methods

1. Average method

- Forecast of all future values is equal to mean of historical data $\{y_1, \ldots, y_T\}$.
- Forecasts: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$

2. Naïve method

- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{T+h|T} = y_T$.
- Consequence of efficient market hypothesis.

3. Seasonal naïve method

- Forecasts equal to **last value from same season** (e.g., the same month of the previous year).
- Forecasts: $\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$, where m = seasonal period and k is the integer part of $(h-1)/m$.

Simplest forecasting methods

4. Drift method

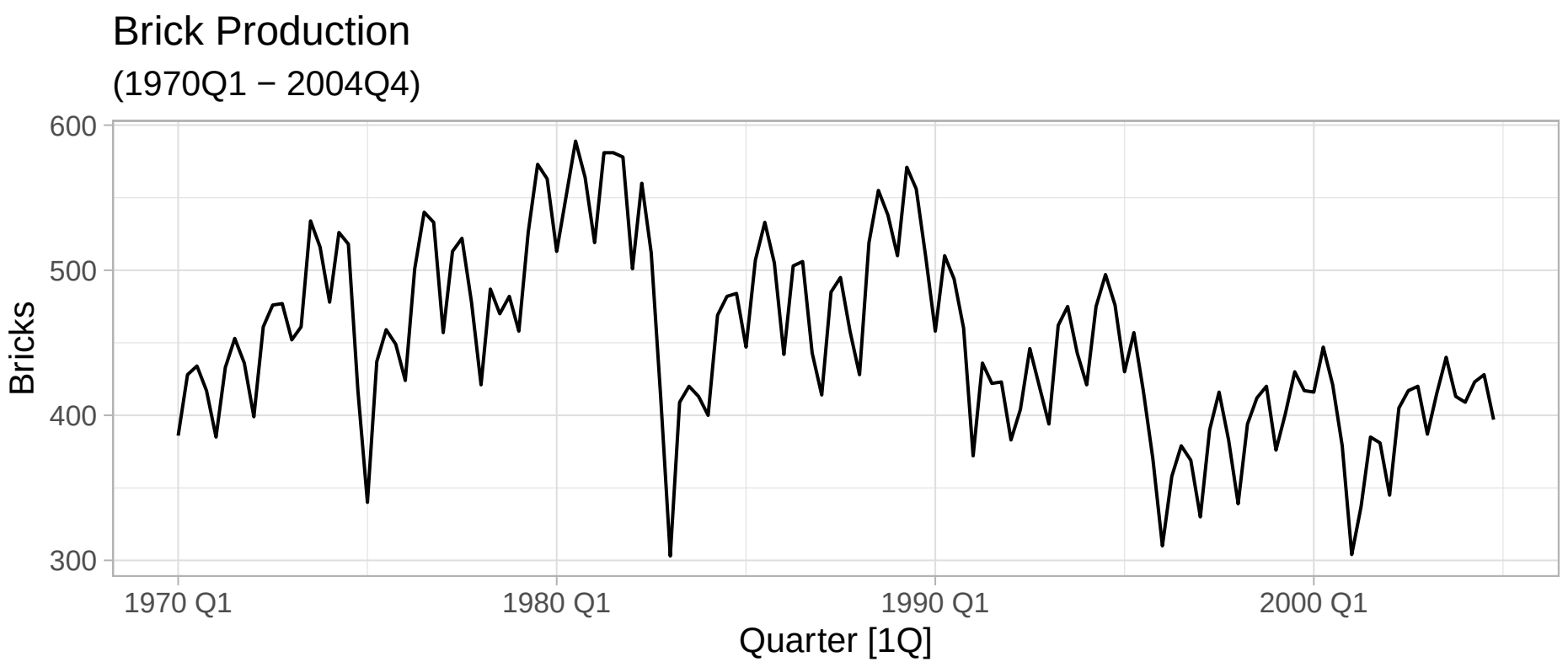
- Variation of the naïve method where forecasts are allowed to increase or decrease over time.
- Drift: The amount of change over time.
- Forecasts equal last value plus average average over the historical data.
- Forecasts:

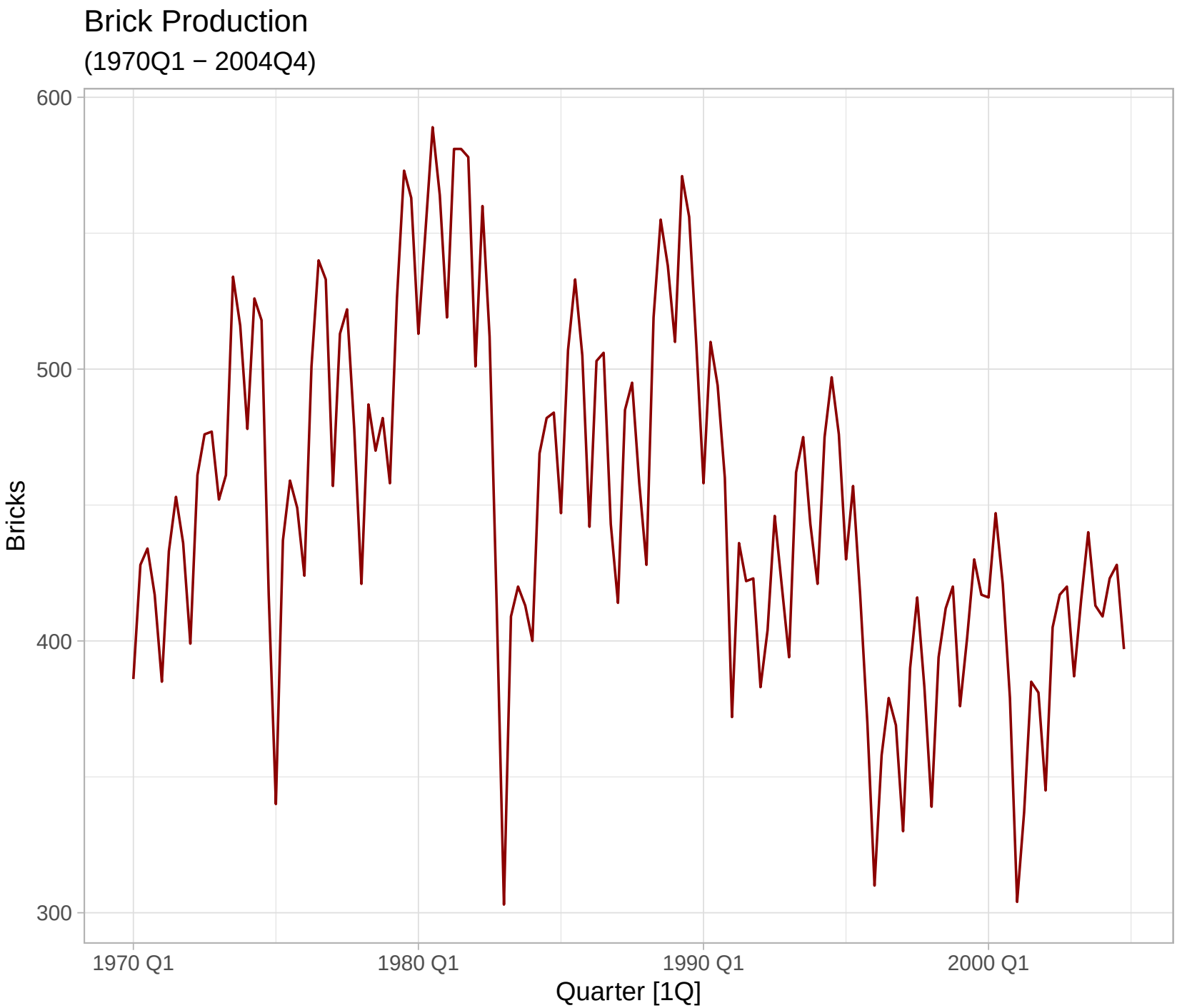
$$\begin{aligned}\hat{y}_{T+h|T} &= y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) \\ &= y_T + \frac{h}{T-1}(y_T - y_1).\end{aligned}$$

- Equivalent to extrapolating a line drawn between first and last observations.

Example: Brick Production

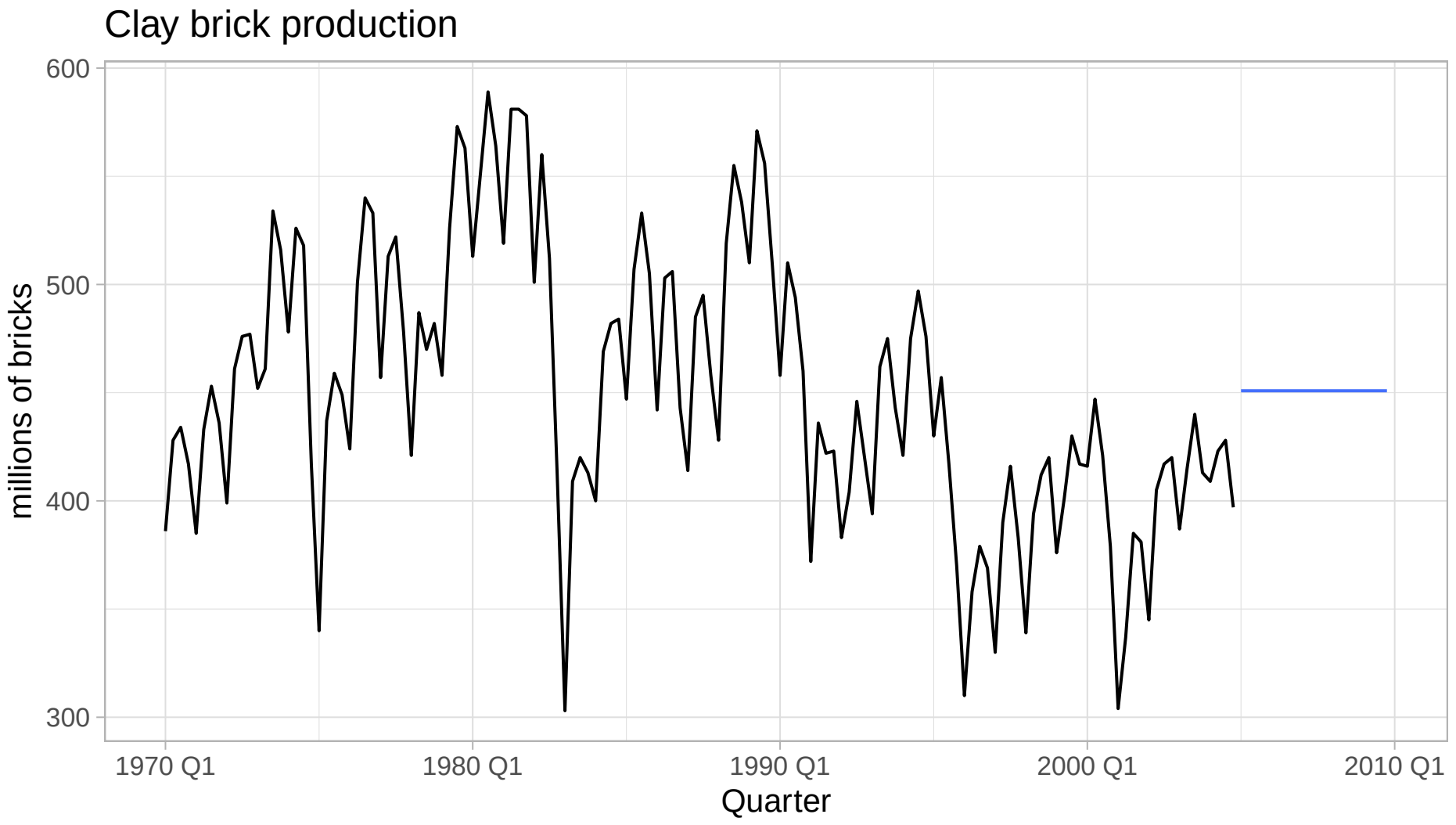
```
bricks <- aus_production %>%
  filter_index("1970 Q1" ~ "2004 Q4") %>%
  select(Bricks)

bricks %>% autoplot(Bricks) +
  labs(title = "Brick Production", subtitle = "(1970Q1 - 2004Q4)")
```




Brick Production
(1970Q1 − 2004Q4)
600
500
400
300
Bricks
1970 Q1
1980 Q1
1990 Q1
2000 Q1
Quarter [1Q]

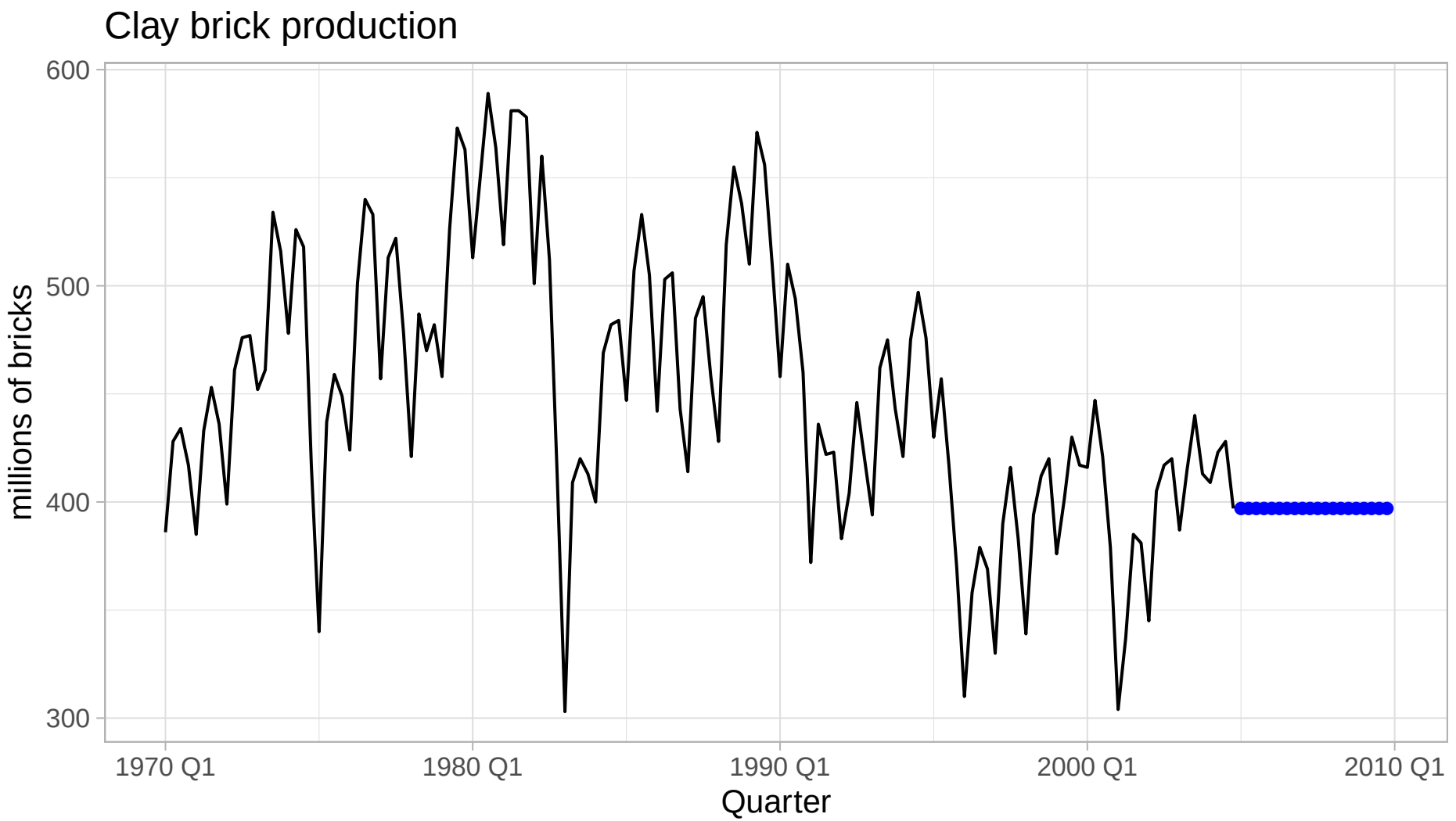
Example: Brick Production (MEAN)

```
fcst <- bricks %>% model(MEAN(Bricks)) %>% forecast(h = "5 years")
fcst %>% autoplot(bricks, level = NULL) +
  labs(y = "millions of bricks",
    title = "Clay brick production")
```



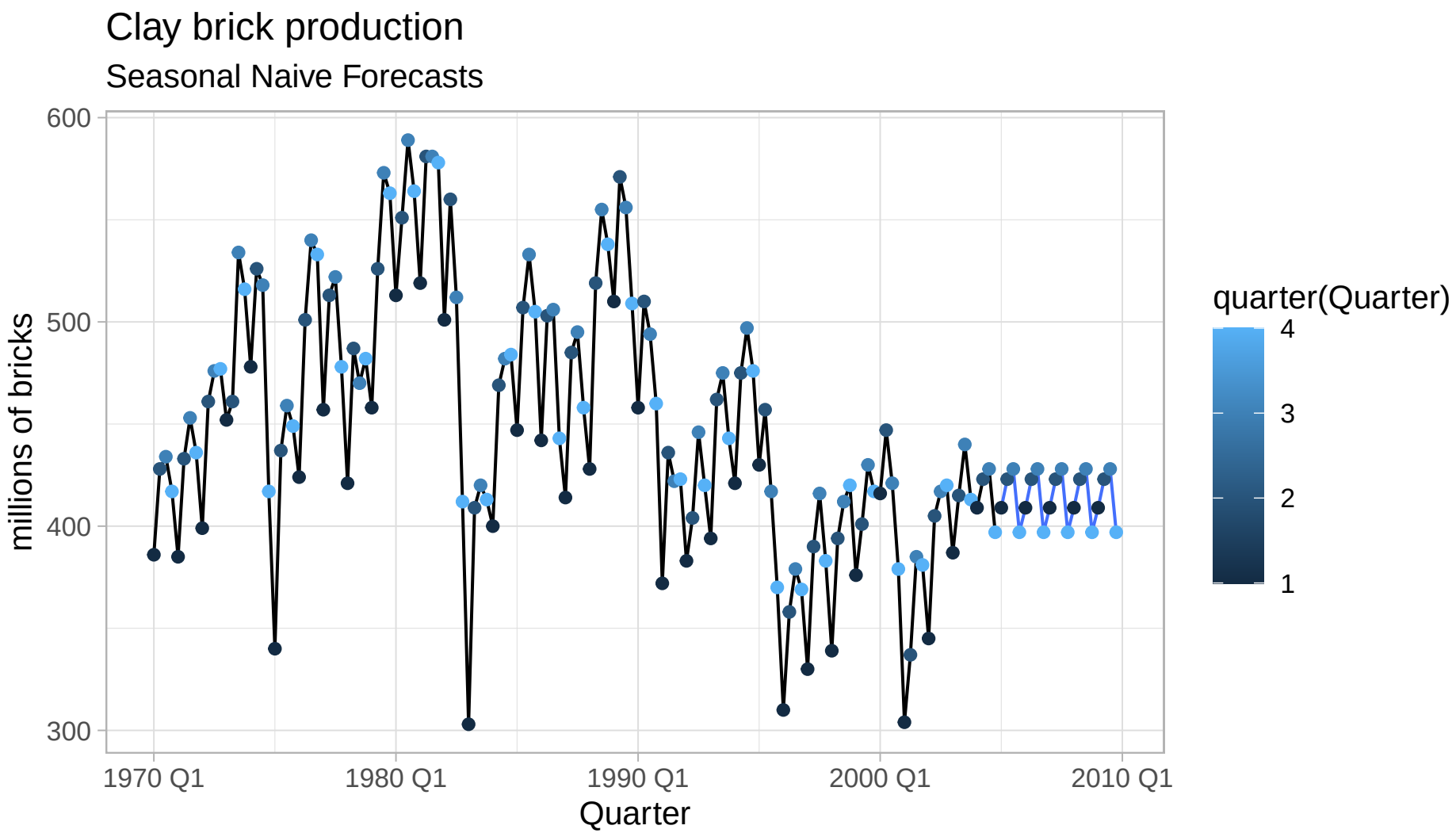
Example: Brick Production (NAIVE)

```
fcst <- bricks %>% model(NAIVE(Bricks)) %>% forecast(h = "5 years")

fcst %>% autoplot(bricks, level = NULL) +
  geom_point(aes(y = .mean), data = fcst, col = "blue") +
  labs(y = "millions of bricks", title = "Clay brick production")
```



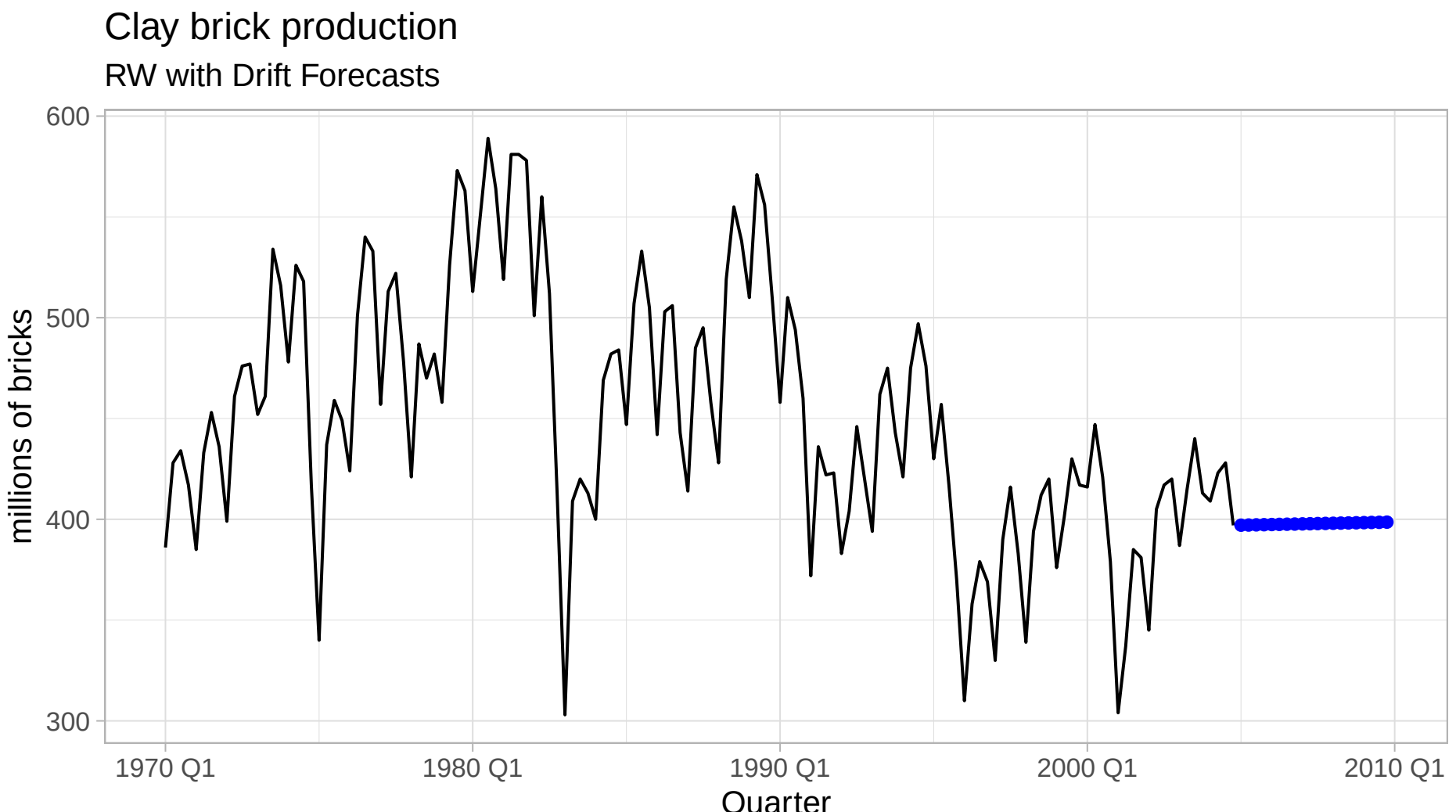
Example: Brick Production (SEASONAL NAIVE)

```
fcst <- bricks %>% model(SNAIVE(Bricks ~ lag("1 year"))) %>%
  forecast(h = "5 years")
fcst %>% autoplot(bricks, level = NULL) +
  geom_point(aes(y = Bricks, color = quarter(Quarter)), data = bricks) +
  geom_point(aes(y = .mean, color = quarter(Quarter)), data = fcst) +
  labs(y = "millions of bricks", title = "Clay brick production", subtitle = "Seasonal Naive Forecasts")
```

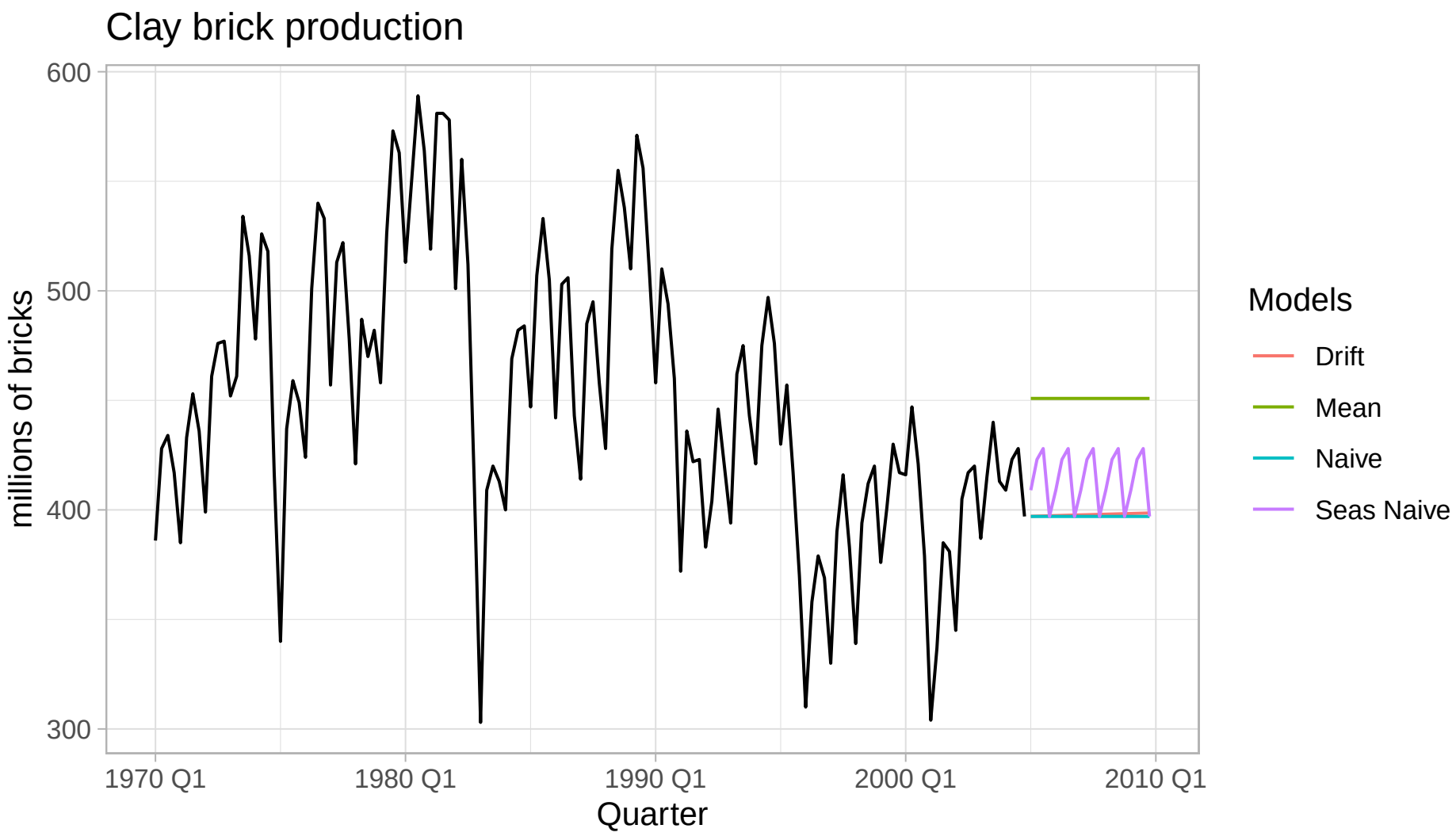


Example: Brick Production (RANDOM WALK W. DRIFT)

```
fcst <- bricks %>% model(RW(Bricks ~ drift())) %>% forecast(h = "5 years")
fcst %>% autoplot(bricks, level = NULL) +
  geom_point(aes(y = .mean), data = fcst, col = "blue") +
  labs(y = "millions of bricks", title = "Clay brick production", subtitle = "RW with Drift Forecasts")
```

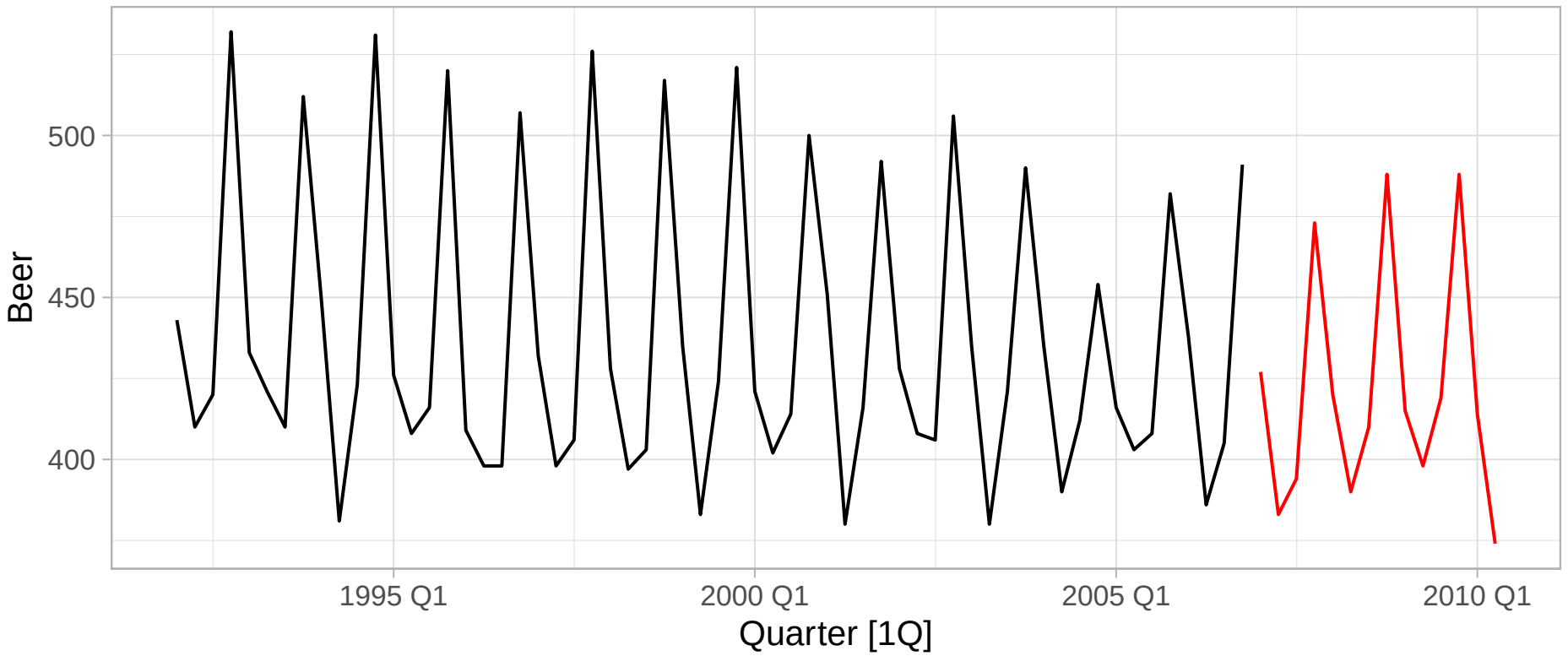


```r
fcst <- bricks %>% model(Mean = MEAN(Bricks),
                         Naive = NAIVE(Bricks),
                         `Seas Naive` = SNAIVE(Bricks),
                         Drift = RW(Bricks ~ drift())) %>% forecast(h = "5 years")
fcst %>% autoplot(bricks, level = NULL) +
  labs(y = "millions of bricks", title = "Clay brick production") +
  guides(colour = guide_legend(title = "Models"))
```

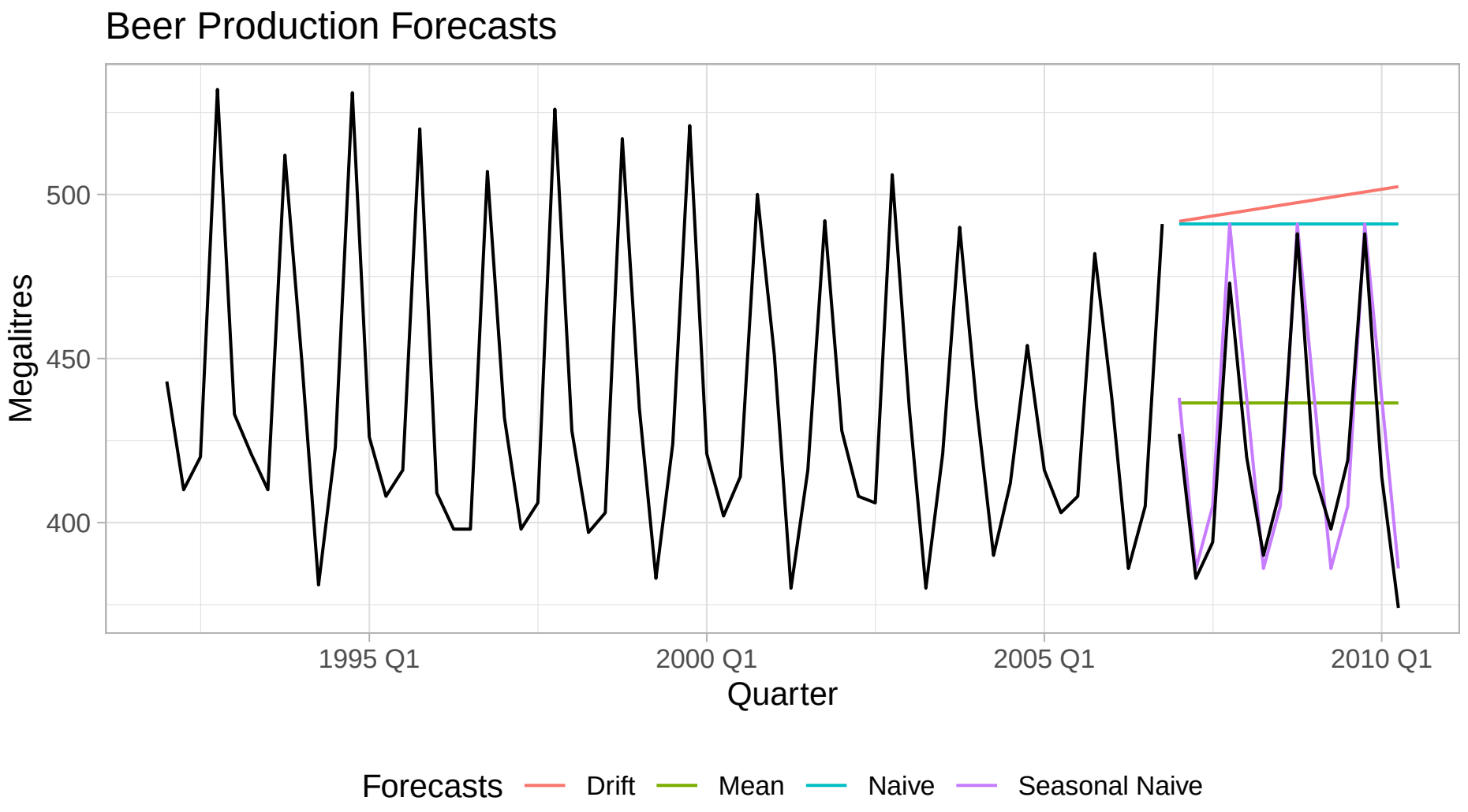


Forecast Comparions

```
beer <- aus_production %>% select(Beer, Quarter)
# Training Data: 1992 Q1 -- 2006 Q4
beer.train <- beer %>% filter_index("1992 Q1" ~ "2006 Q4")
# Test Data: 2007 Q1 -- 2010 Q2
beer.test <- beer %>% filter_index("2007 Q1" ~.)

beer.train %>% autoplot(Beer) +
  autolayer(beer.test,Beer, col = "red")
```



```r
mods <- beer.train %>% model(Mean = MEAN(Beer),
                      Naive = RW(Beer),
                      `Seasonal Naive` = SNAIVE(Beer),
                      Drift = RW(Beer ~ drift())) %>% forecast(h = 14)
mods %>% autoplot(beer.train, level = NULL) +
  autolayer(beer.test, Beer, col = "black") +
  labs(title = "Beer Production Forecasts", y = "Megalitres") +
  guides(colour = guide_legend(title = "Forecasts")) + theme(legend.position = "bottom")
```



Try This!!

Using the `MEAN`, `NAIVE`, and `RW` with `drift` functions in the previous slides, attempt to produce forecasts for the next 100 days for `FB`'s closing prices stored in `gafa_stock`.

Do you encounter any issues?

Try This (Take II)!!

You should have gotten an error to this effect:

```
Error in `mutate()`:
! Problem while computing `MEAN(Close) = (function
  (object, ...) ...`.
Caused by error in `abort_if_irregular()`:
! `new_data.tbl_ts(.data, round(n))` can't handle tsibble of ir
```

- This occurs because only trading days will appear in the dataset so sometimes the data is spaced by 1-day and at other points 2 (think weekends or single day holidays).
- It would be better to "reindex" the series.

Use the code below to create an index to capture the position of the trading day in the data set.

```
FB <- gafa_stock %>% filter(Symbol == "FB") %>%
  mutate(trade_day = row_number()) %>%
  update_tsibble(index = trade_day, regular = TRUE)
```

Try This!!

Reattempting the task

Now, use the `MEAN`, `NAIVE`, and `RW` with `drift` functions to produce forecasts for the next 100 days for the Close series in `FB`.

Additional Practice

How about using data from 2016 to train your model and forecast the first month in 2017?

You could proceed as follows:

i. Create a training set that stores the values in 2016.
ii. Create a test dataset that stores values from `January 2017`.
iii. Produce the forecasts for `h` that matches the length of the test data.
iv. Plot the forecasts from the 3 models against the test data.

Section 2

Residual diagnostics

Fitted values

- $\hat{y}_{t|t-1}$ is the forecast of y_t based on observations $y_1, \ldots, y_t$.
- We call these "fitted values".
- Sometimes drop the subscript: $\hat{y}_t \equiv \hat{y}_{t|t-1}$.
- Often not true forecasts since parameters are estimated on all data.

For example:

- $\hat{y}_t = \bar{y}$ for average method.
- $\hat{y}_t = y_{t-1} + (y_T - y_1)/(T-1)$ for drift method.

Forecasting residuals

Residuals in forecasting: difference between observed value and its fitted value: $e_t = y_t - \hat{y}_{t|t-1}$.

Assumptions

1. $\{e_t\}$ uncorrelated.
   - **If they are not, then information left in residuals that should be used in computing forecasts.**
2. $\{e_t\}$ have mean zero.
   - **If they do not, then forecasts are biased.**

Useful (but not necessary) properties (for prediction intervals)

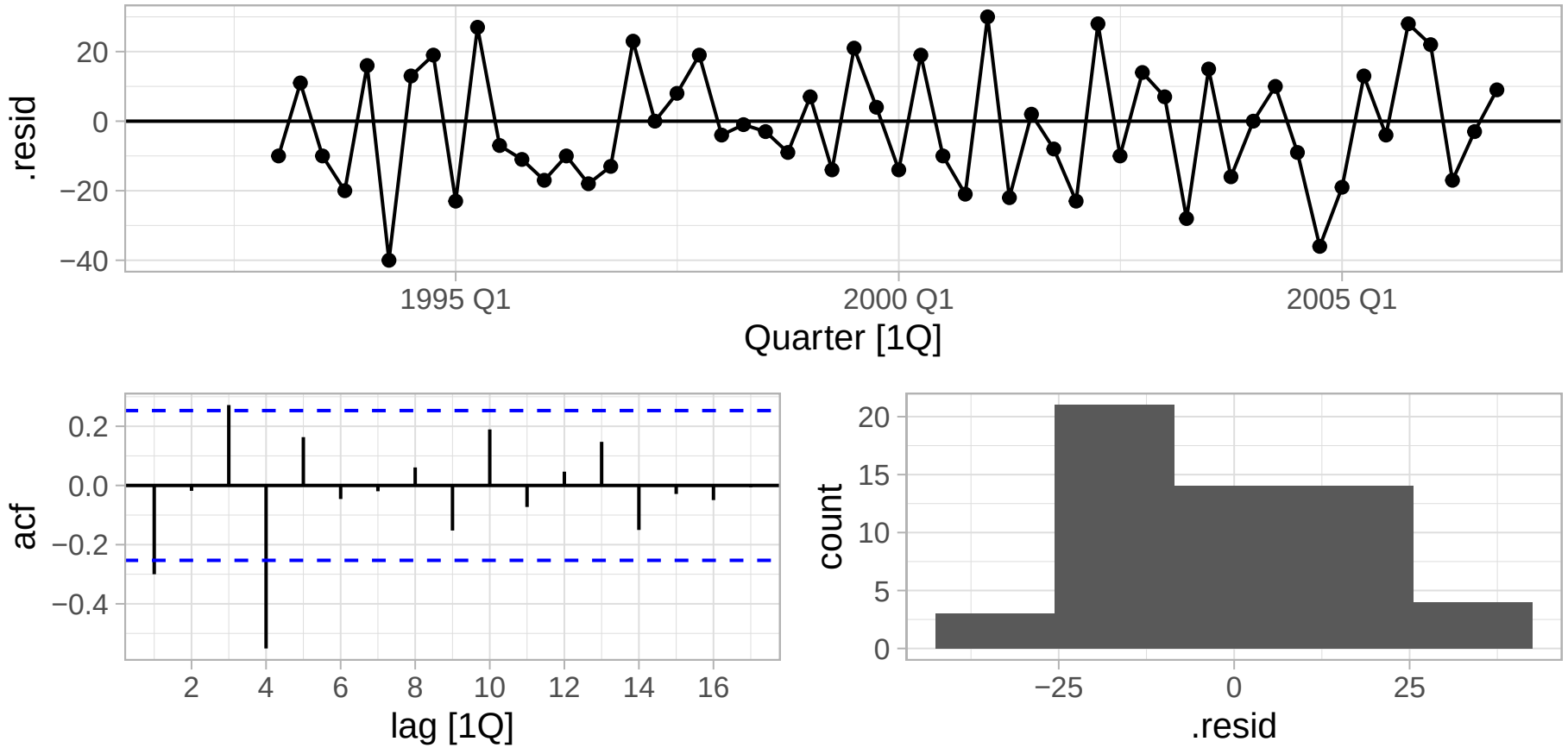
3. $\{e_t\}$ have constant variance– Homoskedasticity
4. $\{e_t\}$ are normally distributed.

Revisting the Beer series

```
mods.beer <- beer.train %>% model(Mean = MEAN(Beer),
                      Naive = RW(Beer),
                      `Seasonal Naive` = SNAIVE(Beer),
                      Drift = RW(Beer ~ drift()))
beer.aug <- mods.beer %>% augment()

beer.aug %>% autoplot(.resid) + geom_abline(slope = 0) +
  labs(title = "Beer: Forecast Model residuals")
```

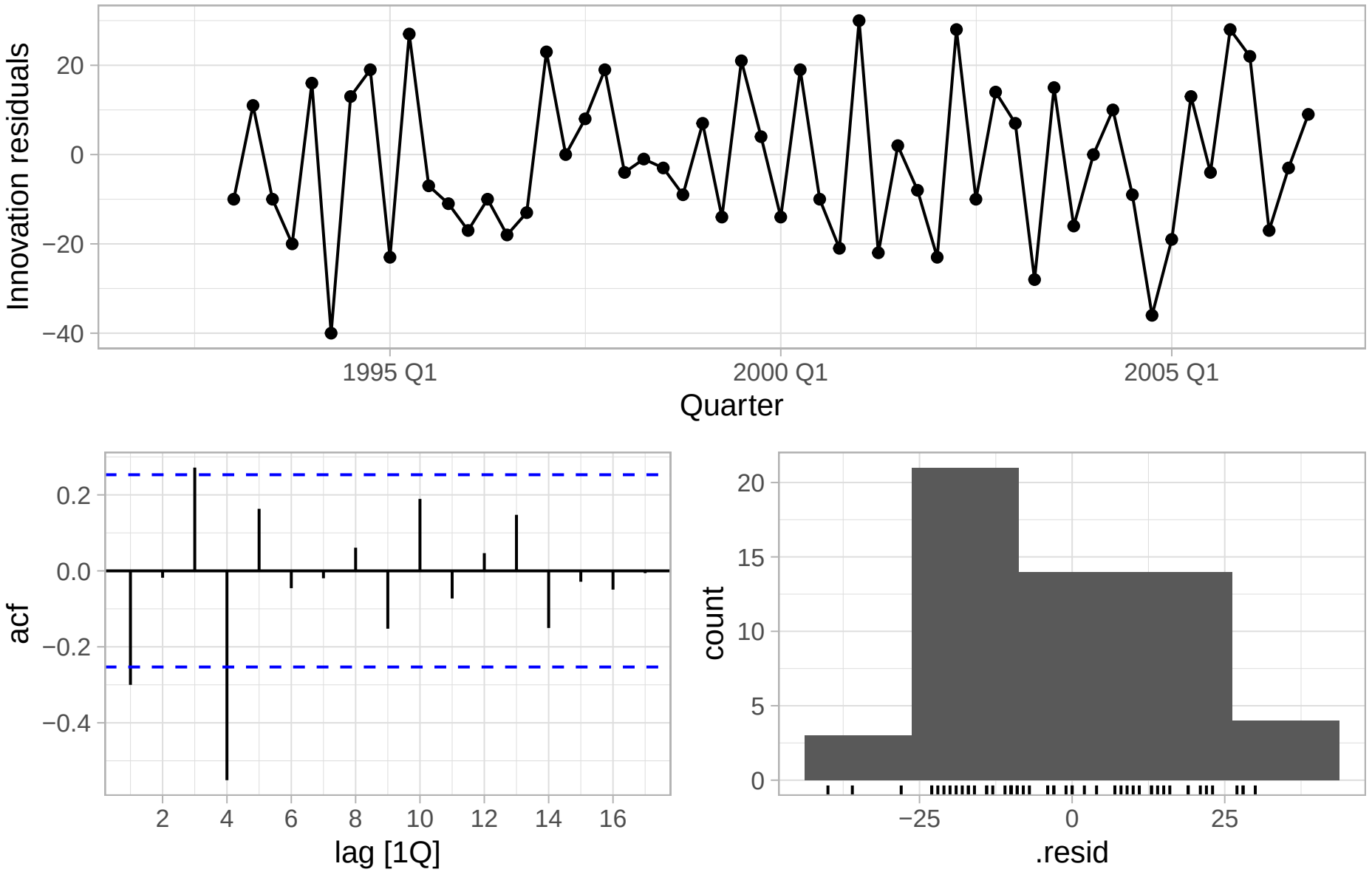
Beer: Seasonal Naive Model Residuals

```
b1 <- beer.aug %>% filter(.model == "Seasonal Naive") %>% autoplot(.resid) +
  geom_point() +
  geom_abline(slope = 0)
b2 <- beer.aug %>% filter(.model == "Seasonal Naive") %>% ACF(.resid) %>%  autoplot()
b3 <- beer.aug %>% filter(.model == "Seasonal Naive") %>% ggplot(aes(x = .resid)) +
  geom_histogram(binwidth = 17)

wrap_plots(b1,b2,b3, design = "AA\nBC")
```



Beer: Seasonal Naive Model Residuals

```
mods.beer %>% select(`Seasonal Naive`) %>% gg_tsresiduals()
```



ACF of residuals

- We assume that the residuals are white noise (uncorrelated, mean zero, constant variance). If they are not, then there is information left in the residuals that should be used in computing forecasts.
- So a standard residual diagnostic is to check the ACF of the residuals of a forecasting method.
- We *expect* these to look like white noise.

Portmanteau tests for autocorrelation

Consider a ***whole set*** of r_k values, rather than assessing each individually. The idea of this test is to see whether the **set** is (**jointly**) significantly different from a zero set.

1. Box-Pierce test

$$Q = T\sum_{k=1}^{\ell} r_k^2 \sim \chi^2_\ell$$

where ℓ is max lag being considered (usually 10 for nonseasonal data and $2m$ for seasonal data) and T is number of observations.

- If each r_k close to zero, Q will be **small**.
- If some r_k values large (positive or negative), Q will be **large**. Evidence that the autocorrelations might not be white noise.

Portmanteau tests for autocorrelation

Consider a ***whole set*** of r_k values, and develop a test to see whether the **set** is (jointly) significantly different from a zero set.

2. Ljung-Box test

$$Q^* = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2 \sim \chi^2_\ell$$

where ℓ is max lag being considered and T is number of observations.

- Lung-Box has better performance, especially in small samples.

Portmanteau tests for autocorrelation

H_0: No Autocorrelation up to lag ℓ

H_1: Autocorrelation up to lag ℓ

```
# Data is seasonal so l = 2*m => 2*4
beer.aug %>% filter(.model == "Seasonal Naive") %>%
  features(.innov, box_pierce, lag = 8)
```

```
## # A tibble: 1 x 3
##   .model          bp_stat bp_pvalue
##   <chr>             <dbl>     <dbl>
## 1 Seasonal Naive     28.0  0.000468
```

```
beer.aug %>% filter(.model == "Seasonal Naive") %>%
  features(.innov, ljung_box, lag = 8)
```

```
## # A tibble: 1 x 3
##   .model          lb_stat lb_pvalue
##   <chr>             <dbl>     <dbl>
## 1 Seasonal Naive     30.9  0.000144
```

Portmanteau tests for autocorrelation

Combing both...

```
beer.aug %>%
  filter(.model == "Seasonal Naive") %>%
  features(.innov, list(box_pierce, ljung_box), lag = 8) %>%
  select(.model, ends_with("_pvalue"))
```

```
## # A tibble: 1 x 3
##   .model         bp_pvalue lb_pvalue
##   <chr>              <dbl>     <dbl>
## 1 Seasonal Naive  0.000468  0.000144
```

Measures of forecast accuracy

Forecast Error

A forecast "error" is the difference between an observed value and its forecast. **Here "error" does not mean a mistake, it means the unpredictable part of an observation.** It can be written as

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T}$$

where the training data is given by $\{y_1, \ldots, y_T\}$ and the test data is given by $\{y_{T+1}, y_{T+2}, \ldots\}$.

Residuals vs Forecast Errors

1. **Residuals** are calculated on the **training set** while **forecast errors** are calculated on the **test set**.
2. Residuals are based on one-step forecasts while forecast errors can involve multi-step forecasts.

Measures of forecast accuracy

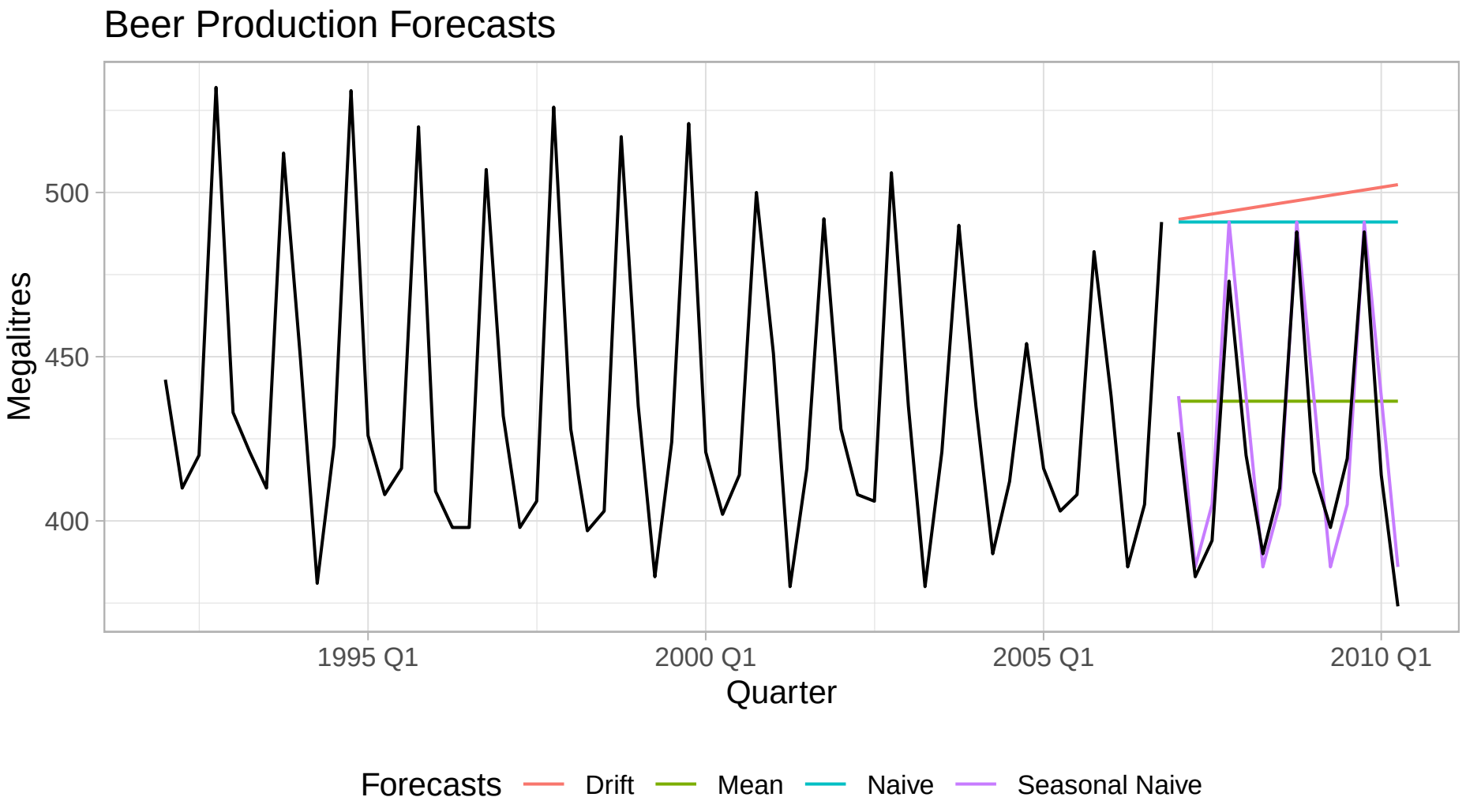
The accuracy of our forecasts can be measured in several ways:

$$\text{MAE} = \text{mean}(|e_t|)$$
$$\text{MSE} = \text{mean}(e_t^2)$$
$$\text{RMSE} = \sqrt{\text{mean}(e_t^2)}$$
$$\text{MAPE} = 100\text{mean}\left(\frac{|e_t|}{|y_t|}\right)$$

- MAE, MSE, RMSE are all scale dependent.
  - Therefore, they cannot be used to make comparisons between series that involve different units.
  - A forecast method that minimizes the MAE will lead to forecasts of the median, while minimizing the RMSE will lead to forecasts of the mean.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all t, and y has a natural zero.

Measures of forecast accuracy

```
mods %>% autoplot(beer.train, level = NULL) +
  autolayer(beer.test, Beer, col = "black") +
  labs(title = "Beer Production Forecasts", y = "Megalitres") +
  guides(colour = guide_legend(title = "Forecasts")) + theme(legend.position = "bottom")
```



Measures of forecast accuracy

```
# Compare forecast results against test data
accuracy(mods, beer) %>% select(.model, RMSE, MAE, MAPE, MASE) %>%
  knitr::kable(digits = 3)
```

.model	RMSE	MAE	MAPE	MASE
Drift	84.114	76.173	18.897	4.785
Mean	38.890	35.471	8.514	2.228
Naive	78.623	70.071	17.439	4.401
Seasonal Naive	13.488	11.500	2.759	0.722

- The preferred model is the one with the lowest statistic under each criteria.
- Selection Criteria may not always agree on the same model. For example, the MAPE might chose model A, but the MAE chose model B.

Your Turn

Can you produce the accuracy statistics for your `FB` series (from the Additional Practice Exercise) earlier?

Section 3

Prediction intervals

Prediction intervals

- A forecast $\hat{y}_{T+h|T}$ is (usually) the mean of the conditional distribution $y_{T+h} \mid y_1, \dots, y_T$.
- A prediction interval gives a region within which we expect y_{T+h} to lie with a specified probability.
- The value of prediction intervals is that they express the uncertainty in the forecasts. Point forecasts are not helpful in determining how accurate the forecasts are.
- Assuming forecast errors are normally distributed, then a 95% PI is

$$\hat{y}_{T+h|T} \pm 1.96\hat{\sigma}_h$$

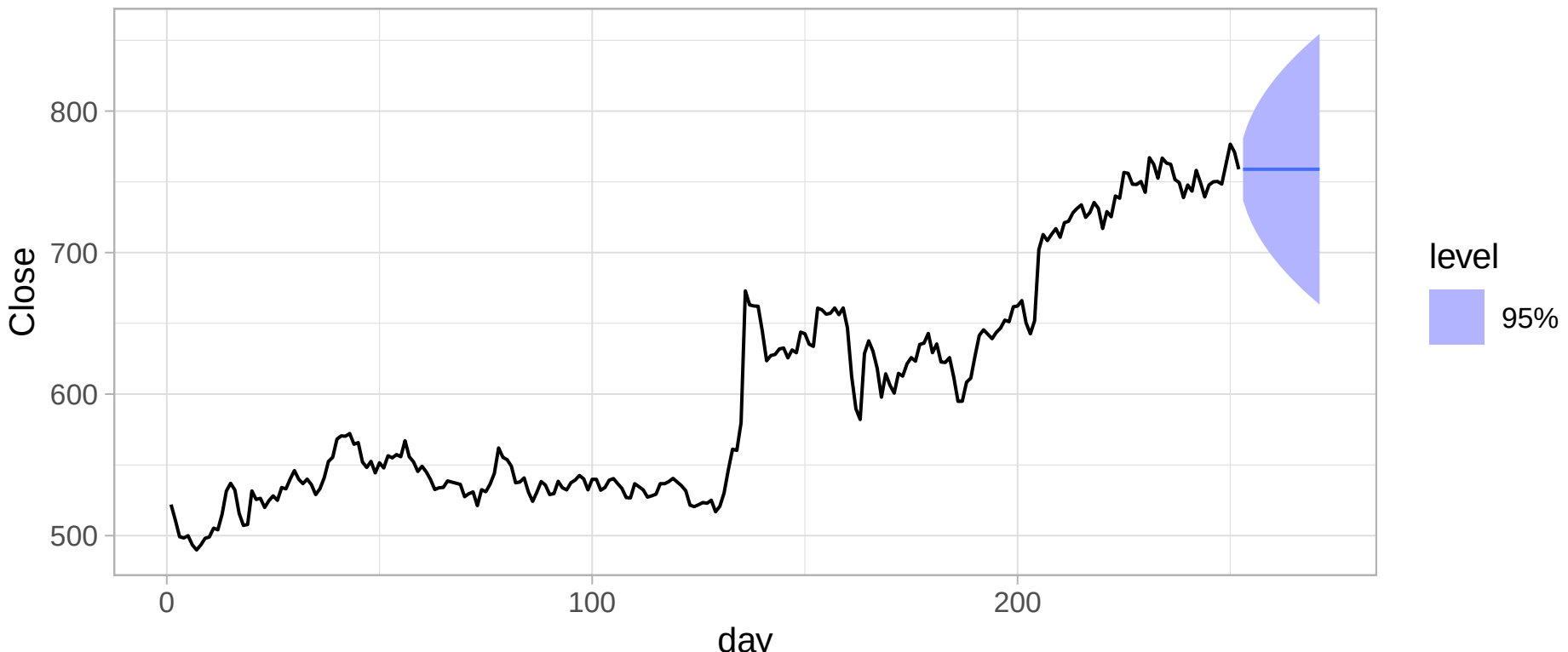
where $\hat{\sigma}_h$ is the st dev of the h-step distribution.

- When $h = 1$, $\hat{\sigma}_h$ can be estimated from the residuals.

Prediction intervals

Naive forecast with prediction interval:

```
goog <- gafa_stock %>% filter(Symbol == "GOOG") %>%
  filter_index("2015") %>% mutate(day = row_number()) %>%
  update_tsibble(index = day, regular = TRUE)

goog %>% model(NAIVE(Close)) %>% forecast(h = 19)%>%
  autoplot(goog, level = 95)
```



Prediction intervals

Naive forecast with prediction interval:

```
goog %>% model(NAIVE(Close)) %>%  forecast(h = 19) %>%  hilo(level = 95) %>% select("95%")
```

```
## # A tsibble: 19 x 2 [1]
##                      `95%`   day
##                     <hilo> <dbl>
##  1 [736.9488, 780.8112]95   253
##  2 [727.8646, 789.8954]95   254
##  3 [720.8941, 796.8659]95   255
##  4 [715.0176, 802.7424]95   256
##  5 [709.8404, 807.9196]95   257
##  6 [705.1598, 812.6002]95   258
##  7 [700.8556, 816.9045]95   259
##  8 [696.8493, 820.9108]95   260
##  9 [693.0865, 824.6735]95   261
## 10 [689.5275, 828.2325]95   262
## 11 [686.1425, 831.6175]95   263
## 12 [682.9082, 834.8518]95   264
## 13 [679.8060, 837.9540]95   265
## 14 [676.8210, 840.9390]95   266
## 15 [673.9409, 843.8191]95   267
## 16 [671.1553, 846.6047]95   268
## 17 [668.4554, 849.3046]95   269
## 18 [665.8339, 851.9261]95   270
## 19 [663.2842, 854.4758]95   271
```

Prediction intervals

Assume residuals are normal, uncorrelated, sd $= \hat{\sigma}$:

Mean forecasts:	$\hat{\sigma}_h = \hat{\sigma}\sqrt{1+1/T}$
Naïve forecasts:	$\hat{\sigma}_h = \hat{\sigma}\sqrt{h}$
Seasonal naïve forecasts	$\hat{\sigma}_h = \hat{\sigma}\sqrt{k+1}$
Drift forecasts:	$\hat{\sigma}_h = \hat{\sigma}\sqrt{h(1+h/(T-1))}.$

where k is the integer part of $(h-1)/m$.

Note that when $h=1$ and T is large, these all give the same approximate value $\hat{\sigma}$.

Bootstrapped Prediction Intervals

- Relaxes the assumption of normally distributed errors.
- Keeps the assumption of uncorrelated errors and constant variance.

A one-step forecast error is defined as $e_t = y_t - \hat{y}_{t|t-1}$. We can re-write this as $y_t = \hat{y}_{t|t-1} + e_t$.

So we can therefore simulate the next observation of a time series using $y_{T+1} = \hat{y}_{T+1|T} + e_{T+1}$.

Assuming that the future errors will be similar to past errors, we can replace e_{T+1} by sampling from the collection of errors we have seen in the past (i.e., the residuals).

Doing this repeatedly, we obtain many possible futures.

Bootstrapped Prediction Intervals

```
fit <- goog %>% model(NAIVE(Close))
sim <- fit %>% generate(h = 30, times = 5, bootstrap = TRUE)
sim
```

```
## # A tsibble: 150 x 5 [1]
## # Key:       Symbol, .model, .rep [5]
##    Symbol .model          day .rep   .sim
##    <chr>  <chr>         <dbl> <chr> <dbl>
##  1 GOOG   NAIVE(Close)   253 1      851.
##  2 GOOG   NAIVE(Close)   254 1      858.
##  3 GOOG   NAIVE(Close)   255 1      843.
##  4 GOOG   NAIVE(Close)   256 1      844.
##  5 GOOG   NAIVE(Close)   257 1      860.
##  6 GOOG   NAIVE(Close)   258 1      850.
##  7 GOOG   NAIVE(Close)   259 1      839.
##  8 GOOG   NAIVE(Close)   260 1      842.
##  9 GOOG   NAIVE(Close)   261 1      838.
```

Bootstrapped Prediction Interval

```
goog %>% ggplot(aes(x = day)) +  geom_line(aes(y = Close)) +
  geom_line(aes(y = .sim, colour = as.factor(.rep)), data = sim) +
  labs(title="Google daily closing stock price", y="$US" ) + guides(colour = "none")
```

Bootstrapped Prediction Interval

```
bs.goog <- goog %>% model(NAIVE(Close)) %>% forecast(h = 30, bootstrap = TRUE)

bs.goog %>% autoplot(goog) + labs(title = "Google: Bootstrapped PI")
```