Models with Conditional Logic

K Arnold

Good Questions

> Final project?

- No final exam, just final project.
- Should demonstrate modeling and validation
- Can optionally be an extension of your midterm project
- Can optionally be groups
- Proposals and matchmaking Moodle forum next week!

> Was there a homework or lab this week?

No, to allow time to work on midterm project & exam. But yes next week.

> Can we review data wrangling stuff like joins and factors?

Review session during my office hours today (3-4pm). NH 295.

Objectives

- Apply dummy encoding to add simple conditional logic to linear regression models
    - Explain how many columns get added in dummy encoding, and why
- Compare and contrast how linear regression and decision tree regression make predictions

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

```r
set.seed(10) # Seed the random number generator
ames_split <- initial_split(ames, prop = 2/3) # Split our data randomly
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
```

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 x 2
##   Gr_Liv_Area Sale_Price
##         <dbl>      <dbl>
## 1        1656     215000
```

What computations can a linear model do?

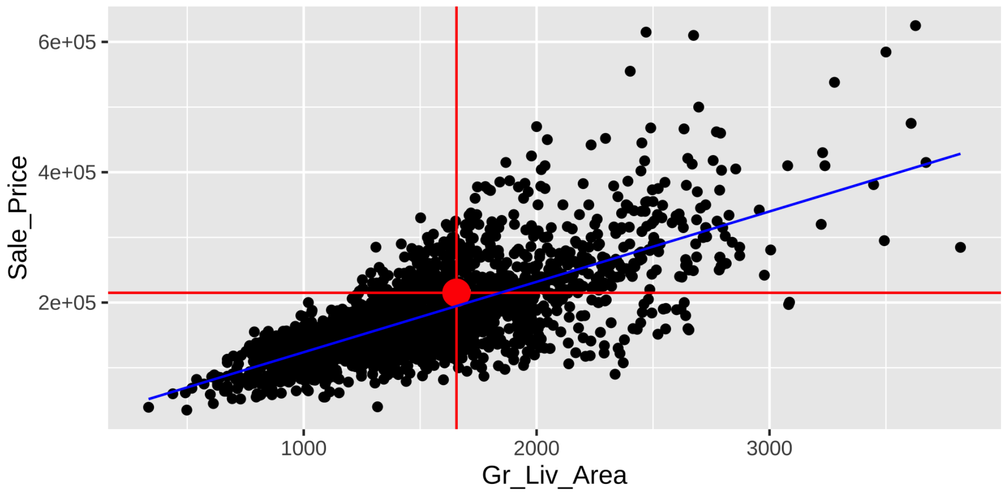
- *Add* up terms.
- Each term: *multiply* a number by a constant.

```r
intercept <- 15793
coef_living_area <- 108
```

```r
intercept + coef_living_area * 1610
```

```
## [1] 189673
```

```r
ggplot(ames, aes(x = Gr_Liv_Area, y = Sale_Price)) +
  geom_point() +
  geom_hline(yintercept = example_home$Sale_Price, color = "red") +
  geom_vline(xintercept = example_home$Gr_Liv_Area, color = "red") +
  geom_point(data = example_home, color = 'red', size = 5) +
  geom_function(fun = function(x) intercept + coef_living_area * x, color = "blue")
```

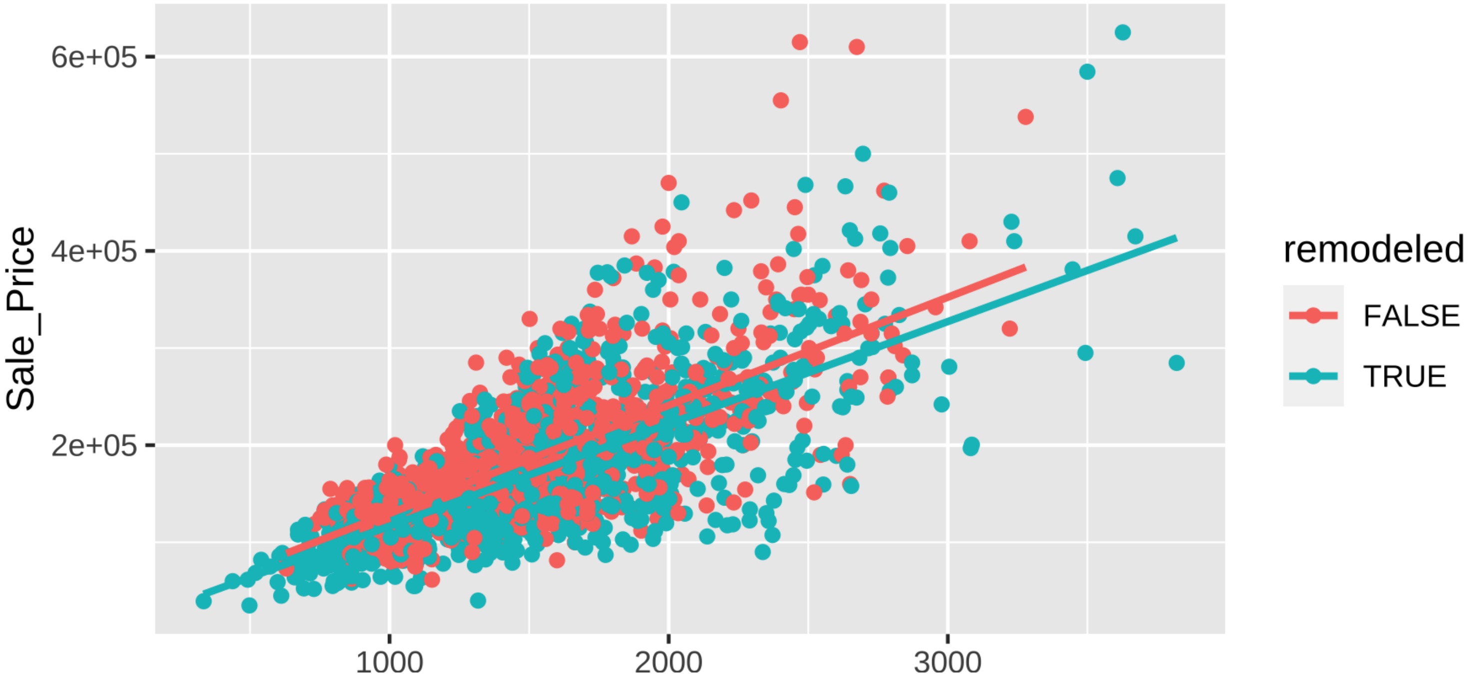


Do remodeled homes sell for more?

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

```
ames_2 <- ames %>% mutate(remodeled = Year_Remod_Add != Year_Built)
```

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



Aside: the *sum-as-count* pattern

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

```
## # A tibble: 2 x 3
##   remodeled      n proportion
##   <lgl>      <int>      <dbl>
## 1 FALSE       1303      0.540
## 2 TRUE        1109      0.460
```

```
ames_2 %>% summarize(
  num_remodeled = sum(remodeled),
  prop_remodeled = mean(remodeled)
)
```

```
## # A tibble: 1 x 2
##   num_remodeled prop_remodeled
##           <int>          <dbl>
## 1          1109          0.460
```

Why does this work?

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

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

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

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_Arera
else:
  Sale_Price = intercept_other + coef_sqft * Gr_Liv_Arera
```

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_Arera
else:
  Sale_Price = intercept_other + coef_sqft * Gr_Liv_Arera
```

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.
896	105000	0
1329	172000	0
1629	189900	1
1604	195500	0
1804	189000	0

Why are is there no column for `remodeled_FALSE`?

Models with dummy variables can be 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()
```

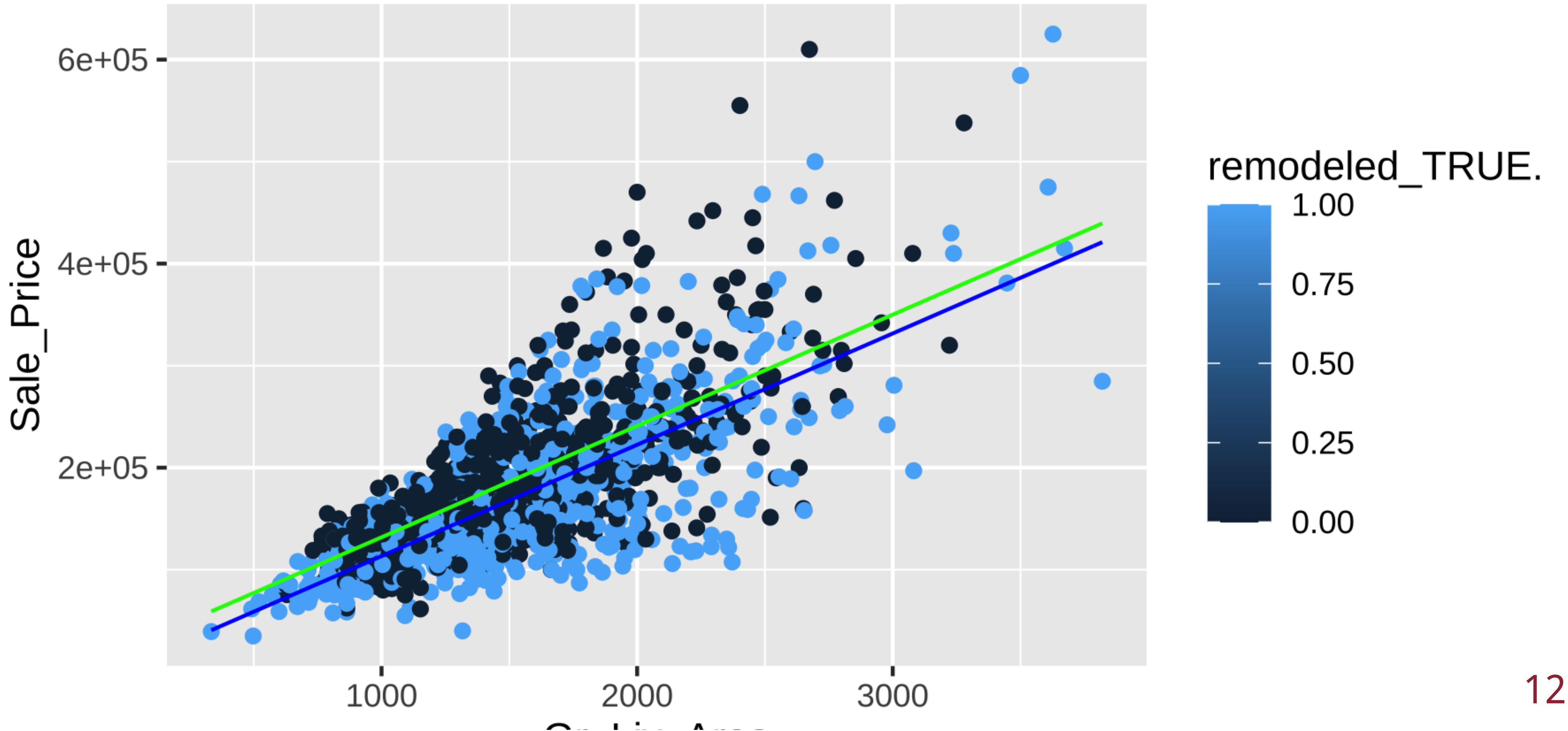
term	estimate
(Intercept)	22643.4248
Gr_Liv_Area	109.1132
remodeled_TRUE.	-18424.0789

```
Sale_Price =
22643.42
   + 109.1132 * Gr_Liv_Area
   + -18424.08  * (1 if remodeled)
```

or, in "code":

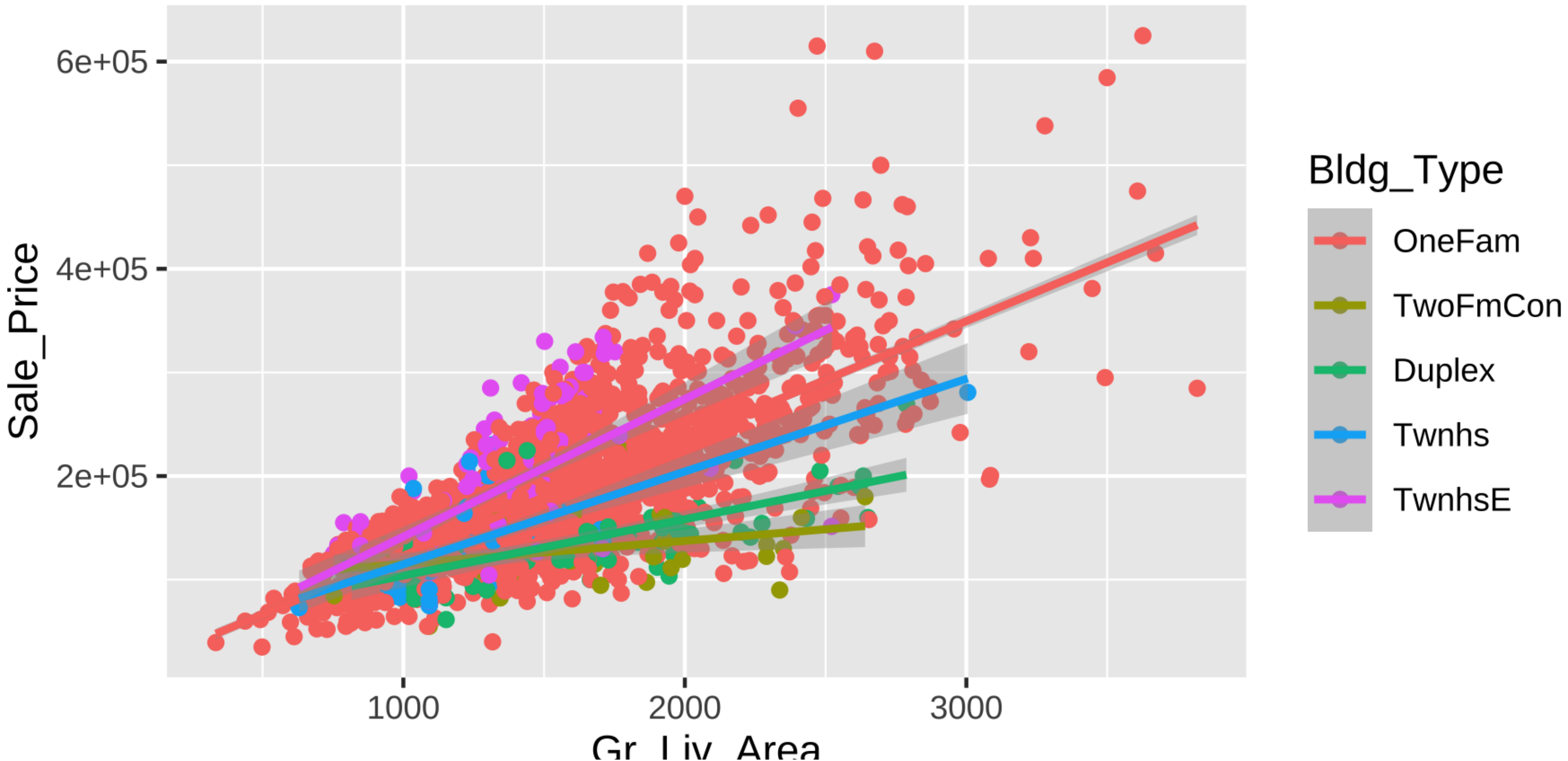
```
if remodeled:
  Sale_Price = 22643.4 + 109.1 * Gr_Liv_Arera - -18424.1 * (1)
  Sale_Price = (22643.4 - -18424.1) + 109.1 * Gr_Liv_Arera
else:
  Sale_Price = 22643.4 + 109.1 * Gr_Liv_Arera
```

```r
ggplot(baked_ames_train, aes(x = Gr_Liv_Area, y = Sale_Price, color = remodeled_TRUE.)) +
  geom_point() +
  geom_function(fun = function(x) (22643.4248 - 18424.0789) + 109.1132 * x, color = "blue") +
  geom_function(fun = function(x) 22643.4248 + 109.1132 * x, color = "green")
```



More than two options

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



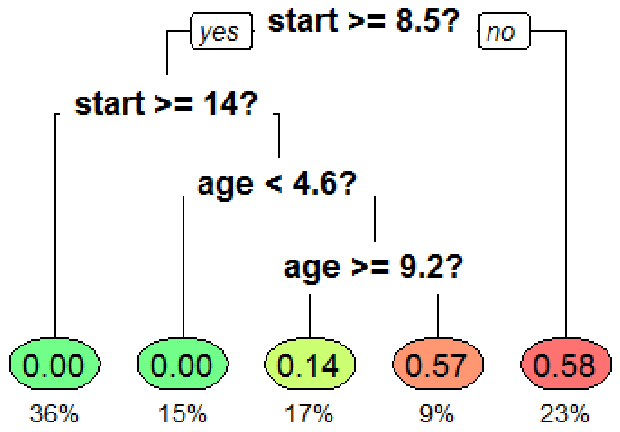
```r
ames_train %>% count(Bldg_Type) %>% kable()
```

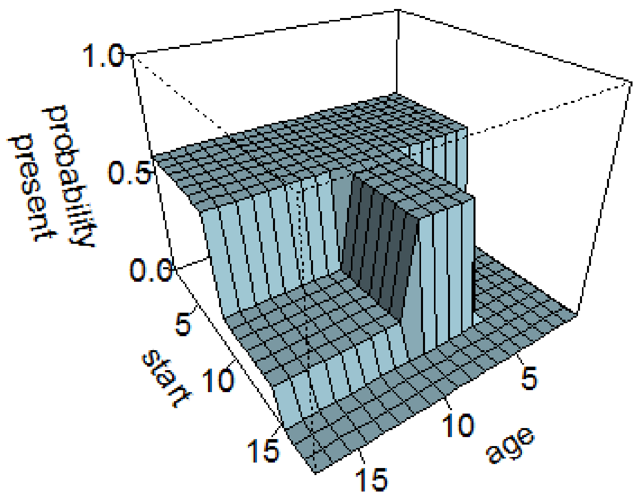
Bldg_Type	n
OneFam	1355
TwoFmCon	37
Duplex	43
Twnhs	56
TwnhsE	117

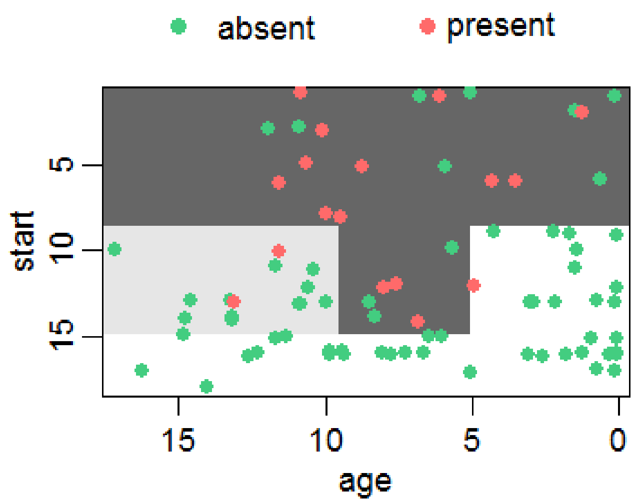
```r
ames_recipe_4 <-
  recipe(Sale_Price ~ Gr_Liv_Area + Bldg_Type, data = ames_train) %>%
  step_dummy(Bldg_Type) %>%
  #step_range(all_numeric(), -all_outcomes(), min = 0, max = 1) %>%
  prep()
baked_ames_train <-
  ames_recipe_4 %>% bake(new_data = ames_train_2)
baked_ames_train %>% head(5) %>% knitr::kable(format = "html")
```

Gr_Liv_Area	Sale_Price	Bldg_Type_TwoFmCon	Bldg_Type_Duplex	Bldg_Type_Twnhs	Bldg_Type_TwnhsE
896	105000	0	0	0	0
1329	172000	0	0	0	0
1629	189900	0	0	0	0
1604	195500	0	0	0	0
1804	189000	0	0	0	0

Another kind of model: Decision Trees



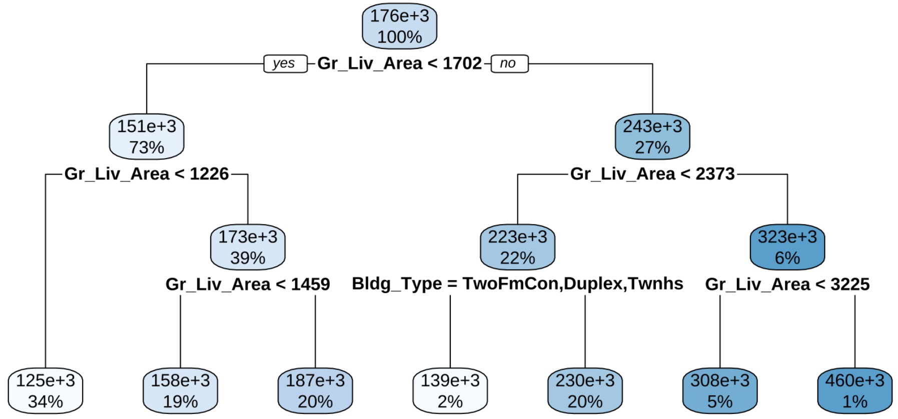




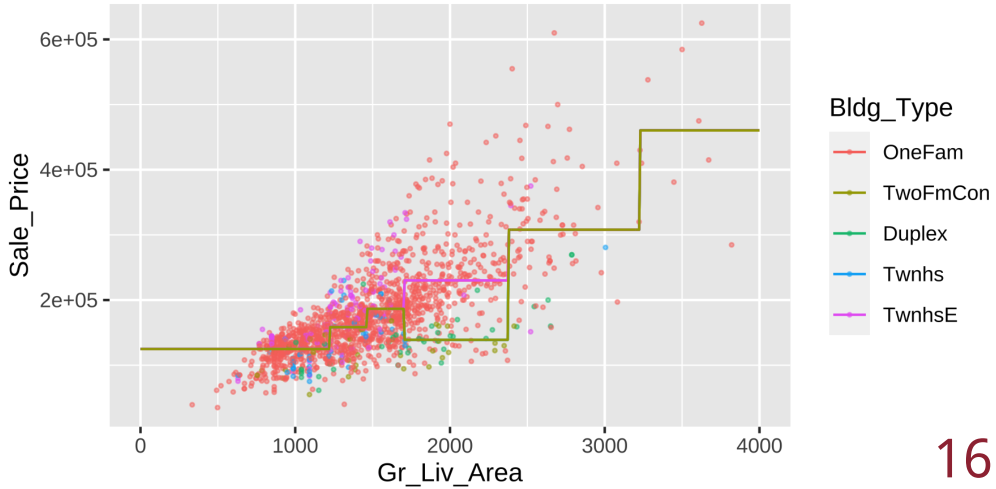
Fit a tree to data: same approach, different model

```
decision_tree_fit <- decision_tree(mode = "regression", tree_depth = 3) %>%
  set_engine("rpart") %>%
  fit(Sale_Price ~ Gr_Liv_Area + Bldg_Type, data = ames_train)
```

```
decision_tree_fit$fit %>% rpart.plot::rpart.plot(roundint = FALSE)
```



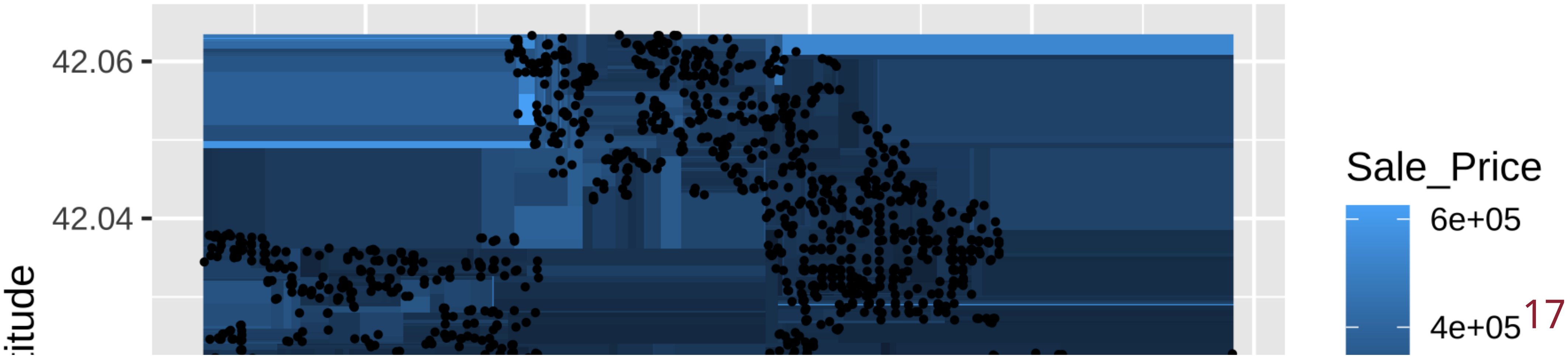

```
sweep_model <- function(model, var_to_sweep, sweep_min, sweep_max, ...) {
  X <- expand_grid(!!enquo(var_to_sweep) := seq(sweep_min, sweep_max, length.out = 5
  model %>%
    predict(X) %>%
    bind_cols(X)
}
ggplot(ames_train, aes(x = Gr_Liv_Area, y = Sale_Price, color = Bldg_Type)) +
  geom_point(alpha = .5, size = .5) +
  geom_line(data = sweep_model(
    decision_tree_fit, Gr_Liv_Area, 0, 4000, Bldg_Type = levels(ames_train$Bldg_Type
    mapping = aes(y = .pred))
```



```r
seq_matching_range <- function(x, length.out = 500) { seq(min(x), max(x), length.out = length.out)}
example_data <- expand_grid(
  Latitude = seq_matching_range(ames_train$Latitude),
  Longitude = seq_matching_range(ames_train$Longitude)
)

example_data <- decision_tree(mode = "regression", cost_complexity = 1e-6, min_n = 2, tree_depth = 3
  set_engine("rpart") %>%
  fit(Sale_Price ~ Latitude + Longitude, data = ames_train) %>%
  predict(example_data) %>%
  rename(Sale_Price = .pred) %>%
  bind_cols(example_data)

ggplot(ames_train_2, aes(x = Longitude, y = Latitude)) +
  geom_raster(data = example_data, mapping = aes(fill = Sale_Price)) +
  geom_point(size = .5)
```



Two kinds of regression models

Linear Regression

- *To make a prediction*: multiply terms by constants, sum it all up
- *Conditional logic* by explicitly transforming data to invent special terms
- *Output looks like* lines (or curves, if you add x^2 terms)

```
Sale_Price =
   22643
   + 18424  * (1 if remodeled)
   + 380368 * Gr_Liv_Area
```

Decision Tree Regression

- *To make a prediction*: follow conditional logic rules (determined automatically from data) to output a number.
- *Output looks like* stair-steps