Feature Engineering

K Arnold

Feedback

Good

- Coding together
- Motivating: lab, hw, visualizations, applications
- Connecting with me (and, sometimes, connecting with others)

(Oops, accidentally asked these questions twice!)

To improve

- Structure in class
- Lots of different things to do. So:
  - Check-in quizzes open at beginning of class. Will leave 1 min at end to fill out.
  - Flexible teamwork: each cohort will get 5 repos. Split up as you desire.
  - Goal: no required teamwork outside of class time
- Support outside of class
  - Many students have reached out on Teams at all hours.

Logistics

- Project: finish next week
  - Delve into the *data*: source? assumptions?
  - Delve into the vis *design*: which retinal variables chosen for which data variables, and why?
- Midterm Quiz: Quiz 8 open for a week, similar structure to Quiz 7
- Review sessions on Teams: ask about any past assignment!

Plan:

- Mon: decision tree regression
- Wed: decision tree classification
- Fri: lab about **overfitting**

General Hints

- Visualization:
  - What *glyph* represents each *observation*?
  - What *attributes*/aesthetics does that glyph have? (x, y, width, color, ...)
  - What *controls* each aesthetic? ("each party has a *y* position", ...)
- Wrangling
  - What does the *input* look like? (Translate the first row into a data sentence in English.)
  - What does the *output* need to look like? (Again, write a sentence.)
  - What *sequence of steps* needs to happen? (e.g., `filter-group_by-summarize-arrange`)
- Modeling
  - What *quantity* are you trying to predict?
  - What *error measure* will tell you the prediction is good / bad?
  - What *features* can help you make that prediction?

Q&A

> Why don't we just use `lm(y ~ x)` like other stats classes?

- We'll be using many other kinds of models; we're starting with linear models because many people have seen them before.
- `tidymodels` gives a unified interface to lots of different models
- Formulas support some kinds of feature engineering (e.g., `y ~ x1 * x2`) but limited.

> Can we use categorical (nominal) variables in predictive models?

Yes, we'll do that today!

> Can we see that dashboard?

https://rsconnect.calvin.edu/mi-covid/

Recipes

Why

Recipes can help us:

- Add expressive power (like conditional logic) to simple models
- Make the model more (or less!) understandable

What

A **recipe** is a data processing *pipeline* (like `%>%`) where the steps can be "smart".

> Smart?

like learning what range the data values fall in, to be able to scale them.

Setup

```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
```

Recipes

```r
ames_recipe_1 <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data = ames_train) %>%
  prep()
ames_recipe_1 %>% summary()
```

```
## # A tibble: 4 x 4
##   variable    type    role      source
##   <chr>       <chr>   <chr>     <chr>
## 1 Gr_Liv_Area numeric predictor original
## 2 Latitude    numeric predictor original
## 3 Longitude   numeric predictor original
## 4 Sale_Price  numeric outcome   original
```

```r
ames_recipe_1 %>% bake(new_data = ames_train)
```

```
## # A tibble: 1,608 x 4
##   Gr_Liv_Area Latitude Longitude Sale_Price
##         <dbl>    <dbl>     <dbl>      <dbl>
## 1         896     42.1     -93.6     105000
## 2        1329     42.1     -93.6     172000
## 3        1629     42.1     -93.6     189900
## 4        1604     42.1     -93.6     195500
## 5        1804     42.1     -93.6     189000
## 6        1655     42.1     -93.6     175900
## # … with 1,602 more rows
```

Workflows

`workflow` = `recipe` + `model`

```
workflow1 <- workflow() %>%
  add_model(linear_reg() %>% set_engine("lm")) %>%
  add_recipe(ames_recipe_1)
```

Workflows can `fit` and `predict`. First let's `fit` it on our training data...

```
fitted_workflow1 <- fit(workflow1, data = ames_train)
```

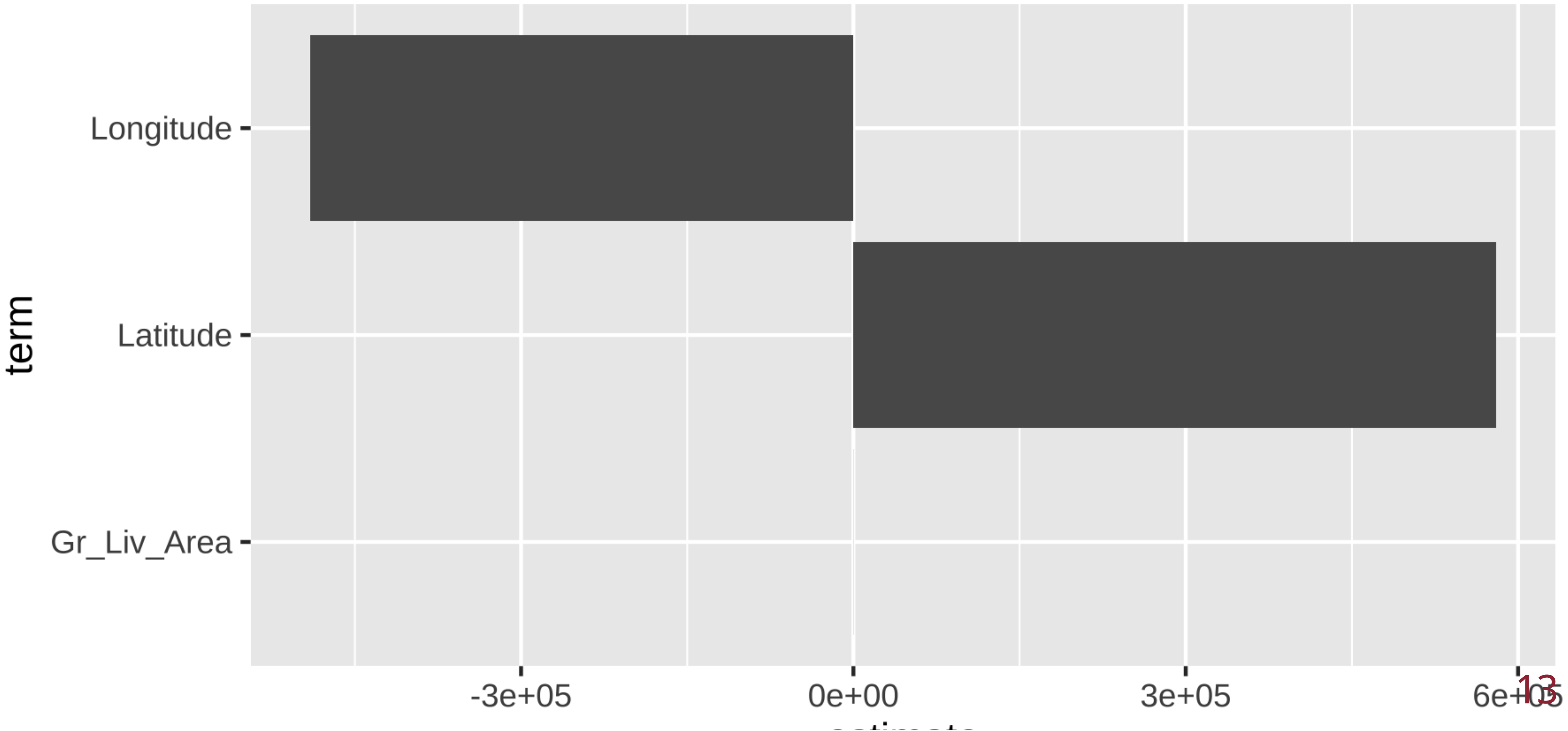
Now let's see what it predicts for our example home. (write this down)

```
fitted_workflow1 %>% predict(example_home)
```

```
## # A tibble: 1 x 1
##     .pred
##     <dbl>
## 1 193865.
```

Let's peek inside the model...

```r
fitted_workflow1 %>%
  tidy() %>%
  filter(term != "(Intercept)") %>%
  ggplot(aes(x = estimate, y = term)) + geom_col()
```



Feature Engineering 101: Scaling the features

The features have very different ranges

```
ames_train %>%
  select(Gr_Liv_Area, Latitude, Longitude) %>%
  summary()
```

```
##   Gr_Liv_Area       Latitude        Longitude     
##  Min.   : 334   Min.   :41.99   Min.   :-93.69  
##  1st Qu.:1103   1st Qu.:42.02   1st Qu.:-93.66  
##  Median :1432   Median :42.03   Median :-93.64  
##  Mean   :1483   Mean   :42.03   Mean   :-93.64  
##  3rd Qu.:1734   3rd Qu.:42.05   3rd Qu.:-93.62  
##  Max.   :3820   Max.   :42.06   Max.   :-93.58  
```

So the coefficients need to have very different scales to have a similar effect.

```
tidy(fitted_workflow1) %>% select(1:2)
```

```
## # A tibble: 4 x 2
##   term          estimate
##   <chr>            <dbl>
## 1 (Intercept) -70302361.
## 2 Gr_Liv_Area       101.
## 3 Latitude       580250.
```

Solution: scale the features to all be in the same range

```r
ames_recipe_2 <-
  recipe(Sale_Price ~ Gr_Liv_Area + Latitude + Longitude, data = ames_train) %>%
  step_range(all_numeric(), -all_outcomes(), min = 0, max = 1) %>%
  prep()
ames_recipe_2 %>% bake(new_data = ames_train) %>% summary()
```

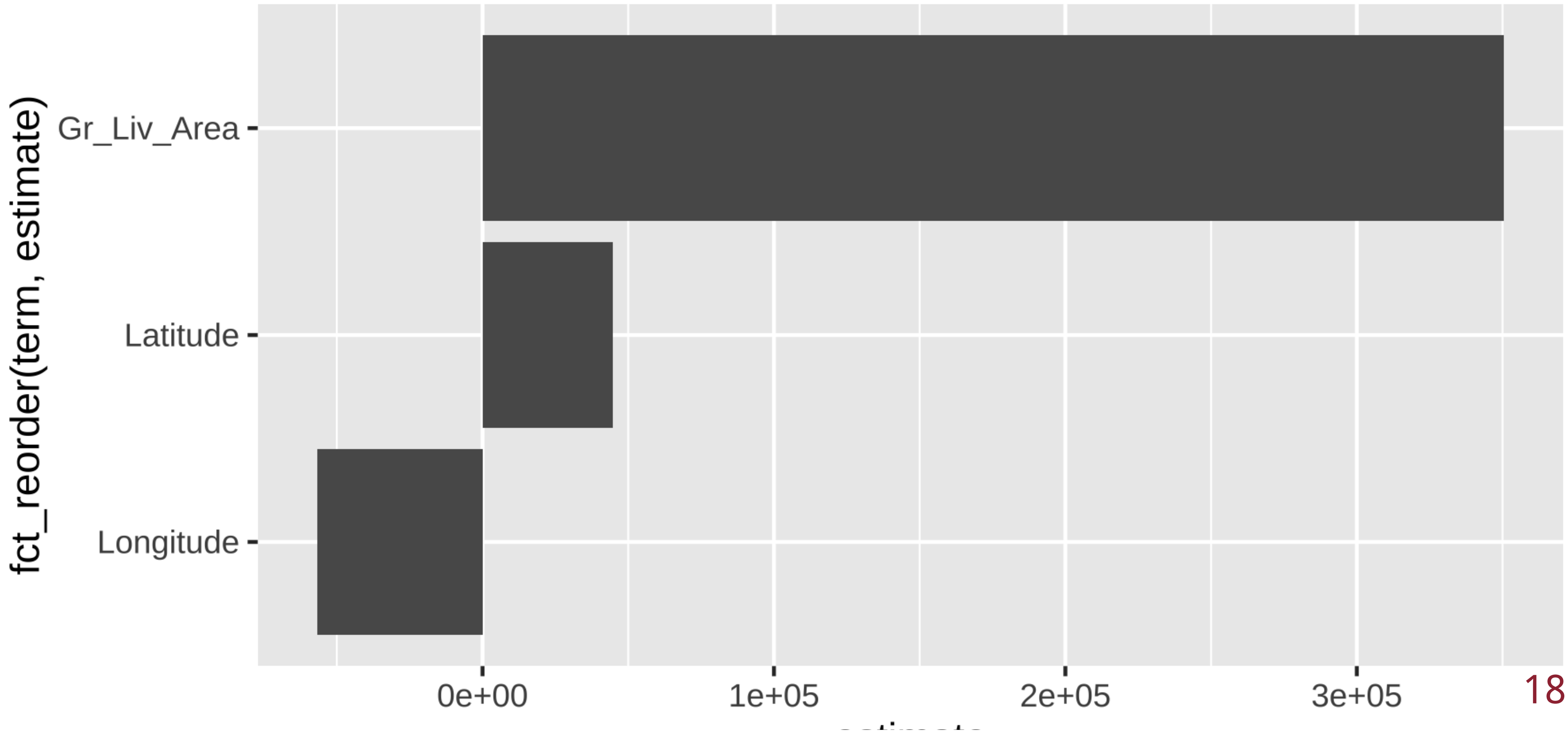
```
##   Gr_Liv_Area        Latitude         Longitude         Sale_Price
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 35000
##  1st Qu.:0.2206   1st Qu.:0.4558   1st Qu.:0.2724   1st Qu.:129000
##  Median :0.3148   Median :0.6248   Median :0.4488   Median :159898
##  Mean   :0.3297   Mean   :0.6141   Mean   :0.4355   Mean   :176011
##  3rd Qu.:0.4016   3rd Qu.:0.7987   3rd Qu.:0.6192   3rd Qu.:205000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :625000
```

```r
fitted_workflow2 <- workflow() %>%
  add_model(linear_reg() %>% set_engine("lm")) %>%
  add_recipe(ames_recipe_2) %>%
  fit(data = ames_train)
```

```r
fitted_workflow2 %>% predict(example_home)
```

```
## # A tibble: 1 x 1
##     .pred
##     <dbl>
## 1 193865.
```

```r
fitted_workflow2 %>%
  tidy() %>%
  filter(term != "(Intercept)") %>%
  ggplot(aes(x = estimate, y = fct_reorder(term, estimate))) + geom_col()
```



Recipe steps remember things from training data

- Remember we had: `step_range(..., min = 0, max = 1)`
- `output = (input - input_min) / (input_max - input_min)`
- It had to remember `input_min` and `input_max`!

Question: Suppose we apply this recipe to the **test set**. *What do we expect the minimum and maximum values to be* for `Gr_Liv_Area` etc.?

Recipe steps remember things from training data

- Remember we had: `step_range(..., min = 0, max = 1)`
- `output = (input - input_min) / (input_max - input_min)`
- It had to remember `input_min` and `input_max`!

Question: Suppose we apply this recipe to the **test set**. *What do we expect the minimum and maximum values to be* for `Gr_Liv_Area` etc.?

```
fitted_workflow2 %>%
  pull_workflow_prepped_recipe() %>%
  bake(new_data = ames_test) %>%
  summary()
```

```
##   Gr_Liv_Area         Latitude            Longitude           Sale_Price    
##  Min.   :0.02983   Min.   :0.0001431   Min.   :0.002473   Min.   : 45000  
##  1st Qu.:0.21916   1st Qu.:0.4561704   1st Qu.:0.276807   1st Qu.:130438  
##  Median :0.31268   Median :0.6255804   Median :0.464684   Median :157000  
##  Mean   :0.32349   Mean   :0.6145951   Mean   :0.445049   Mean   :173960  
##  3rd Qu.:0.39472   3rd Qu.:0.8007264   3rd Qu.:0.619621   3rd Qu.:207500  
##  Max.   :0.90620   Max.   :0.9994147   Max.   :0.806102   Max.   :615000  
```

What computations can a linear model do?

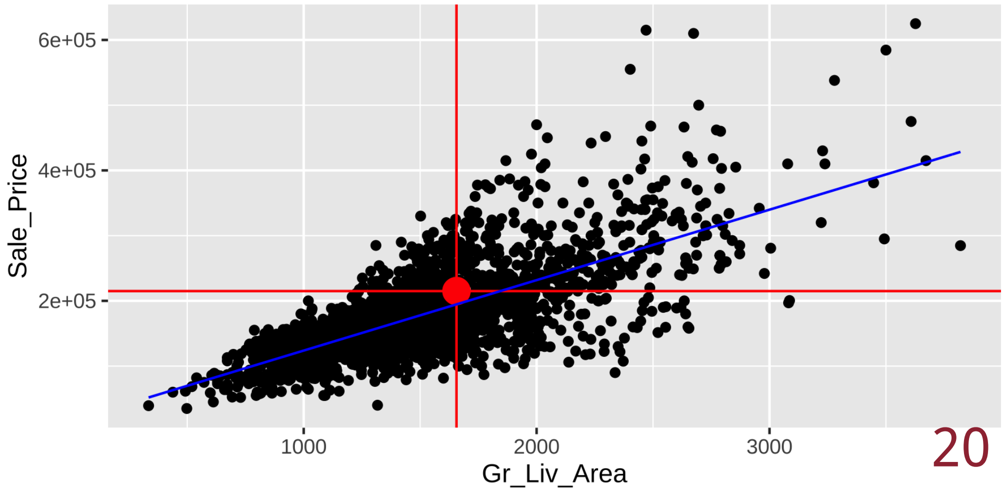
- *Add* up terms.
- Each term: *multiply* a number by a constant.

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

```
intercept + coef_living_area * 1610
```

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

```
ggplot(ames, aes(x = Gr_Liv_Area, y = Sale_Pri
  geom_point() +
  geom_hline(yintercept = example_home$Sale_Pr
  geom_vline(xintercept = example_home$Gr_Liv_
  geom_point(data = example_home, color = 'red
  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 %>%
  mutate(remodeled = Year_Remod_Add != Year_Built) %>%
  ggplot(aes(x = Gr_Liv_Area, y = Sale_Price, color = remodeled)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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 = case_when(
    Year_Built == Year_Remod_Add ~ "no",
    TRUE                         ~ "yes") %>%
    as_factor()
  )
```

```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()
ames_recipe_3 %>% bake(new_data = ames_train_2) %>% head(10) %>% knitr::kable(format = "html")
```

Gr_Liv_Area	Sale_Price	remodeled_yes
0.1612163	105000	0
0.2854274	172000	0
0.3714859	189900	1
0.3643144	195500	0
0.4216867	189000	0
0.3789443	175900	1