Surge Pricing

Applying segmented regression to marketplace pricing

Clara Yuan | Research Scientist | Convoy

Cascadia R Conference | June 8, 2018


CONVOY
OUR MISSION: TRANSPORTING THE WORLD WITH ENDLESS CAPACITY AND ZERO WASTE

Freight brokering

Two-sided marketplace

- Shippers: Companies who need to move freight
- Carriers: Companies with trucks who can move freight

Shipment lifecycle

- Shipper tenders a load to Convoy
- Convoy accepts the tender
- Convoy matches the shipment to a carrier
- Carrier delivers the load

Marketplace pricing

- Shippers pay Convoy for the tender (`price`)
- Convoy pays carriers for the shipment (`cost`)
- Convoy predicts cost at the time of tender (`expected_cost`)

Surge pricing motivation

“It’s been a really unusual year. Typical summers are big; this one was huge.” ~ FreightWaves CEO Craig Fuller

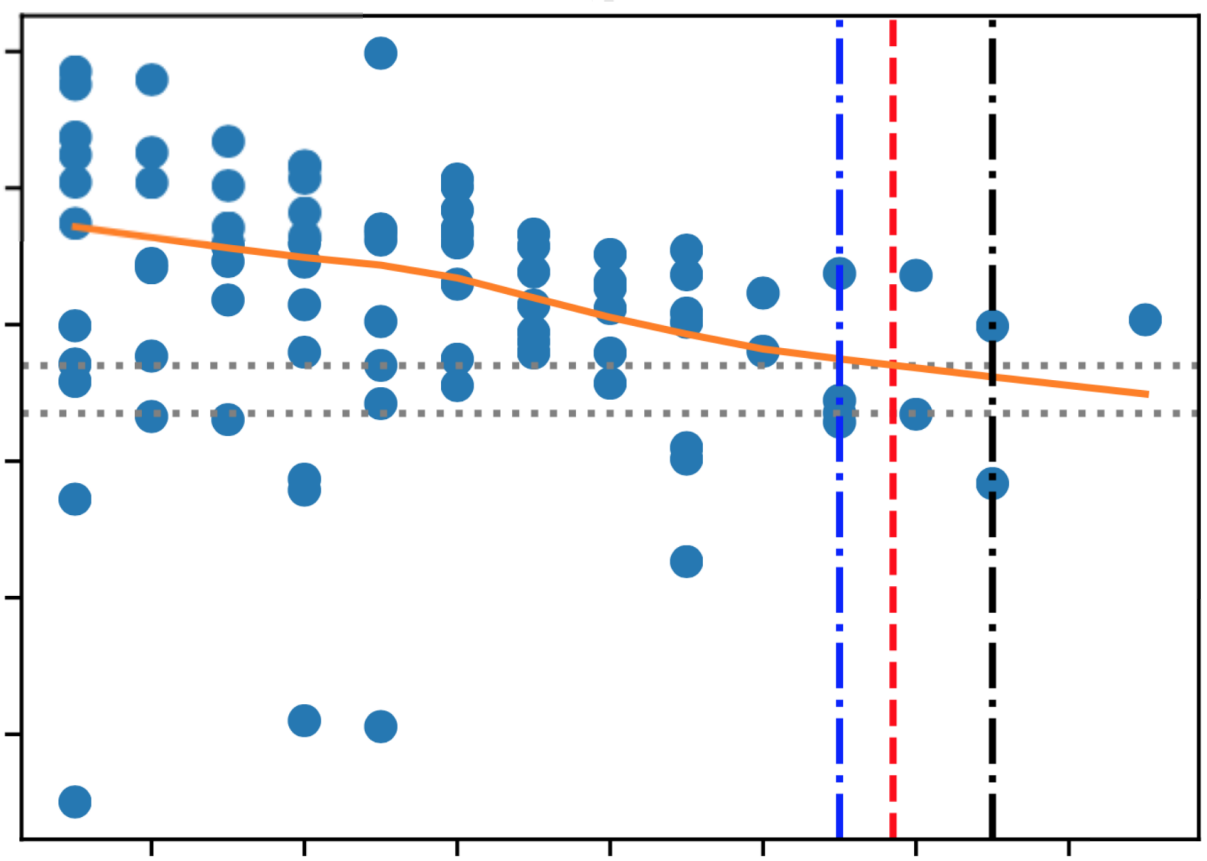
In the summers of 2017 and 2018, carrier supply was much tighter than usual, and hence costs were much higher than expected.

If Convoy can anticipate tight conditions, and therefore higher than expected costs, then we can pass on higher prices to shippers to reduce our downside.

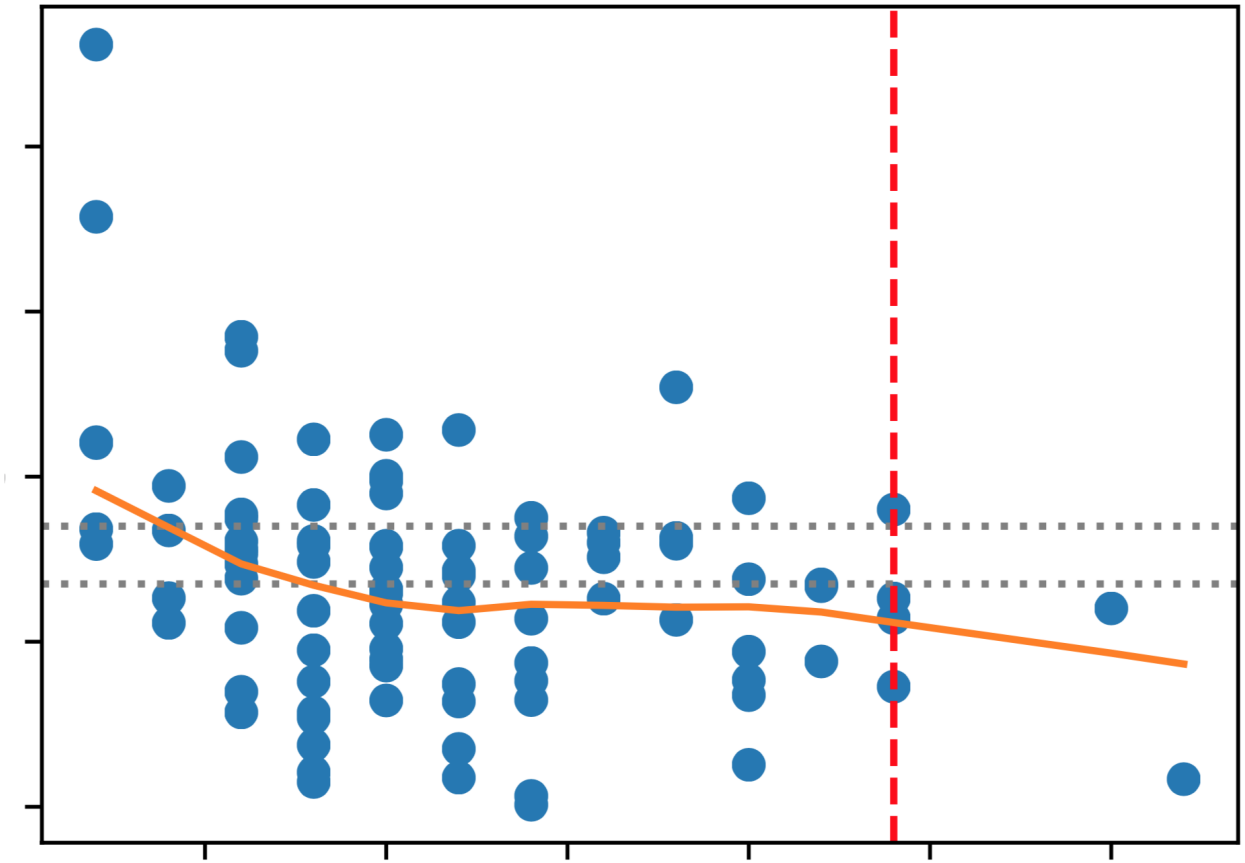
Surge pricing approach

- Estimate ‘stable capacity’ by market, the shipment volume on a given day beyond which we consider the market to be under surge conditions
- Estimate ‘surge premium’, the predicted error in our expected cost due to shipment volume in excess of the stable capacity
- Predict ‘used capacity’ at the time of tender receipt, the shipment volume in the tender’s market on the pickup day of the tender

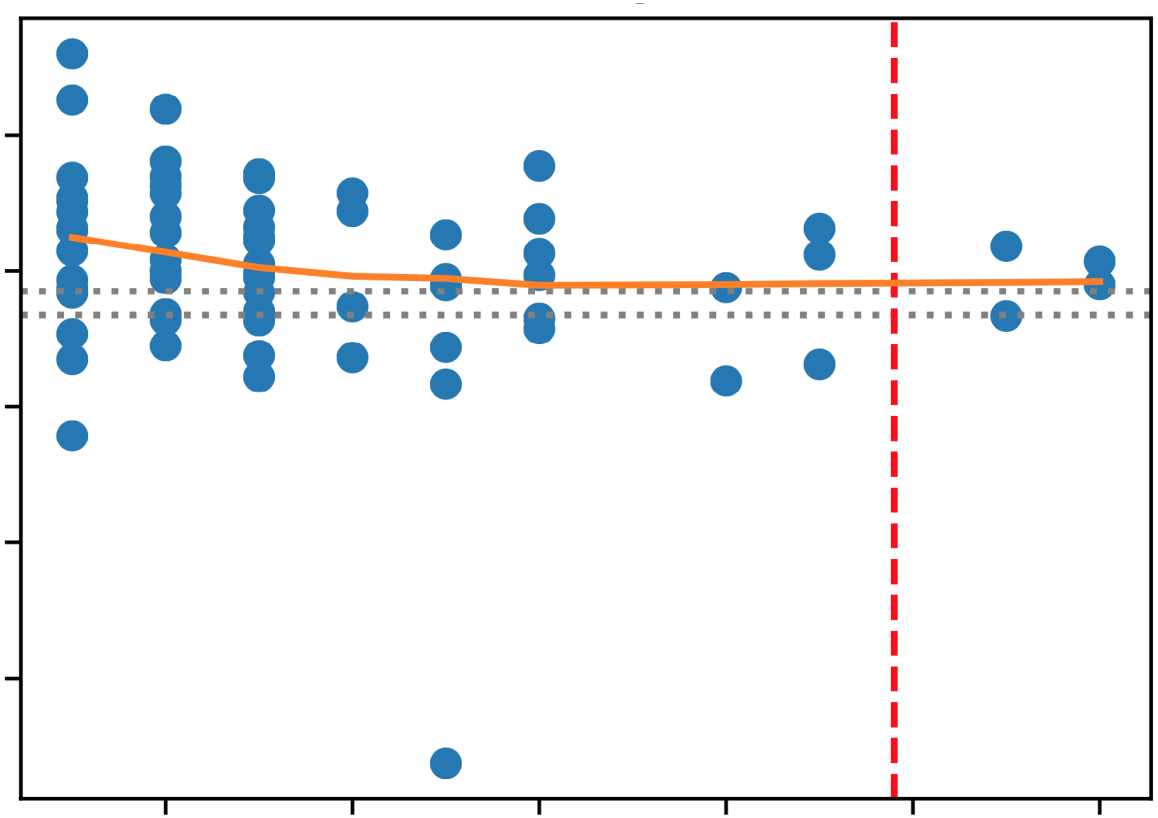
Stable capacity: initial approach



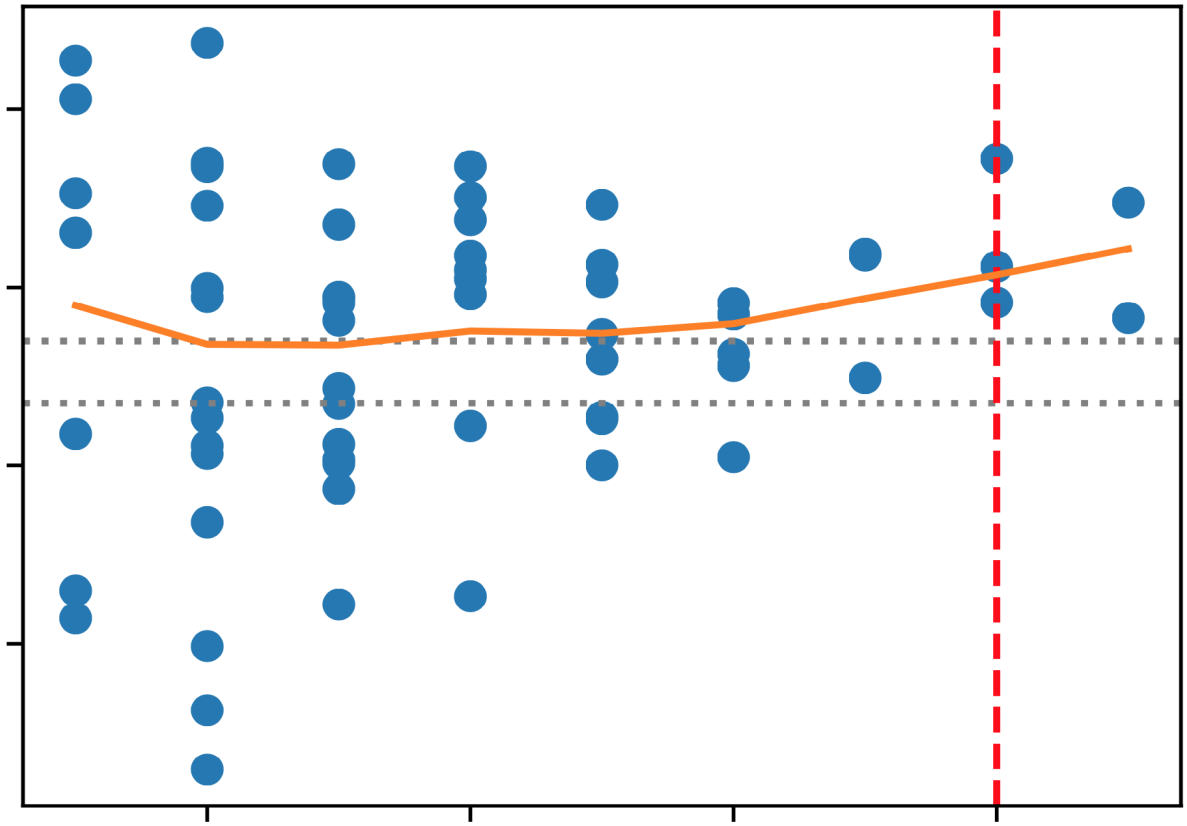
Stable capacity: initial approach



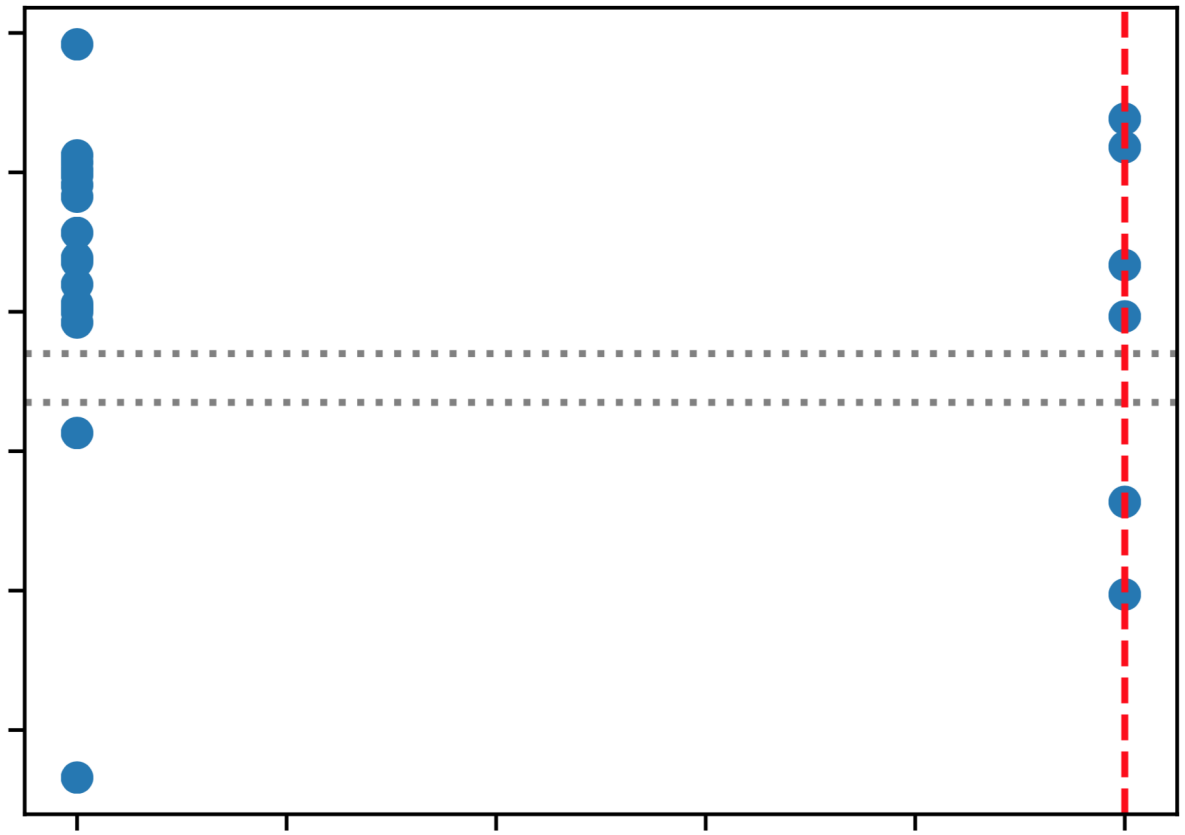
Stable capacity: initial approach



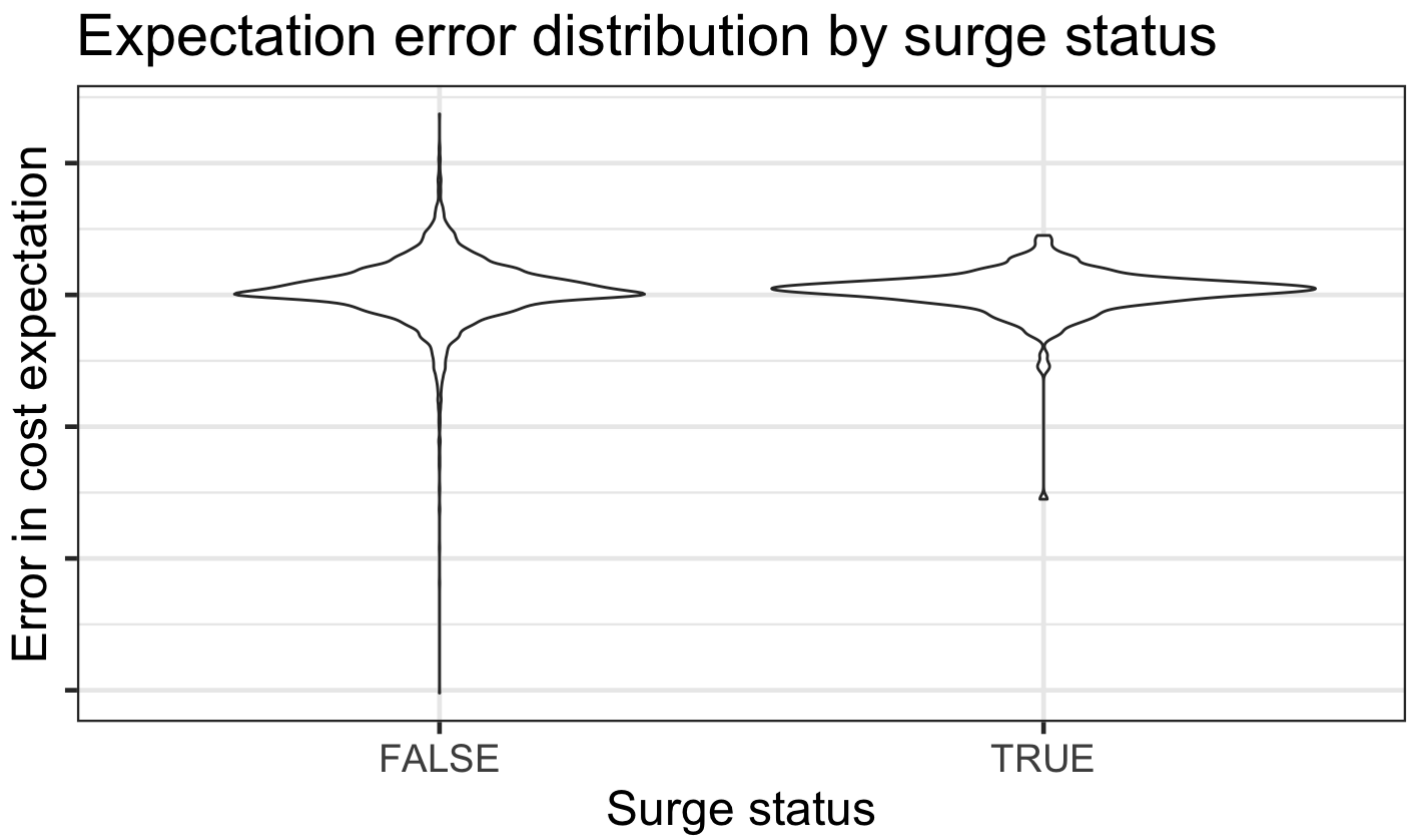
Stable capacity: initial approach



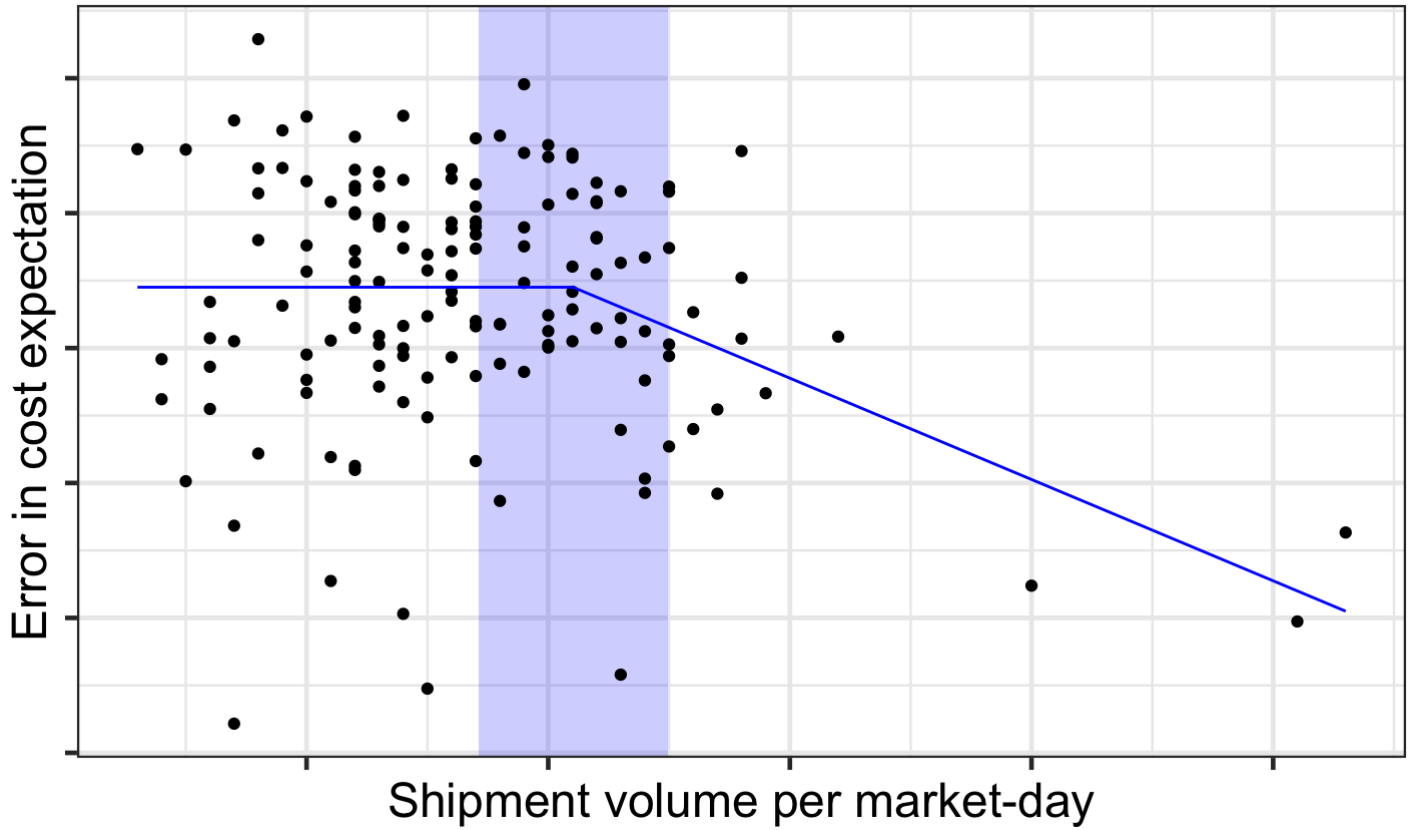
Stable capacity: initial approach



Surge pricing: initial approach



Stable capacity: segmented regression



Segmented regression

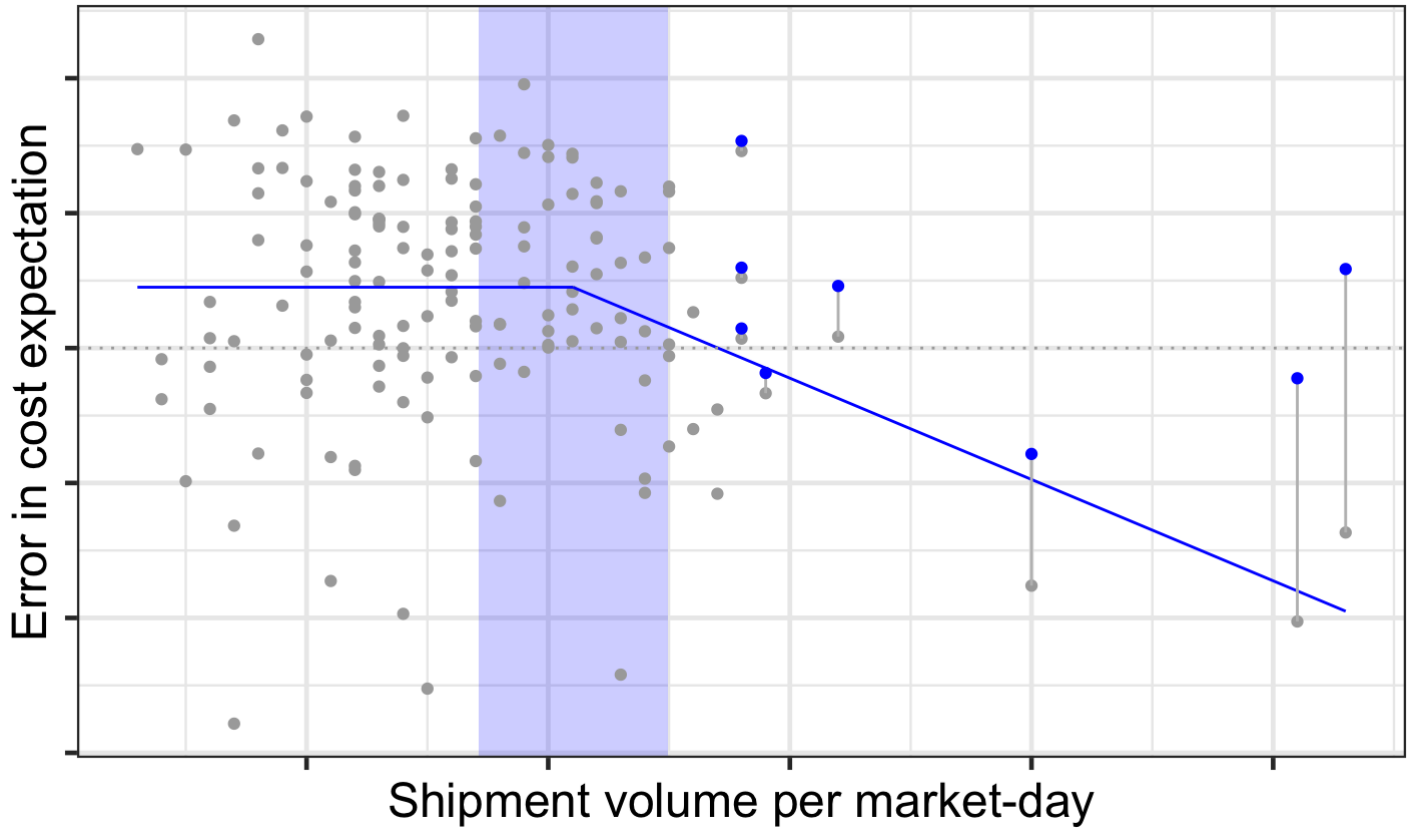
```
# Fit intercept-only OLS, constraining LHS slope to 0
lm_fit = lm(expectation_error ~ 1, data = market_days)

# Fit segmented regression, choosing midpoint as initial breakpoint
seg_fit = tryCatch(
  segmented::segmented(
    lm_fit,
    seg.Z = ~n_shipments,
    psi = with(market_days,
               (max(n_shipments) - min(n_shipments)) / 2)),
  error = function(e) NULL)
```

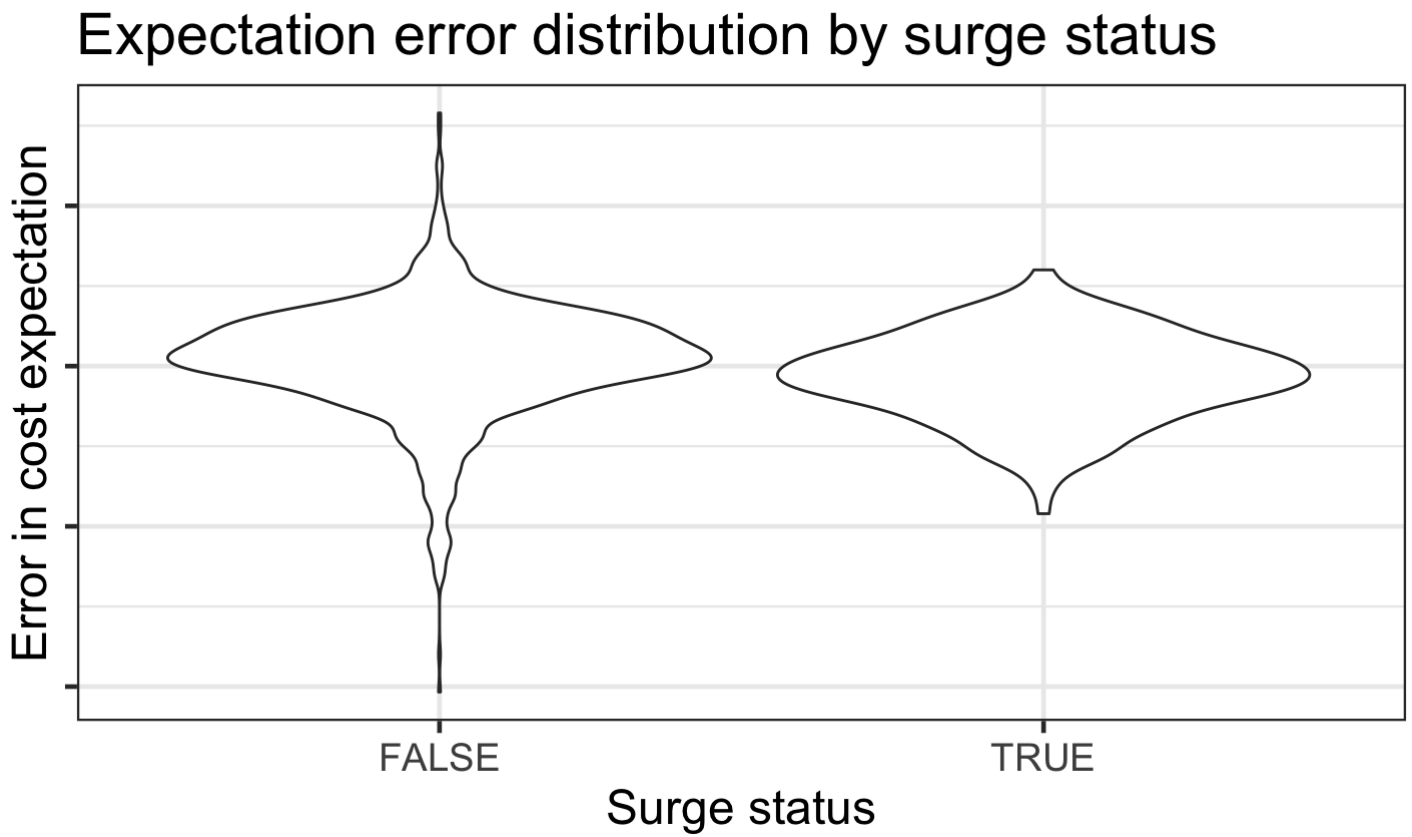
Segmented regression

```
##
##  ***Regression Model with Segmented Relationship(s)***
##
## Call:
## segmented.lm(obj = lm_fit, seg.Z = ~n_shipments)
##
## Estimated Break-Point(s):
##    Est. St.Err
##  0.032  1.393
##
## Meaningful coefficients of the linear terms:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      23.503     16.385   1.434    0.155
## U1.n_shipments   12.218      8.678   1.408       NA
##
## Residual standard error: 23.17 on 97 degrees of freedom
## Multiple R-Squared: 0.02115,  Adjusted R-squared: 0.0009699
##
## Convergence attained in 1 iterations with relative change -0.03979376
```

Surge pricing: segmented regression



Surge pricing: segmented regression



Segmented regression: all markets

n_markets	n_estimated	n_davies	p_negative	coverage
1040	47	12	100%	16.2%

Conclusion

- “This method seems *perfect*.” ~ Principal machine learning engineer
- Ultimately not implemented due to limited number of markets to which method could be applied.
- Summer is coming …