Different Types of Models

DATA 202 21FA

```r
data(ames, package = "modeldata")
ames_all <- ames %>%
  filter(Gr_Liv_Area < 4000, Sale_Condition == "Normal") %>%
  mutate(across(where(is.integer), as.double)) %>%
  mutate(Sale_Price = Sale_Price / 1000)
rm(ames)
```

```r
metrics <- yardstick::metric_set(mae, mape, rsq_trad)

set.seed(10) # Seed the random number generator
ames_split <- initial_split(ames_all, prop = 2 / 3)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
```

```r
model2 <-
  decision_tree(mode = "regression", tree_depth = 30) %>%
  fit(Sale_Price ~ Latitude + Longitude, data = ames_train)
```

Objectives

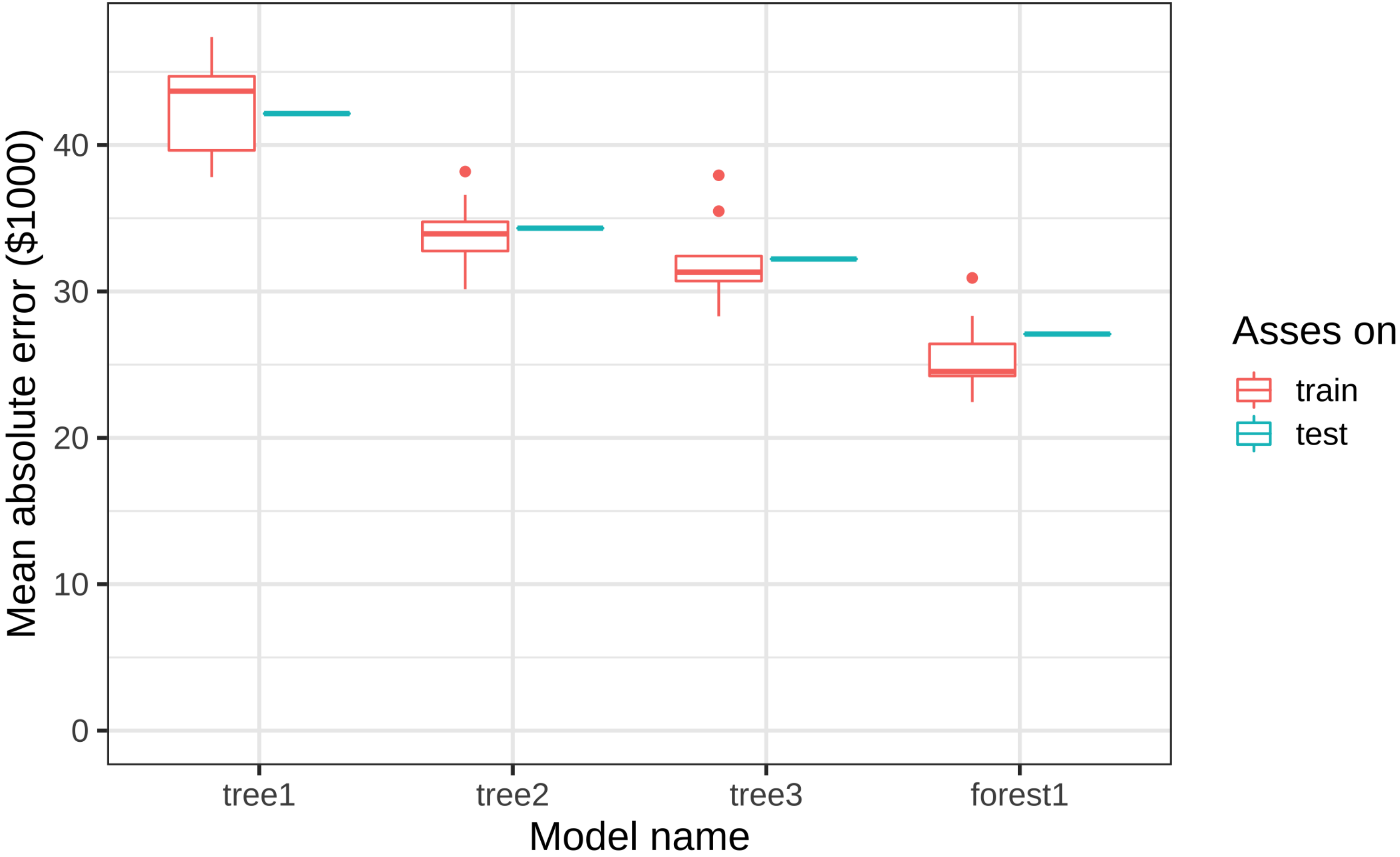
What type of model for what type of data?

- Describe how a Random Forest makes predictions
- Describe how a linear model makes predictions
- Compare and contrast linear and tree models

Reference: Fitting and Predicting with `parsnip`

Random Forests

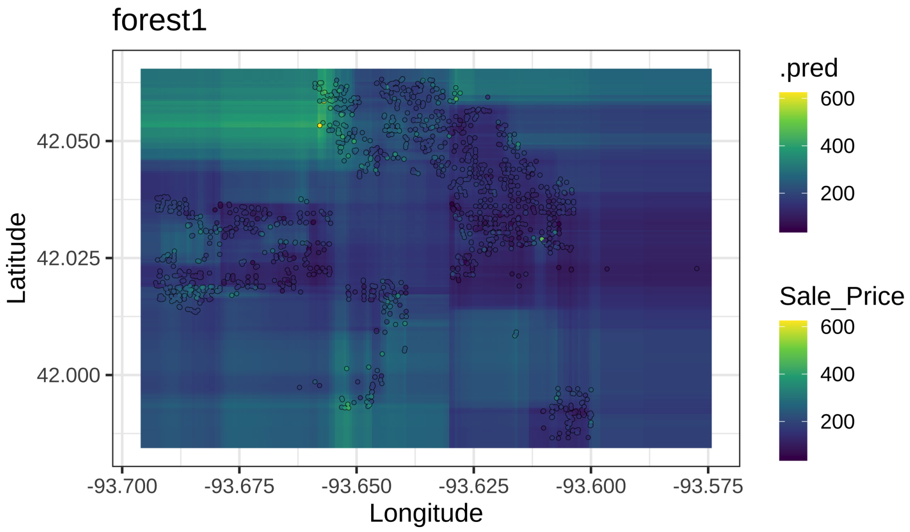
Why?



Mean absolute error ($1000)
0
10
20
30
40
tree1
tree2
tree3
forest1
Model name
Asses on
train
test

Using random forests

```
forest1 <- rand_forest(mode = "regression") %>%
  fit(Sale_Price ~ Latitude + Longitude, ames_train)
show_latlong_model(ames_train, forest1)
```



A random forest has many trees.

```
forest_internals <- extract_fit_engine(forest1)
forest_internals
```

```
Ranger result

Call:
 ranger::ranger(x = maybe_data_frame(x), y = y, num.threads = 1,

Type:                             Regression
Number of trees:                  500
Sample size:                      1608
Number of independent variables:  2
Mtry:                             1
Target node size:                 5
Variable importance mode:         none
Splitrule:                        variance
OOB prediction error (MSE):       1444.623
R squared (OOB):                  0.6894006
```

Each tree was trained on a different subset of data

```
ranger::treeInfo(forest_internals, 1) %>% select(-splitvarID) %>%
```

nodeID	leftChild	rightChild	splitvarName	splitval	terminal	prediction
0	1	2	Latitude	42.04601	FALSE	NA
1	3	4	Latitude	42.01889	FALSE	NA
2	5	6	Latitude	42.05305	FALSE	NA
3	7	8	Longitude	-93.63958	FALSE	NA
4	9	10	Longitude	-93.67910	FALSE	NA
5	11	12	Longitude	-93.64695	FALSE	NA
6	13	14	Longitude	-93.65092	FALSE	NA
7	15	16	Longitude	-93.65184	FALSE	NA
8	17	18	Latitude	41.99299	FALSE	NA
9	19	20	Latitude	42.02273	FALSE	NA

RF averages the predictions of each tree

```
forest_internals %>%
  predict(
    data = ames_test %>% head(8),
    predict.all = TRUE, num.trees = 5) %>%
  pluck("predictions")
```

```
        [,1]     [,2]    [,3]     [,4]     [,5]
[1,] 185.00 192.7500 185.750 191.0000 187.1000
[2,] 186.25 178.8333 225.000 176.7500 615.0000
[3,] 171.20 185.0000 173.500 166.5000 170.0000
[4,] 246.60 214.8333 209.750 215.7000 237.5000
[5,] 173.00 160.8000 320.759 278.0000 229.0883
[6,] 173.00 182.8000 320.759 278.0000 229.0883
[7,] 203.00 200.2667 164.380 160.9500 188.0000
[8,] 227.00 227.3250 231.875 226.6667 233.5800
```

Value of Diversity

> I looked and there before me
> was a great multitude that no one could count,
> from every nation, tribe, people and language,
> standing before the throne.

Revelation 7:9, as quoted in Calvin's "From Every Nation"

- Random Forests work because they combine diverse perspectives (from different training data, different choices)
- Reflects value of diversity in God's Kingdom (see also Rev 5:9, 1 Cor 12, etc.)

Linear Models

Fitting a linear model

```r
linear_model <- linear_reg() %>%
  fit(Sale_Price ~ Gr_Liv_Area, data = ames_train)
linear_model
```

```
parsnip model object

Fit time:  3ms

Call:
stats::lm(formula = Sale_Price ~ Gr_Liv_Area, data = data)

Coefficients:
(Intercept)  Gr_Liv_Area
    22.9161       0.1029
```

Aside: you may have seen this in stats class.

```
stats::lm(
  formula = Sale_Price ~ Gr_Liv_Area,
  data = ames_train)
```

```

Call:
stats::lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_train)

Coefficients:
(Intercept)  Gr_Liv_Area
    22.9161       0.1029
```

We'll use one example home from the test set.

```r
example_home <- ames_test %>% slice(1)
example_home %>% select(Gr_Liv_Area, Sale_Price)
```

```
# A tibble: 1 × 2
  Gr_Liv_Area Sale_Price
        <dbl>      <dbl>
1        1804        189
```

What computations can a linear model do?

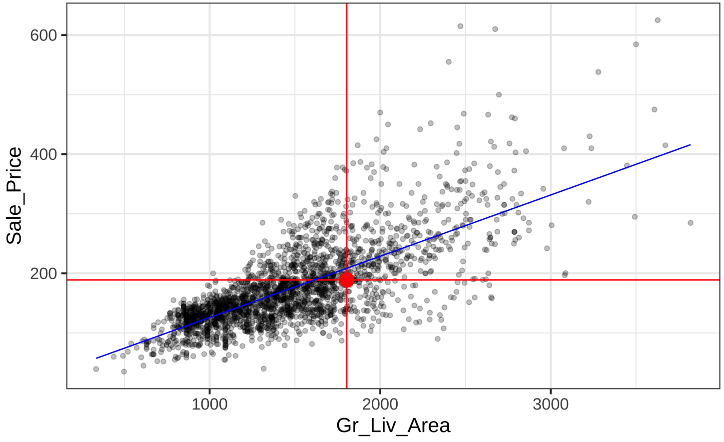
- *Add* up terms.
- Each term: *multiply* a number by a constant.

```r
intercept <- 22.9161
coef_living_area <- 0.1029
```

```r
intercept + coef_living_area *
```

```
[1] 208.5477
```

```r
ggplot(ames_all, aes(x = Gr_Liv_Area, y = S
  geom_point(alpha = .25) +
  geom_hline(yintercept = example_home$Sale
  geom_vline(xintercept = example_home$Gr_L
  geom_point(data = example_home, color = '
  geom_function(fun = function(x) intercept
```

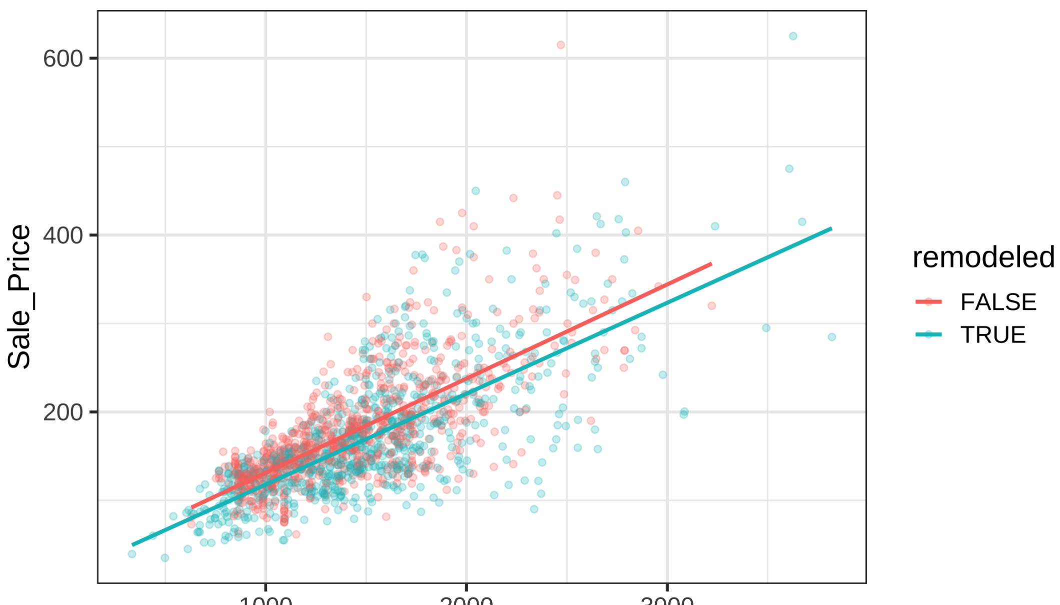


Do remodeled homes sell for more?

`Year Remod/Add`: Remodel date *(same as construction date if no remodeling or additions)* (from dataset documentation)

```
ames_2 <- ames_train %>% mutate(remodeled = Year_Remod_Add != Ye
```

```
ggplot(ames_2, aes(x = Gr_Liv_Area, y = Sale_Price, color = remodeled)) +
  geom_point(alpha = .25) +
  geom_smooth(method = "lm", se = FALSE)
```



Aside: the *sum-as-count* pattern

```
ames_2 %>%
  group_by(remodeled) %>%
  summarize(n = n()) %>%
  mutate(proportion = n / sum(
```

```
# A tibble: 2 × 3
  remodeled     n proportion
  <lgl>     <int>      <dbl>
1 FALSE       866      0.539
2 TRUE        742      0.461
```

```
ames_2 %>% summarize(
  num_remodeled = sum(remodele
  prop_remodeled = mean(remode
)
```

```
# A tibble: 1 × 2
  num_remodeled prop_remodeled
          <int>          <dbl>
1           742          0.461
```

Why does this work?

```
as.numeric(remodeled[1:10])
```

```
 [1] 0 1 0 1 0 0 0 1 1 0
```

Its *sum* is the number of 1's (rows where the condition is true). Its *mean* is the sum

Conditional Logic: Simple Conditions

How could a *linear model* treat remodeled homes differently from non-remodeled?

```
if remodeled:
  Sale_Price = intercept_remodeled + coef_sqft * Gr_Liv_Area
else:
  Sale_Price = intercept_other + coef_sqft * Gr_Liv_Area
```

Conditional Logic: Simple Conditions

How could a *linear model* treat remodeled homes differently from non-remodeled?

```
if remodeled:
  Sale_Price = intercept_remodeled + coef_sqft * Gr_Liv_Area
else:
  Sale_Price = intercept_other + coef_sqft * Gr_Liv_Area
```

Solution: "dummy encoding"

```
Sale_Price =
   intercept_other
   + coef_remodeled * (1 if remodeled)
   + coef_sqft      * Gr_Liv_Area
```

```r
ames_train_2 <- ames_train %>%
  mutate(remodeled = as_factor(Year_Built != Year_Remod_Add))
```

```r
ames_recipe_3 <-
  recipe(Sale_Price ~ Gr_Liv_Area + remodeled, data = ames_train_2) %>%
  step_dummy(remodeled) %>%
  #step_range(all_numeric(), -all_outcomes(), min = 0, max = 1) %>%
  prep()
baked_ames_train <-
  ames_recipe_3 %>% bake(new_data = ames_train_2)
baked_ames_train %>% head(5) %>% knitr::kable(format = "html")
```

Gr_Liv_Area	Sale_Price	remodeled_TRUE.
912	123.0	0
1120	148.8	1
1589	226.5	0
666	64.5	1
2643	380.0	0

Why are is there no column for `remodeled_FALSE`?

Now, fit as normal.

```
ames_model_2 <- linear_reg() %>% set_engine("lm") %>%
  fit(Sale_Price ~ ., data = baked_ames_train)
ames_model_2 %>% tidy() %>% select(term, estimate) %>% kable()
```

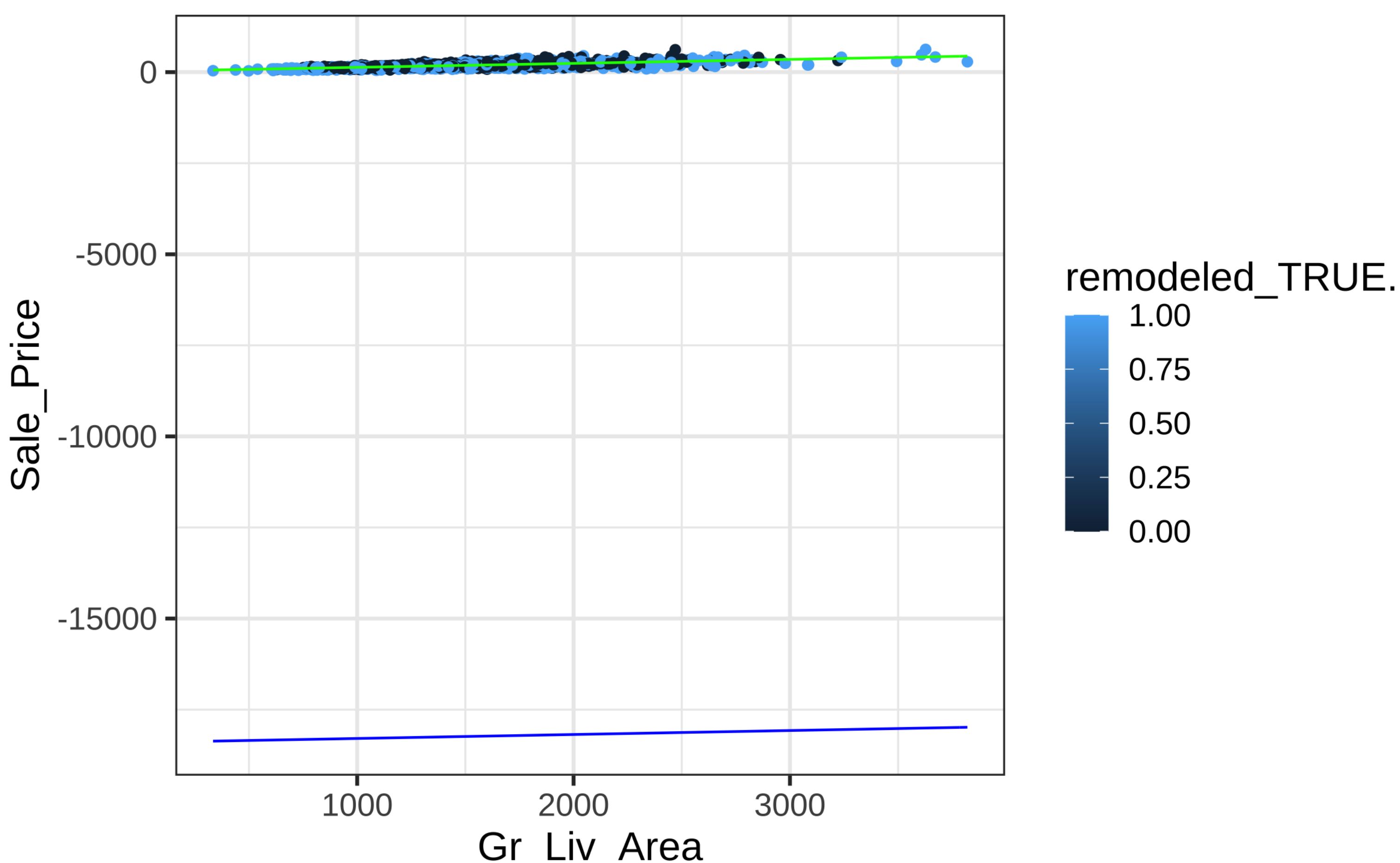
term	estimate
(Intercept)	27.7730315
Gr_Liv_Area	0.1043859
remodeled_TRUE.	-15.1087279

```
Sale_Price =
27.77303
    + 0.1043859 * Gr_Liv_Area
    + -15.10873  * (1 if remodeled)
```

or, in "code":

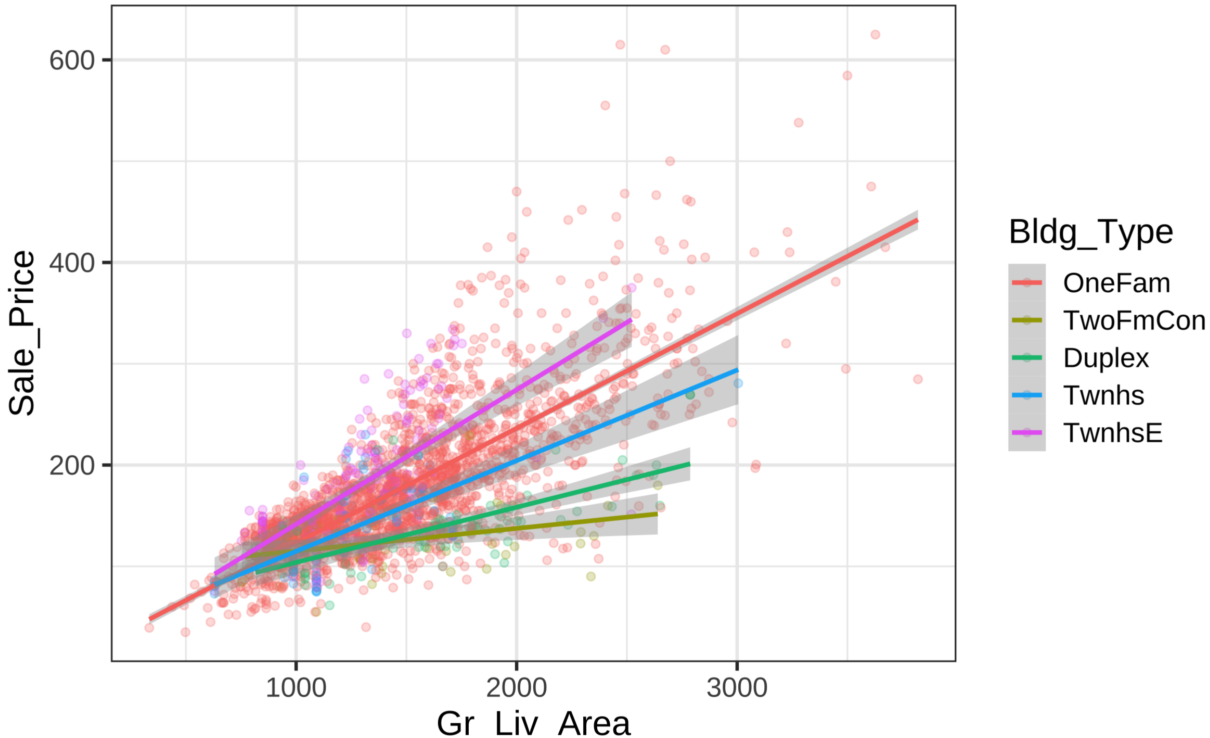
```
if remodeled:
  Sale_Price = 27.8 + 0.1 * Gr_Liv_Area - -15.1 * (1)
  Sale_Price = (27.8 - -15.1) + 0.1 * Gr_Liv_Area
else:
```

```
ggplot(baked_ames_train, aes(x = Gr_Liv_Area, y = Sale_Price, co
  geom_point() +
  geom_function(fun = function(x) (22.6434248 - 18424.0789) + .1
  geom_function(fun = function(x) 22.6434248 + .1091132 * x, col
```



More than two options

```
Bldg Type (Nominal): Type of dwelling
       1Fam    Single-family Detached
       2FmCon    Two-family Conversion; originally built as one-family
       Duplx    Duplex
       TwnhsE    Townhouse End Unit
       TwnhsI    Townhouse Inside Unit
```



```
ames_train %>% count(Bldg_Type) %>% kable()
```

Bldg_Type	n
OneFam	1324
TwoFmCon	31
Duplex	52
Twnhs	68
TwnhsE	133

```
ames_recipe_4 <-
  recipe(Sale_Price ~ Gr_Liv_Area + Bldg_Type, data = ames_train) %>%
  step_dummy(Bldg_Type) %>%
  prep()
baked_ames_train <-
  ames_recipe_4 %>% bake(new_data = ames_train_2)
baked_ames_train %>% head(5) %>% knitr::kable(format = "html")
```

Gr_Liv_Area	Sale_Price	Bldg_Type_TwoFmCon	Bldg_Type_Duplex	Bldg_Type_Twn
912	123.0	0	0	
1120	148.8	0	0	
1589	226.5	0	0	