# Quantile Regression

Rather than make a prediction for the mean and then add a measure of variance to produce a prediction interval, quantile regression predicts the intervals directly. In quantile regression, predictions don’t correspond with the arithmetic mean but instead with a specified quantile. To create a 90% prediction interval, you just make predictions at the 5th and 95th percentiles – together the two predictions constitute a prediction interval.

For these reasons, quantile regression is often a highly practical choice for many modeling scenarios that require prediction intervals.

# Example

The {parsnip} package does not yet have a parsnip::linear\_reg() method that supports linear quantile regression. Hence I took this as an opportunity to set-up an example for a random forest model using the {ranger} package as the engine in my workflow.

A more apples-to-apples comparison would have been to abandon the {parsnip} framework and gone through an example using the {quantreg} package for quantile regression… maybe in a future post.

## Quantile Regression Forest

Load packages:

library(tidyverse)

library(tidymodels) library(AmesHousing) library(gt)

# (simpler solution should be implemented in future versions of {gt}) fmt\_if\_number <- function(..., digits = 2) {

input <- c(...)

fmt <- paste0("%.", digits, "f")

if (is.numeric(input)) return(sprintf(fmt, input)) return(input)

}

Load data:

ames <- make\_ames() %>%

mutate(Years\_Old = Year\_Sold - Year\_Built,

Years\_Old = ifelse(Years\_Old < 0, 0, Years\_Old))

set.seed(4595)

data\_split <- initial\_split(ames, strata = "Sale\_Price", p = 0.75)

ames\_train <- training(data\_split) ames\_holdout <- testing(data\_split)

We can remove a few of the transformations that had been important for linear models:

transformations that don’t change the order of observations in a regressor generally don’t make a difference for tree-based methods, so we can remove most of the step\_log()’s tree based models are also good at capturing interactions / dependent relationships on their own, hence we can also remove step\_interact()

#RF models require comparably less pre-processing to linear models rf\_recipe <-

recipe(

Sale\_Price ~ Lot\_Area + Neighborhood + Years\_Old + Gr\_Liv\_Area + Overall\_Qual + Total\_Bsmt\_SF + Garage\_Area,

data = ames\_train

) %>%

step\_log(Sale\_Price, base = 10) %>% step\_other(Neighborhood, Overall\_Qual, threshold = 50) %>% step\_novel(Neighborhood, Overall\_Qual) %>% step\_dummy(Neighborhood, Overall\_Qual)

For our quantile regression example, we are using a random forest model rather than a linear model. Specifying quantreg = TRUE tells {ranger} that we will be estimating quantiles rather than averages8.

rf\_mod <- rand\_forest() %>%

set\_engine("ranger", importance = "impurity", seed = 63233, quantreg

= TRUE) %>%

set\_mode("regression")

set.seed(63233)

rf\_wf <- workflows::workflow() %>% add\_model(rf\_mod) %>% add\_recipe(rf\_recipe) %>% fit(ames\_train)

# Review

Tidymodels does not yet have a predict() method for extracting quantiles.. Hence in the code below I first extract the {ranger} fit object and then use this to make predictions for the quantiles.

preds\_bind <- function(data\_fit, lower = 0.05, upper = 0.95){ predict(

rf\_wf$fit$fit$fit,

workflows::pull\_workflow\_prepped\_recipe(rf\_wf) %>% bake(data\_fit), type = "quantiles",

quantiles = c(lower, upper, 0.50)

) %>%

with(predictions) %>% as\_tibble() %>%

set\_names(paste0(".pred", c("\_lower", "\_upper", ""))) %>% mutate(across(contains(".pred"), ~10^.x)) %>% bind\_cols(data\_fit) %>%

select(contains(".pred"), Sale\_Price, Lot\_Area, Neighborhood, Years\_Old, Gr\_Liv\_Area, Overall\_Qual, Total\_Bsmt\_SF, Garage\_Area)

}

rf\_preds\_test <- preds\_bind(ames\_holdout)

Let’s review a sample of prediction intervals.

set.seed(1234) rf\_preds\_test %>%

mutate(pred\_interval = ggplot2::cut\_number(Sale\_Price, 10)) %>% group\_by(pred\_interval) %>%

sample\_n(2) %>% ggplot(aes(x = .pred))+

geom\_point(aes(y = .pred, color = "prediction interval"))+ geom\_errorbar(aes(ymin = .pred\_lower, ymax = .pred\_upper, color =

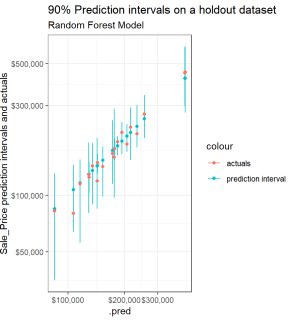
"prediction interval"))+

geom\_point(aes(y = Sale\_Price, color = "actuals"))+ scale\_x\_log10(labels = scales::dollar)+ scale\_y\_log10(labels = scales::dollar)+

labs(title = "90% Prediction intervals on a holdout dataset", subtitle = "Random Forest Model",

y = "Sale\_Price prediction intervals and actuals")+ theme\_bw()+

coord\_fixed()



If we compare these against similar samples when using the analytic and simulation based approaches for linear regression models, we find that the width of the intervals vary substantially more when built using quantile regression forests.

## Performance

Overall model performance on a holdout dataset is similar (maybe slightly better) for an (untuned) Quantile Regression Forest compared to the linear model (MAPE on holdout dataset of 11% vs. 11.8% with linear model).

we can compare performance on train and holdout datasets to provide an indicator of overfitting:

rf\_preds\_train <- preds\_bind(ames\_train)

bind\_rows(

yardstick::mape(rf\_preds\_train, Sale\_Price, .pred), yardstick::mape(rf\_preds\_test, Sale\_Price, .pred)

) %>%

mutate(dataset = c("training", "holdout")) %>% gt::gt() %>%

gt::fmt\_number(".estimate", decimals = 1)

### .metric .estimator .estimate dataset

|  |  |  |  |
| --- | --- | --- | --- |
| mape | standard | 4.7 | training |
| mape | standard | 11.0 | holdout |

We see a substantial discrepancy in performance. This puts the validity of the expected coverage of our prediction intervals in question…

## Coverage

Let’s check our coverage rates on a holdout dataset:

coverage <- function(df, ...){

df %>%

mutate(covered = ifelse(Sale\_Price >= .pred\_lower & Sale\_Price <=

.pred\_upper, 1, 0)) %>% group\_by(...) %>% summarise(n = n(),

n\_covered = sum( covered

),

stderror = sd(covered) / sqrt(n), coverage\_prop = n\_covered / n)

}

rf\_preds\_test %>% coverage() %>%

mutate(across(c(coverage\_prop, stderror), ~.x \* 100)) %>% gt::gt() %>%

gt::fmt\_number("stderror", decimals = 2) %>% gt::fmt\_number("coverage\_prop", decimals = 1)

### n n\_covered stderror coverage\_prop

731 706 0.67 96.6

Surprisingly, we see a coverage probability for our prediction intervals of >96% on our holdout dataset – greater than our expected coverage of 90%. This suggests our prediction intervals are, in aggregate, quite conservative. Typically the coverage on the holdout dataset would be the same or *less* than the expected coverage.

I described how to tune prediction intervals using coverage rates on holdout data. The code below applies this approach, though due to the surprising finding of a *higher empirical coverage* rate (which is opposite of what we typically observe) I will be identifying a more narrow (rather than broader) *expected coverage*, i.e. prediction interval.

tune\_alpha\_coverage <- function(lower, upper){ preds <- preds\_bind(ames\_holdout, lower, upper)

preds %>% coverage() %>%

pull(coverage\_prop)

}

If we review the expected coverage against the empirical coverage rates, we see the coverage of this model seems, across prediction intervals, to be underestimated.

coverages <- tibble(lower = seq(0.025, 0.2, by = 0.005)) %>% mutate(upper = 1 - lower,

expected\_coverage = upper - lower) %>%

mutate(hold\_out\_coverage = map2\_dbl(lower, upper, tune\_alpha\_coverage))

coverages %>% ggplot()+

geom\_line(aes(x = expected\_coverage, y = hold\_out\_coverage))+ geom\_line(aes(x = expected\_coverage, y = expected\_coverage, colour =

"expected = empirical"), alpha = 0.5)+

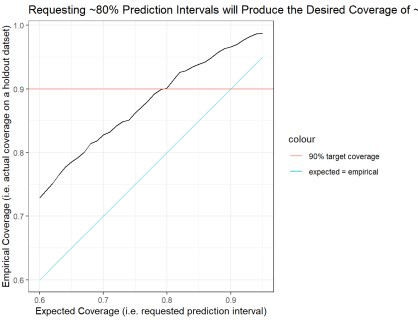
geom\_hline(aes(yintercept = 0.90, colour = "90% target coverage"))+ coord\_fixed()+

theme\_bw()+

labs(title = "Requesting ~80% Prediction Intervals will Produce the Desired Coverage of ~90%",

x = "Expected Coverage (i.e. requested prediction interval)", y = "Empirical Coverage (i.e. actual coverage on a holdout

datset)")



The figure above suggests that an expected prediction interval of 80% will produce an interval with actual coverage of about 90%. For the remainder of the body of this post I will use the 80% expected prediction intervals (90% empirical prediction intervals) from our quantile regression forest model.

### Coverage Across Deciles

separate\_cut <- function(df, group\_var = price\_grouped){ df %>%

mutate(x\_tmp = str\_sub({{ group\_var }}, 2, -2)) %>% separate(x\_tmp, c("min", "max"), sep = ",") %>% mutate(across(c(min, max), as.double))

}

rf\_preds\_test\_80 <- preds\_bind(ames\_holdout, lower = 0.10, upper = 0.90)

coverage\_80 <- rf\_preds\_test\_80 %>%

mutate(price\_grouped = ggplot2::cut\_number(.pred, 5)) %>% coverage(price\_grouped) %>%

separate\_cut() %>% mutate(expected\_coverage = "80%")

coverage\_80 %>%

ggplot(aes(x = forcats::fct\_reorder(scales::dollar(max, scale = 1/1000), max), y = coverage\_prop))+

geom\_line(aes(group = expected\_coverage))+ geom\_errorbar(aes(ymin = coverage\_prop - 2 \* stderror, ymax =

ifelse(coverage\_prop + 2 \* stderror > 1, 1, coverage\_prop + 2 \* stderror)))+

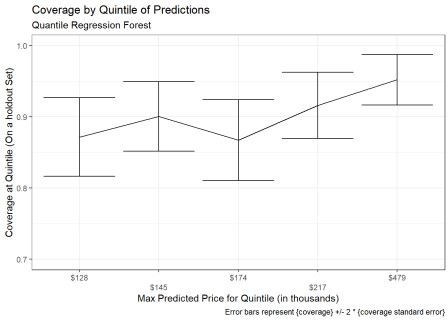
coord\_cartesian(ylim = c(0.70, 1))+ scale\_x\_discrete(guide = guide\_axis(n.dodge = 2))+ # facet\_wrap(~expected\_coverage)+

labs(x = "Max Predicted Price for Quintile (in thousands)", y = "Coverage at Quintile (On a holdout Set)",

title = "Coverage by Quintile of Predictions", subtitle = "Quantile Regression Forest",

caption = "Error bars represent {coverage} +/- 2 \* {coverage standard error}")+

theme\_bw()



There *appears* to be slightly lower empirical coverage rates for smaller predicted prices, however a statistical test suggests any difference is not significant:

*Chi-squared test of association between {covered} ~ {predicted price group}:*

rf\_preds\_test\_80 %>%

mutate(price\_grouped = ggplot2::cut\_number(.pred, 5)) %>% mutate(covered = ifelse(Sale\_Price >= .pred\_lower & Sale\_Price <=

.pred\_upper, 1, 0)) %>% with(chisq.test(price\_grouped, covered)) %>% pander::pander()

Pearson’s Chi-squared test:

price\_grouped and covered

### Test statistic df P value

7.936 4 0.09394

## Interval Width

### In aggregate:

get\_interval\_width <- function(df, ){

df %>%

mutate(interval\_width = .pred\_upper - .pred\_lower, interval\_pred\_ratio = interval\_width / .pred) %>%

group\_by(...) %>% summarise(n = n(),

mean\_interval\_width\_percentage = mean(interval\_pred\_ratio)

stdev = sd(interval\_pred\_ratio),

stderror = sd(interval\_pred\_ratio) / sqrt(n))

}

rf\_preds\_test\_80 %>% get\_interval\_width() %>%

mutate(across(c(mean\_interval\_width\_percentage, stdev, stderror),

~.x\*100)) %>%

gt::gt() %>%

gt::fmt\_number(c("stdev", "stderror"), decimals = 2) %>% gt::fmt\_number("mean\_interval\_width\_percentage", decimals = 1)

### n mean\_interval\_width\_percentage stdev stderror

731 44.1 19.19 0.71

### By quintiles of predictions:

interval\_width\_80 <- rf\_preds\_test\_80 %>% mutate(price\_grouped = ggplot2::cut\_number(.pred, 5)) %>% get\_interval\_width(price\_grouped) %>%

separate\_cut() %>%

select(-price\_grouped) %>% mutate(expected\_coverage = "80%")

interval\_width\_80 %>% mutate(across(c(mean\_interval\_width\_percentage, stdev, stderror),

~.x\*100)) %>%

gt::gt() %>%

gt::fmt\_number(c("stdev", "stderror"), decimals = 2) %>% gt::fmt\_number("mean\_interval\_width\_percentage", decimals = 1)

### n mean\_interval\_width\_percentage stdev stderror min max expected\_coverage

|  |  |  |
| --- | --- | --- |
| 148 53.3 | 19.86 1.63 | 63900 128000 80% |
| 151 40.8 | 18.30 1.49 | 128000 145000 80% |

**n mean\_interval\_width\_percentage stdev stderror min max expected\_coverage**

|  |  |  |  |
| --- | --- | --- | --- |
| 143 42.4 | 17.31 | 1.45 | 145000 174000 80% |
| 143 37.1 | 17.35 | 1.45 | 174000 217000 80% |
| 146 46.7 | 19.04 | 1.58 | 217000 479000 80% |

This suggests that quantile regression forests are better able to differentiate measures of uncertainty by observations compared to the linear models from the previous posts.

# Closing Notes

This post walked through an example using quantile regression forests within {tidymodels} to build prediction intervals. Such an approach is relatively simple and computationally efficient to implement and flexible in its ability to vary interval width according to the uncertainty associated with an observation. The section on Coverage suggests that additional review may be required of prediction intervals and that alpha levels may need to be tuned according to coverage rates on holdout data.

*Advantages of Quantile Regression for Building Prediction Intervals:*

Quantile regression methods are generally more robust to model assumptions (e.g. heteroskedasticity of errors).

For random forests and other tree-based methods, estimation techniques allow a single model to produce predictions at all quantiles.

While higher in computation costs than analytic methods, costs are still low compared to

simulation based approaches.

*Downsides:*

Are not immune to overfitting and related issues (though these also plague parametric methods and can sometimes be improved by tuning).

Models are more commonly designed to predict the mean rather than a quantile, so there may be fewer model classes or packages available that are ready to use out-of-the-box. These may require editing the objective function so that the model is optimized on the quantile loss, for example.

# Appendix

## Residual Plots

Remember that these are on a log(10) scale. *Residual plot on training data:* rf\_preds\_train %>%

mutate(covered = ifelse(

Sale\_Price >= .pred\_lower & Sale\_Price <= .pred\_upper, "covered",

"not covered")

) %>%

mutate(across(c("Sale\_Price", contains(".pred")), ~log(.x, 10))) %>% mutate(.resid = Sale\_Price - .pred) %>%

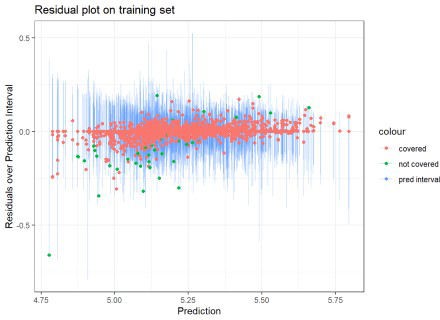
mutate(pred\_original = .pred) %>% mutate(across(contains(".pred"), ~(.x - .pred))) %>% ggplot(aes(x = pred\_original,))+

geom\_errorbar(aes(ymin = .pred\_lower, ymax = .pred\_upper, colour = "pred interval"), alpha = 0.3)+

geom\_point(aes(y = .resid, colour = covered))+ theme\_bw()+

labs(x = "Prediction",

y = "Residuals over Prediction Interval", title = "Residual plot on training set")



The several points with residuals of zero on the training data made me think that {ranger} might be set-up such that if there are many residuals of 0, it may assume there is overfitting going on and then default to some kind of conservative or alternative approach to computing the prediction intervals. However this proved false as when I tried higher values of min\_n – which would reduce any overfitting – I still had similar results regarding a higher than expected coverage rate.

I also wondered whether some of the extreme points (e.g. the one with a residual of -0.5) may be contributing to the highly conservative intervals. When I removed outliers I found a *slight* narrowing of the prediction intervals, but not much… so that also didn’t seem to explain things.

*Residual plot on holdout data:*

rf\_preds\_test %>% mutate(covered = ifelse(

Sale\_Price >= .pred\_lower & Sale\_Price <= .pred\_upper, "covered",

"not covered")

) %>%

mutate(across(c("Sale\_Price", contains(".pred")), ~log(.x, 10))) %>% mutate(.resid = Sale\_Price - .pred) %>%

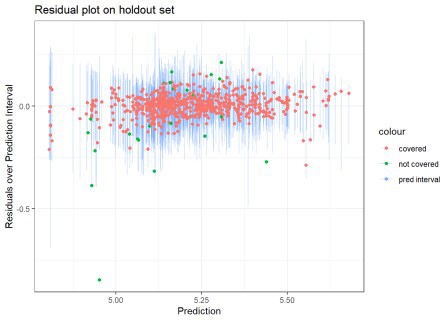
mutate(pred\_original = .pred) %>% mutate(across(contains(".pred"), ~(.x - .pred))) %>% ggplot(aes(x = pred\_original,))+

geom\_errorbar(aes(ymin = .pred\_lower, ymax = .pred\_upper, colour = "pred interval"), alpha = 0.3)+

geom\_point(aes(y = .resid, colour = covered))+ theme\_bw()+

labs(x = "Prediction",

y = "Residuals over Prediction Interval", title = "Residual plot on holdout set")



## Other Charts

### Sample of observations, but now using 80% Prediction Intervals:

set.seed(1234) rf\_preds\_test\_80 %>%

mutate(pred\_interval = ggplot2::cut\_number(Sale\_Price, 10)) %>% group\_by(pred\_interval) %>%

sample\_n(2) %>% ggplot(aes(x = .pred))+

geom\_point(aes(y = .pred, color = "prediction interval"))+ geom\_errorbar(aes(ymin = .pred\_lower, ymax = .pred\_upper, color =

"prediction interval"))+

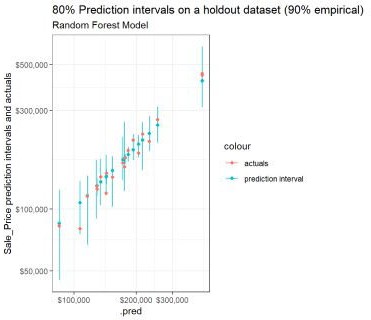
geom\_point(aes(y = Sale\_Price, color = "actuals"))+ scale\_x\_log10(labels = scales::dollar)+ scale\_y\_log10(labels = scales::dollar)+

labs(title = "80% Prediction intervals on a holdout dataset (90% empirical)",

subtitle = "Random Forest Model",

y = "Sale\_Price prediction intervals and actuals")+ theme\_bw()+

coord\_fixed()



### Coverage rates across quintiles for expected coverage of 80% and 90%:

coverage\_90 <- rf\_preds\_test %>%

mutate(price\_grouped = ggplot2::cut\_number(.pred, 5)) %>% coverage(price\_grouped) %>%

separate\_cut() %>% mutate(expected\_coverage = "90%")

bind\_rows(coverage\_80, coverage\_90) %>%

ggplot(aes(x = forcats::fct\_reorder(scales::dollar(max, scale = 1/1000), max), y = coverage\_prop, colour = expected\_coverage))+

geom\_line(aes(group = expected\_coverage))+ geom\_errorbar(aes(ymin = coverage\_prop - 2 \* stderror, ymax =

ifelse(coverage\_prop + 2 \* stderror > 1, 1, coverage\_prop + 2 \* stderror)))+

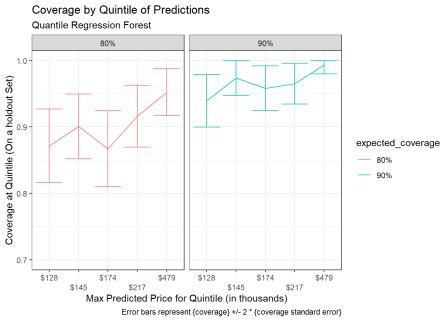
coord\_cartesian(ylim = c(0.70, 1))+ scale\_x\_discrete(guide = guide\_axis(n.dodge = 2))+ facet\_wrap(~expected\_coverage)+

labs(x = "Max Predicted Price for Quintile (in thousands)", y = "Coverage at Quintile (On a holdout Set)",

title = "Coverage by Quintile of Predictions", subtitle = "Quantile Regression Forest",

caption = "Error bars represent {coverage} +/- 2 \* {coverage standard error}")+

theme\_bw()



### Interval Widths across quantiles for expected coverage of 80% and 90%:

interval\_width\_90 <- rf\_preds\_test %>%

mutate(price\_grouped = ggplot2::cut\_number(.pred, 5)) %>% get\_interval\_width(price\_grouped) %>%

separate\_cut() %>%

select(-price\_grouped) %>% mutate(expected\_coverage = "90%")

bind\_rows(interval\_width\_80, interval\_width\_90) %>%

ggplot(aes(x = forcats::fct\_reorder(scales::dollar(max, scale = 1/1000), max), y = mean\_interval\_width\_percentage, colour = expected\_coverage))+

geom\_line(aes(group = expected\_coverage))+ geom\_errorbar(aes(ymin = mean\_interval\_width\_percentage - 2 \*

stderror, ymax = mean\_interval\_width\_percentage + 2 \* stderror))+ # coord\_cartesian(ylim = c(0.70, 1.01))+

scale\_x\_discrete(guide = guide\_axis(n.dodge = 2))+ facet\_wrap(~expected\_coverage)+

labs(x = "Max Predicted Price for Quintile (in thousands)",

y = "Average Interval Width as a Percentage of Prediction", title = "Interval Width by Quintile of Predictions (On a holdout

Set)",

subtitle = "Quantile Regression Forest",

caption = "Error bars represent {interval width} +/- 2 \* {interval width standard error}")+

theme\_bw()

