Lecture 7

Linear regression

Julian Reif
Fall 2025

RStudio setup for this lecture

- Log into RStudio on your Amazon EC2 instance
  - Use AMI `FIN550-RStudio` with IAM role `BigDataEC2Role`

- Copy file `lecture-07-tryit.Rmd` from the course Amazon S3 bucket to this folder

```
# Enter this command via RStudio Terminal
aws s3 cp --recursive s3://bigdata-fin550-reif/lecture-07 ~/fin550/lecture-07
```

Linear regression theory

Recall: machine learning

> "All models are wrong, but some are useful"
>
> --- George Box, statistician

- Goal is to model f, the systematic relationship between Y and X

Regression vs classification

- Regression is commonly used when outcome variable is *continuous*
  - Income
  - Spending
  - Height

Goal: model $f(X) = E[Y|X]$, the conditional expectation of Y given X

- By contrast, classification is used when outcome variable is *categorical*
  - Do you attend the University of Illinois? (yes/no)
  - What program are you enrolled in? (MSF/MSFE/MSBA/other)

Goal: model $f(X) = Pr[Y = label|X]$, the probability that Y is *label*, given X

Linear regression

- Approximates $f(X)$, the conditional expectation function, using a linear function:

$$f(X) = E[Y|X] = \beta_0 + \beta_1 X$$

- Plugging this into $Y = f(X) + \epsilon$ gives the equivalent form of this relationship as:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Univariate linear regression

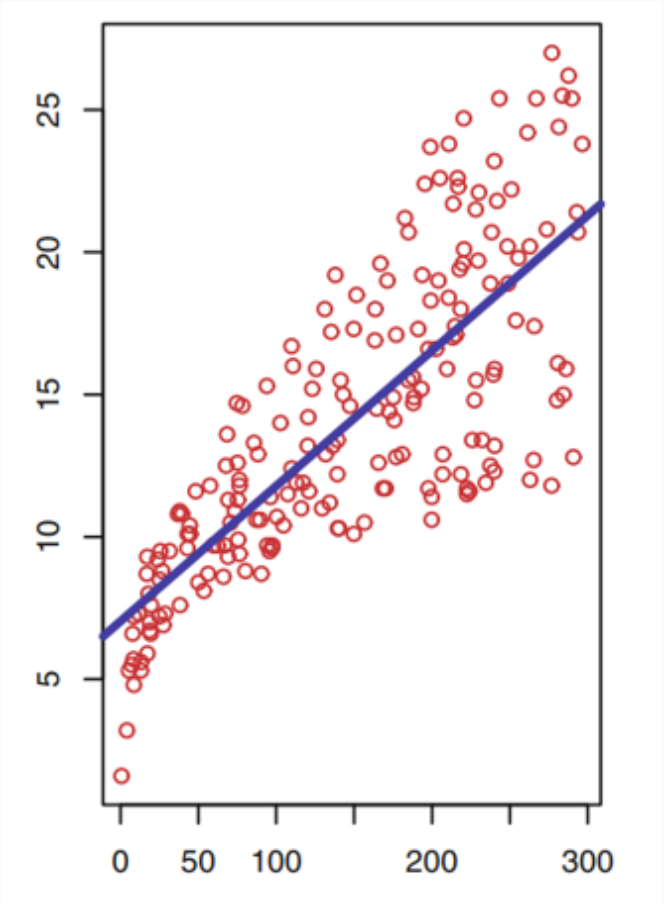
- Today, we will study univariate linear regression $(X = X_1)$

- Data (x, y), where x and y are vectors with length equal to n, the number of observations (rows)

- Let x_i and y_i denote the values of x and y for observation i, where $i = 1, \ldots, n$

Univariate linear regression

- Can be expressed graphically as a "trendline" drawn through a scatter plot of y versus x



Linear regression

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- β_0 and β_1 are the *parameters* (or *coefficients*) of the model
- In this univariate linear model, β_0 can be called the "intercept" and β_1 the "slope"
- Error term ϵ represents features that affect Y but are unavailable in our data

Our regression will estimate the true (fixed) parameters

- Let $\hat{\beta}_0$ and $\hat{\beta}_1$ be our best estimates of the parameters β_0 and β_1

- Given these parameter estimates, the model's prediction for observation i is

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- The prediction error is equal to $\hat{\epsilon}_i = y_i - \hat{y}_i$

Linear regression

- Answers prediction questions
  - "What is our best prediction of Y if we only know X?"

- Can also be used to describe correlations
  - "What is the relationship between X and Y in the data?"

- Later in the course, we will use linear regression for causal inference
  - "If we change X how will Y change?"

Linear regression minimizes RSS

- Linear regression minimizes the square of the prediction error

- Coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to minimize the Residual Sum of Squares (RSS):

$$
\begin{aligned}
RSS &= \sum_{i=1}^{n} {\hat{\epsilon}_i}^2 \\
&= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \sum_{i=1}^{n} \Big(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) \Big)^2
\end{aligned}
$$

- Why minimize the **square** and not just the sum?

Linear regression solution

- Minimizing RSS is a standard calculus optimization problem
- Let $\bar{y}$ and $\bar{x}$ be the mean of y and x, respectively
- The solution to this minimization problem is:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Linear regression in R

Load libraries, turn off scientific notation

```
library(tidyverse)
library(ggplot2)

# scipen: a penalty for deciding whether to print fixed or exponential notation.
# Positive values encourage fixed notation, negative encourage scientific notation
options(scipen=999)
```

Load and inspect our data

```
housing <- read_csv("lecture-07-housing.csv")
nrow(housing)
names(housing)
```

```
# [1] 300
#  [1] "PID"            "Lot_Area"       "House_Style"    "Overall_Qual"
#  [5] "Overall_Cond"   "Year_Built"     "Heating_QC"     "Central_Air"
#  [9] "Gr_Liv_Area"    "Bedroom_AbvGr"  "Fireplaces"     "Garage_Area"
# [13] "Mo_Sold"        "Yr_Sold"        "SalePrice"      "Basement_Area"
# [17] "Full_Bathroom"  "Half_Bathroom"  "Total_Bathroom" "Deck_Porch_Area"
# [21] "Age_Sold"       "Season_Sold"    "Garage_Type_2"  "Foundation_2"
# [25] "Masonry_Veneer" "Lot_Shape_2"    "House_Style2"   "Overall_Qual2"
# [29] "Overall_Cond2"  "Log_Price"      "Bonus"          "score"
```

Ames Iowa Housing Data, 2006-2010

Variable name	Definition
PID	Parcel identification number
Lot_Area	Lot size in square feet
House_Style	Style of dwelling
Overall_Qual	Rates the overall material and finish of the house
Overall_Cond	Rates the overall condition of the house
Year_Built	Original construction date
Heating_QC	Heating quality and condition
Gr_Liv_Area	Above grade (ground) living area square feet
SalePrice	Selling price
Mo_Sold	Month sold
Season_Sold	Season sold

Summarize a few variables

```
summary(housing$SalePrice)
summary(housing$Mo_Sold)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   35000  114000  135000  137525  159238  290000
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   1.000   4.000   6.000   5.907   7.000  12.000
```

Build a model of SalePrice, with just an intercept

$$SalePrice = \beta_0 + \epsilon$$

```
lm1 <- lm(SalePrice ~ 1, data = housing) # Outcome variable is SalePrice
summary(lm1)
```

```
#
# Call:
# lm(formula = SalePrice ~ 1, data = housing)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -102525  -23525   -2525   21713  152475
#
# Coefficients:
#             Estimate Std. Error t value            Pr(>|t|)
# (Intercept)   137525       2172   63.31 <0.0000000000000002 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
```

Model with only an intercept

```
# lm1 is a linear model "object"

# Extract the value of the alpha coefficient from this object
lm1$coefficients
```

```
# (Intercept)
#    137524.9
```

```
# Note: in this intercept model, the coefficient is just the mean
mean(housing$SalePrice)
```

```
# [1] 137524.9
```

Model with a numeric (continuous) predictor

- Estimate a model where sale price is a function of above-ground living area

$$SalePrice = \beta_0 + \beta_1 \text{Gr\_Liv\_Area} + \epsilon$$

- Above-ground living area is measured in square feet

```
summary(housing$Gr_Liv_Area)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#     334     952    1135    1131    1337    1500
```

Try it: Model with a numeric (continuous) predictor

```
lm2 <- lm(SalePrice ~ Gr_Liv_Area, data = housing)
summary(lm2)
```

Try it: Model with a numeric (continuous) predictor

```
lm2 <- lm(SalePrice ~ Gr_Liv_Area, data = housing)
summary(lm2)
```

```
#
# Call:
# lm(formula = SalePrice ~ Gr_Liv_Area, data = housing)
#
# Residuals:
#    Min     1Q Median     3Q    Max
# -77582 -20114    358  19208 122153
#
# Coefficients:
#              Estimate Std. Error t value            Pr(>|t|)
# (Intercept) 18583.434   8213.438   2.263              0.0244 *
# Gr_Liv_Area   105.189      7.115  14.784 <0.0000000000000002 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 28620 on 298 degrees of freedom
# Multiple R-squared:  0.4231,    Adjusted R-squared:  0.4212
# F-statistic: 218.6 on 1 and 298 DF,  p-value: < 0.00000000000000022
```

- Parameter estimate for the intercept is 18583.434
- Parameter estimate for the slope of `Gr_Liv_Area` is 105.189
- Regression equation is `SalePrice=18583.434 + 105.189*Gr_Liv_Area`
- Model indicates that each additional square foot of `Gr_Liv_Area` is associated with about $105 higher sale price

Model with a numeric (continuous) predictor

```
# The lm2 object includes predicted values ("y hat")
head(lm2$fitted.values)
```

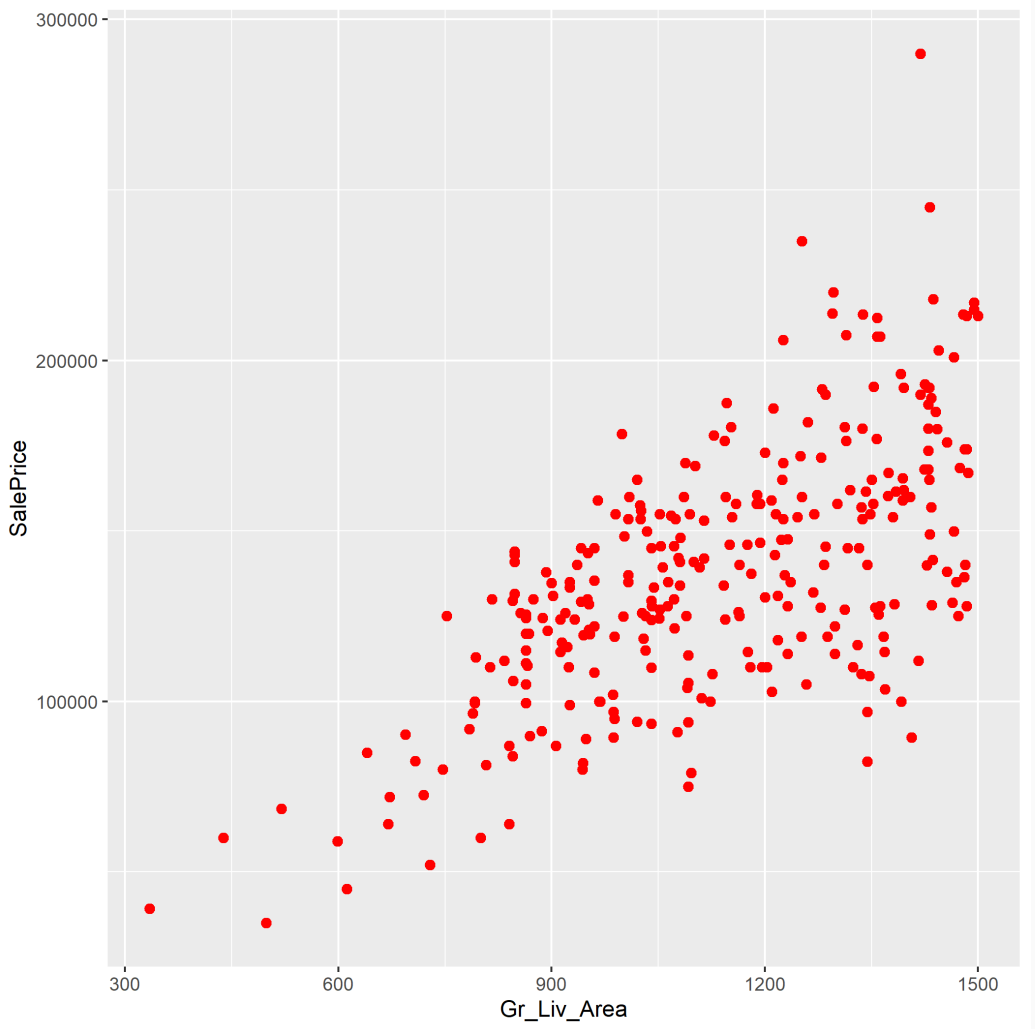
```
#         1         2         3         4         5         6
# 159326.34 153225.38 109466.75 139024.86 169003.73  97685.58
```

Try it: scatter plot plus trendline

```
# Plot SalePrice (y-axis) vs Gr_Liv_Area (x-axis)
# Make the scatterpoints red, with size=2
ggplot(housing, aes( )) +
  geom_point()
```

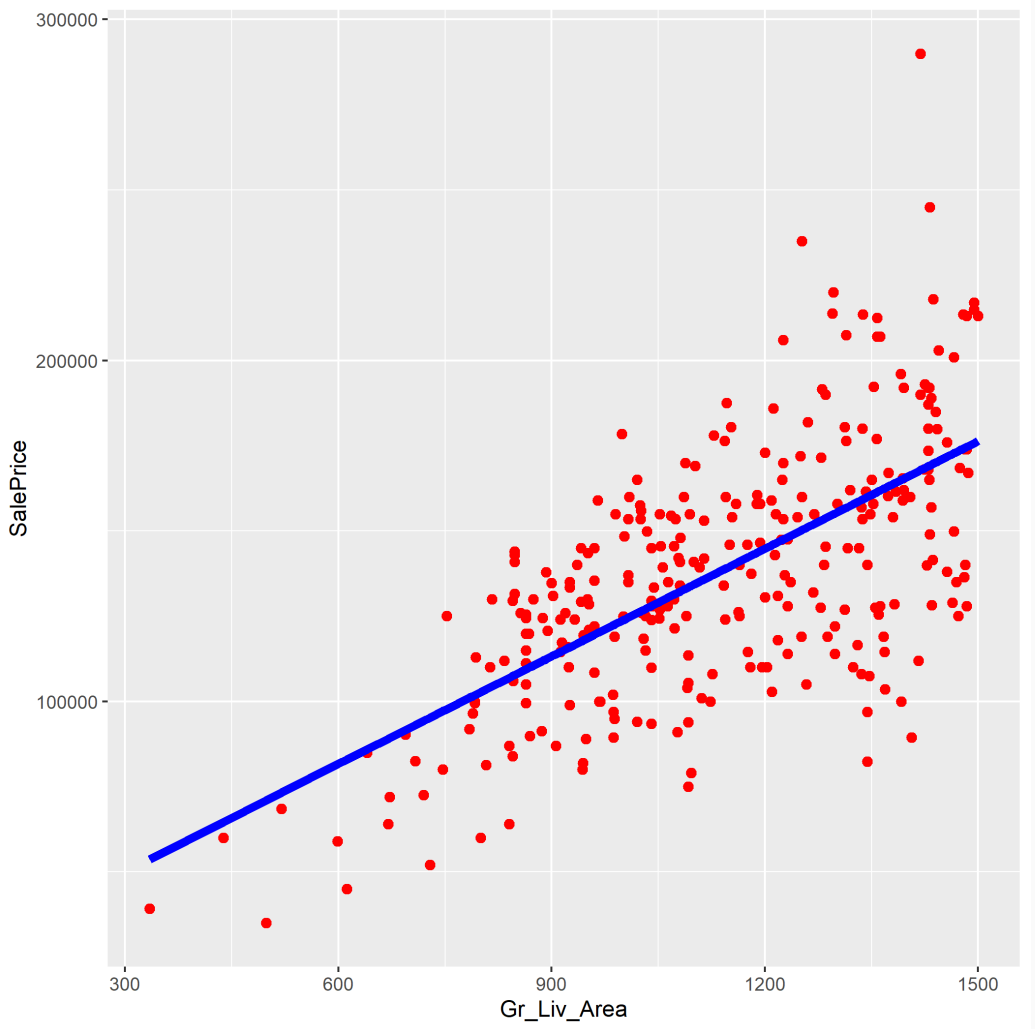
Try it: scatter plot plus trendline

```
# Plot SalePrice (y-axis) vs Gr_Liv_Area (x-axis)
# Make the scatterpoints red, with size=2
ggplot(housing, aes(x = Gr_Liv_Area, y = SalePrice)) +
  geom_point(color="red", size=2)
```



Try it: scatter plot plus trendline

```
# Plot SalePrice (y-axis) vs Gr_Liv_Area (x-axis)
# Make the scatterpoints red, with size=2
ggplot(housing, aes(x = Gr_Liv_Area, y = SalePrice)) +
  geom_point(color="red", size=2) +
  geom_line(aes(x = Gr_Liv_Area, y = lm2$fitted.values),
            color="blue", linewidth=2)
```



Model with categorical predictor

- Does the season when the house is sold matter for the sale price?
- `Season_Sold` is a categorical predictor

```
table(housing$Season_Sold)
```

```
#
#   1   2   3   4
#  35 100 130  35
```

- Does treating it as a continuous variable make sense?

$$SalePrice = \beta_0 + \beta_1 \text{Season\_Sold} + \epsilon$$

- No!

Model with categorical predictor

- Categorical predictors must be split up into their respective categories

- Create a set of **3 (= 4 - 1)** "dummy" variables that are equal to eithe 0 or 1
  - `dummy2`: equal to 1 if `Season_Sold==2`, 0 otherwise
  - `dummy3`: equal to 1 if `Season_Sold==3`, 0 otherwise
  - `dummy4`: equal to 1 if `Season_Sold==4`, 0 otherwise

- What happened to `Season_Sold==1`?
  - It is an "omitted" (baseline) category

Model with categorical predictor

- The function `as.factor()` automatically creates dummy variables for us

```
lm(SalePrice ~ as.factor(Season_Sold), data = housing)
```

```
#
# Call:
# lm(formula = SalePrice ~ as.factor(Season_Sold), data = housing)
#
# Coefficients:
#             (Intercept)  as.factor(Season_Sold)2  as.factor(Season_Sold)3
#                  120917                    19962                    16493
# as.factor(Season_Sold)4
#                   24060
```

$$\text{SalePrice} = 120917 + 19962*\text{dummy2} + 16493*\text{dummy3} + 24060*\text{dummy4}$$

Model with categorical predictor

```
SalePrice = 120917 + 19962*dummy2 + 16493*dummy3 + 24060*dummy4
```

- Predicted price for house sold in "omitted" season 1 (winter): `120917`
- Predicted price for house sold in season 2 (spring): `120917 + 19962 = 140879`
- Predicted price for house sold in season 3 (summer): `120917 + 16493 = 137410`
- Predicted price for house sold in season 4 (fall): `120917 + 24060 = 144977`

- Note: these predictions are just the `SalePrice` means for these groups

Categorical regressions are means

```
SalePrice = 120917 + 19962*dummy2 + 16493*dummy3 + 24060*dummy4
```

```
housing %>%
  group_by(Season_Sold) %>%
  summarize(SalePrice_mean = mean(SalePrice))
```

```
# # A tibble: 4 × 2
#   Season_Sold SalePrice_mean
#         <dbl>          <dbl>
# 1           1        120917.
# 2           2        140879.
# 3           3        137410.
# 4           4        144977.
```

Categorical regressions are means

- Regressions with a categorical predictor always yield means
- Note: not necessarily true if you also include other predictors

```
# Does not report simple means
lm(SalePrice ~ Gr_Liv_Area + as.factor(Season_Sold), data = housing)
```

```
#
# Call:
# lm(formula = SalePrice ~ Gr_Liv_Area + as.factor(Season_Sold),
#     data = housing)
#
# Coefficients:
#             (Intercept)              Gr_Liv_Area  as.factor(Season_Sold)2
#                 13395.4                    103.7                    6813.0
# as.factor(Season_Sold)3  as.factor(Season_Sold)4
#                  8185.0                   8593.2
```

What happens if the regressor has non-numeric values?

```
# "Ex" = "Excellent","Gd" = "Good", "TA" = "Typical", "Fa" = "Fair"
typeof(housing$Heating_QC)
table(housing$Heating_QC)
lm(SalePrice ~ Heating_QC, data = housing)
```

```
# [1] "character"
#
#  Ex  Fa  Gd  TA
# 107  16  58 119
#
# Call:
# lm(formula = SalePrice ~ Heating_QC, data = housing)
#
# Coefficients:
#  (Intercept)  Heating_QCFa  Heating_QCGd  Heating_QCTA
#       154919        -57800        -24075        -24346
```

Creating dummy variables by hand

```r
# First 6 values
head(housing$Heating_QC)

# Create dummies using model.matrix. 0 specifies no intercept
head(model.matrix(~ 0 + Heating_QC, data = housing))
```

```
# [1] "Ex" "Ex" "TA" "Ex" "Ex" "TA"
#   Heating_QCEx Heating_QCFa Heating_QCGd Heating_QCTA
# 1            1            0            0            0
# 2            1            0            0            0
# 3            0            0            0            1
# 4            1            0            0            0
# 5            1            0            0            0
# 6            0            0            0            1
```

Creating dummy variables by hand

```
# Turn the matrix into a data frame
Heating_QC_dummy <- as.data.frame(model.matrix(~ 0 + Heating_QC, data = housing))
head(Heating_QC_dummy)
```

```
#   Heating_QCEx Heating_QCFa Heating_QCGd Heating_QCTA
# 1            1            0            0            0
# 2            1            0            0            0
# 3            0            0            0            1
# 4            1            0            0            0
# 5            1            0            0            0
# 6            0            0            0            1
```

Creating dummy variables by hand

```r
# Add to housing dataset, excluding the first dummy variable
housing <- cbind(housing, Heating_QC_dummy[,-1])

# fit a linear regression of Sale Price on three dummies
lm(SalePrice ~ Heating_QCFa + Heating_QCGd + Heating_QCTA, data = housing)
```

```
#
# Call:
# lm(formula = SalePrice ~ Heating_QCFa + Heating_QCGd + Heating_QCTA,
#     data = housing)
#
# Coefficients:
#  (Intercept)  Heating_QCFa  Heating_QCGd  Heating_QCTA
#       154919        -57800        -24075        -24346
```

Summary

- Regression is a popular prediction method for continuous outcome variables
- Estimate regression by minimizing the Residual Sum of Squares
- Categorical predictors must be turned into "dummy" variables

- Lab-07 due Sunday at 11:59pm
- Make sure to **stop your instance** when you are done working