Gov 50: 13. Regression

Matthew Blackwell

Harvard University

Roadmap

1. Prediction

2. Modeling with a line

3. Linear regression in R

1/ Prediction

Predicting my weight

Predicting weight with activity: `health` data

Name	Description
`date`	date of measurements
`active_calories`	calories burned
`steps`	number of steps taken (in 1,000s)
`weight`	weight (lbs)
`steps_lag`	steps on day before (in 1,000s)
`calories_lag`	calories burned on day before

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?
  - what's our best guess about my weight this morning if I know how many steps I took yesterday?

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?
  - what's our best guess about my weight this morning if I know how many steps I took yesterday?
- Terminology:

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?
    - what's our best guess about my weight this morning if I know how many steps I took yesterday?
- Terminology:
    - **Dependent/outcome variable**: what we want to predict (weight).

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?
    - what's our best guess about my weight this morning if I know how many steps I took yesterday?
- Terminology:
    - **Dependent/outcome variable**: what we want to predict (weight).
    - **Independent/explanatory variable**: what we're using to predict (steps).

Weight data

- Load the data:

Weight data

- Load the data:

```
library(gov50data)
health <- drop_na(health)
```

Weight data

- Load the data:

```
library(gov50data)
health <- drop_na(health)
```

- Plot the data:

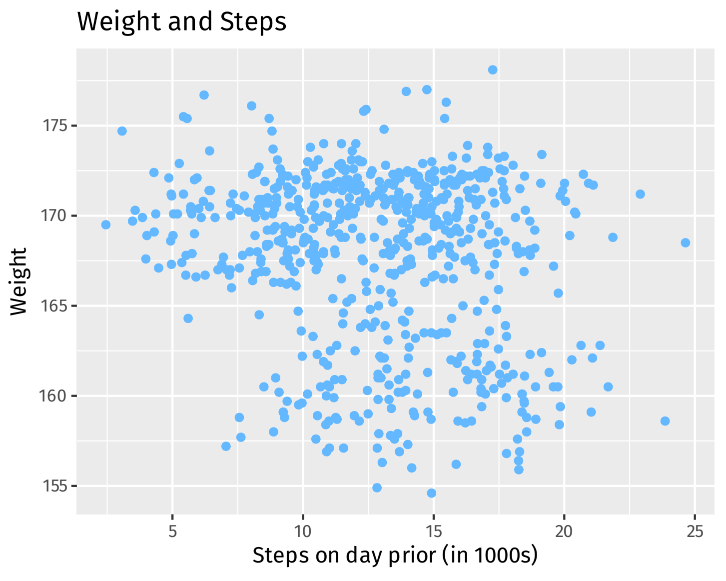
Weight data

- Load the data:

```
library(gov50data)
health <- drop_na(health)
```

- Plot the data:

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1") +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  )
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.
- Prediction with another variable: for any value of X, what's the best guess about Y?

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.
- Prediction with another variable: for any value of X, what's the best guess about Y?
    - Need a function $y = f(x)$ that maps values of X into predictions.

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.
- Prediction with another variable: for any value of X, what's the best guess about Y?
    - Need a function $y = f(x)$ that maps values of X into predictions.
    - **Machine learning**: fancy ways to determine $f(x)$

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.
- Prediction with another variable: for any value of X, what's the best guess about Y?
    - Need a function $y = f(x)$ that maps values of X into predictions.
    - **Machine learning**: fancy ways to determine $f(x)$
- Example: what if did 5,000 steps today? What's my best guess about weight?

Start with looking at a narrow strip of X

Let's find all values that round to 5,000 steps:

```
health |>
  filter(round(steps_lag) == 5)
```

```
## # A tibble: 12 x 6
##    date       active_calories steps weight steps_lag
##    <date>               <dbl> <dbl>  <dbl>     <dbl>
##  1 2015-09-08           1111. 15.2    169.      5.02
##  2 2015-12-12            728. 14.7    167.      5.36
##  3 2015-12-28            430.  8.94   170.      5.19
##  4 2016-01-29            475.  8.26   171.      4.95
##  5 2016-02-14            264.  5.42   172.      4.86
##  6 2016-02-15            892. 13.1    171.      5.42
##  7 2016-05-02            627. 11.8    170.      5.04
##  8 2016-06-27            352.  7.21   169.      4.93
##  9 2016-07-22            766. 14.8    167.      4.96
## 10 2016-11-25            452   9.4    173.      5.26
## 11 2016-11-28            577. 11.8    171.      4.97
## 12 2016-12-30            621. 12.4    176.      5.42
## # i 1 more variable: calorie_lag <dbl>
```

Best guess about Y for this X

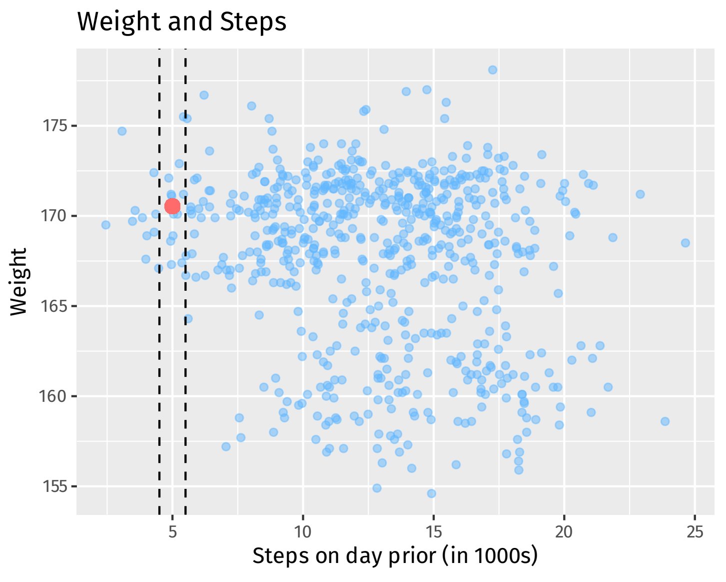
Best prediction about weight for a step count of roughly 5,000 is the average weight for observations around that value:

```
mean_wt_5k_steps <- health |>
  filter(round(steps_lag) == 5) |>
  summarize(mean(weight)) |>
  pull()
mean_wt_5k_steps
```

```
## [1] 171
```

Plotting the best guess

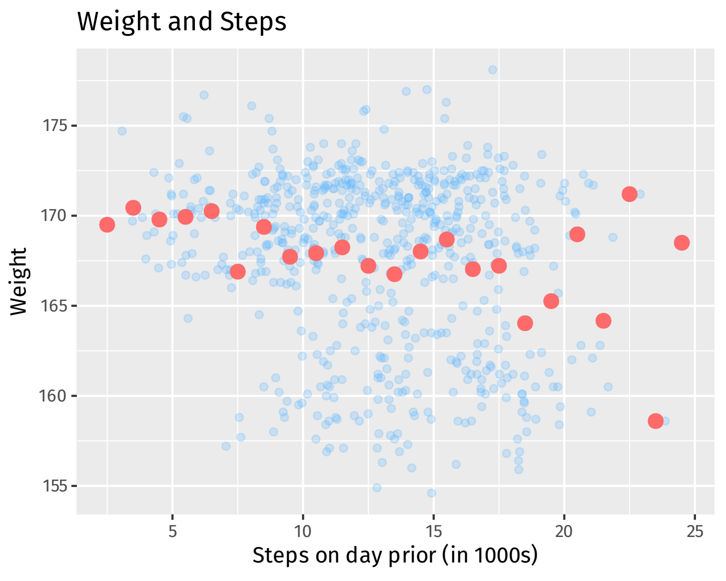
```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.5) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_vline(xintercept = c(4.5, 5.5), linetype = "dashed") +
  geom_point(aes(x = 5, y = mean_wt_5k_steps), color = "indianred1",
             size = 3)
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Binned means

We can use a `stat_summary_bin()` to add these binned means all over the scatter plot:

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 3,
                   geom = "point", binwidth = 1)
```



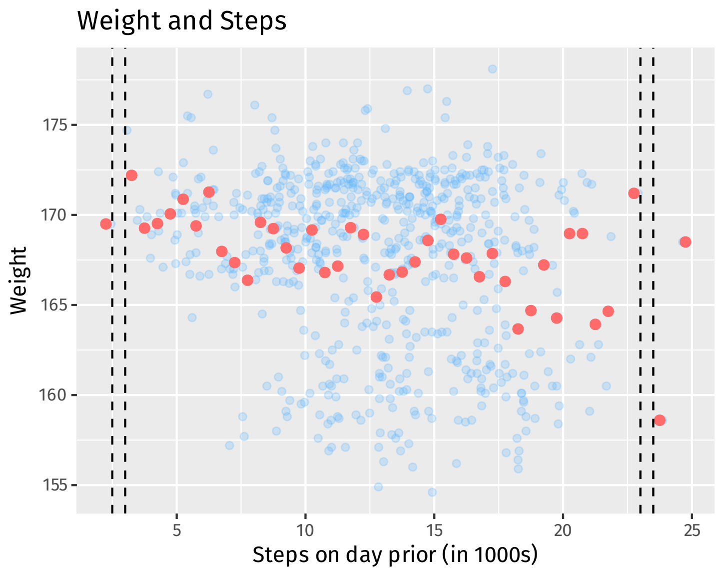
Weight and Steps
175
170
165
160
155
Weight
5
10
15
20
25
Steps on day prior (in 1000s)

Smaller bins

But what happens when we make the bins too small?

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 2,
                   geom = "point", binwidth = 0.5) +
  geom_vline(xintercept = c(2.5, 3, 23, 23.5), linetype = "dashed")
```

Gaps and bumps:



2/ Modeling with a line

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

- Problem: for any line we draw, not all the data is on the line.

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

- Problem: for any line we draw, not all the data is on the line.
  - Some points will be above the line, some below.

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

- Problem: for any line we draw, not all the data is on the line.
    - Some points will be above the line, some below.
    - Need a way to account for **chance variation** away from the line.

Linear regression model

- Model for the line of best fit:

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.
  - Each observation allowed to be off the regression line.

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.
    - Each observation allowed to be off the regression line.
    - Chance errors are 0 on average.

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.
  - Each observation allowed to be off the regression line.
  - Chance errors are 0 on average.
- Useful fiction: this model represents the **data generating process**

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.
    - Each observation allowed to be off the regression line.
    - Chance errors are 0 on average.
- Useful fiction: this model represents the **data generating process**
    - George Box: "all models are wrong, some are useful"

Interpreting the regression line

$$Y_i = \alpha + \beta \cdot X_i + \epsilon_i$$

- **Intercept** α: average value of Y when X is 0

Interpreting the regression line

$$Y_i = \alpha + \beta \cdot X_i + \epsilon_i$$

- **Intercept** α: average value of Y when X is 0
  - Average weight when I take 0 steps the day prior.

Interpreting the regression line

$$Y_i \quad = \quad \alpha + \beta \cdot X_i + \epsilon_i$$

- **Intercept** α: average value of Y when X is 0
    - Average weight when I take 0 steps the day prior.
- **Slope** β: average change in Y when X increases by one unit.

Interpreting the regression line

$$Y_i = \alpha + \beta \cdot X_i + \epsilon_i$$

- **Intercept** α: average value of Y when X is 0
  - Average weight when I take 0 steps the day prior.
- **Slope** β: average change in Y when X increases by one unit.
  - Average decrease in weight for each additional 1,000 steps.

Estimated coefficients

- Parameters: α, β

Estimated coefficients

- Parameters: α, β
    - Unknown features of the **data-generating process**.

Estimated coefficients

- Parameters: α, β
    - Unknown features of the **data-generating process**.
    - Chance error makes these impossible to observe directly.

Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$

Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$
  - An **estimate** is our best guess about some parameter.

Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$
  - An **estimate** is our best guess about some parameter.
- **Regression line**: $\widehat{Y} = \hat{\alpha} + \hat{\beta} \cdot x$

Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$
  - An **estimate** is our best guess about some parameter.
- **Regression line**: $\widehat{Y} = \hat{\alpha} + \hat{\beta} \cdot x$
  - Average value of Y when X is equal to x.

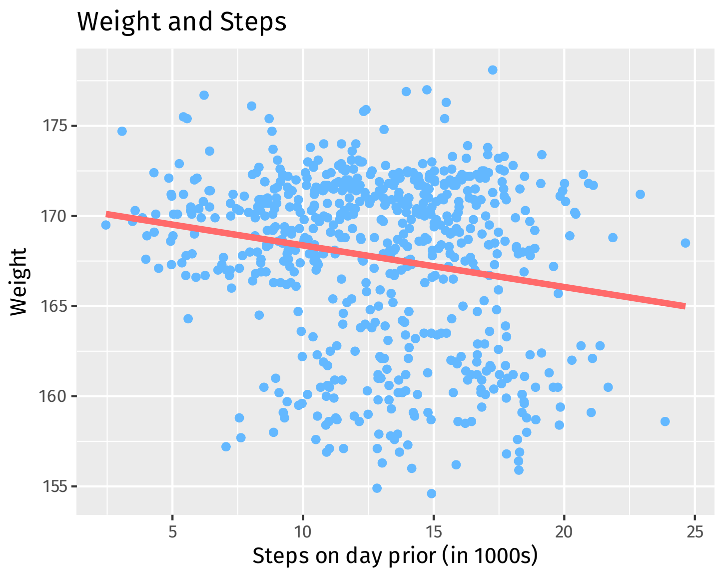
Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$
  - An **estimate** is our best guess about some parameter.
- **Regression line**: $\widehat{Y} = \hat{\alpha} + \hat{\beta} \cdot x$
  - Average value of Y when X is equal to x.
  - Represents the best guess or **predicted value** of the outcome at x.

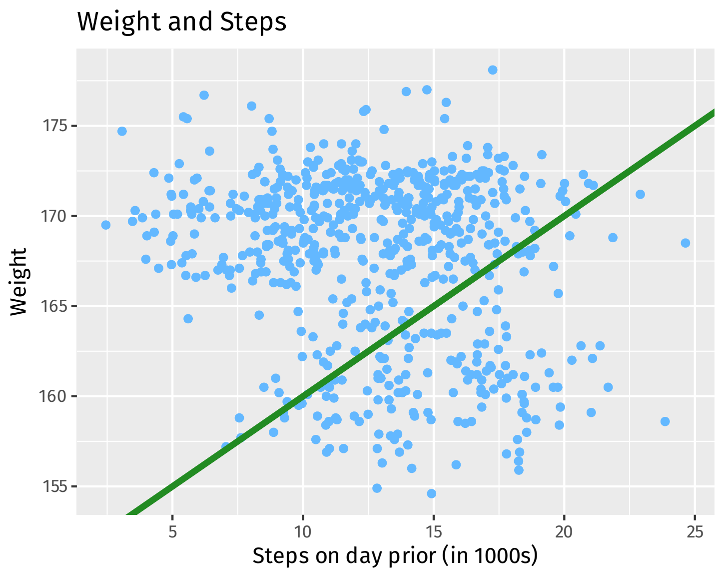
Line of best fit

```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1") +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```

Line of best fit



Why not this line?



Prediction error

Let's understand the **prediction error** for a line with intercept a and slope b.

Prediction error

Let's understand the **prediction error** for a line with intercept a and slope b.

Fitted/predicted value for unit i:

$$a + b \cdot X_i$$

Prediction error

Let's understand the **prediction error** for a line with intercept a and slope b.

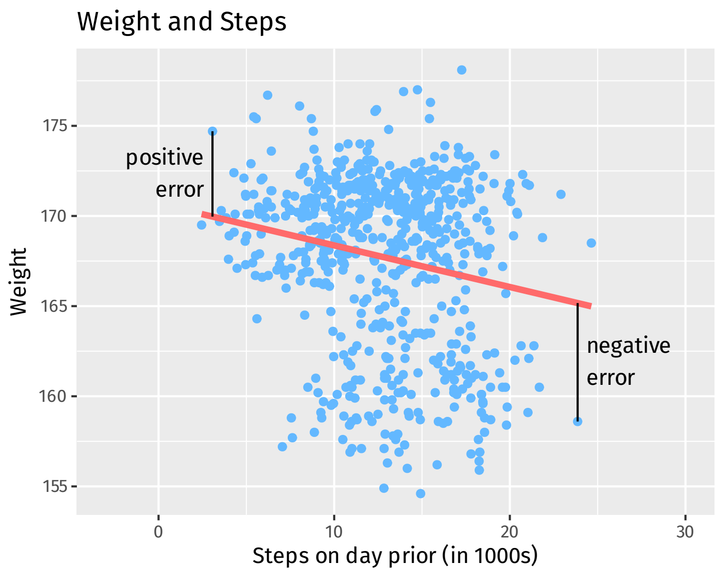
Fitted/predicted value for unit i:

$$a + b \cdot X_i$$

Preidiction error (residual):

$$\text{error = actual - predicted} = Y_i - (a + b \cdot X_i)$$

Prediction errors/residuals



Least squares

- Get these estimates by the **least squares method**.

Least squares

- Get these estimates by the **least squares method**.
- Minimize the **sum of the squared residuals** (SSR):

$$\text{SSR} = \sum_{i=1}^{n} (\text{prediction error}_i)^2 = \sum_{i=1}^{n} (Y_i - a - b \cdot X_i)^2$$

Least squares

- Get these estimates by the **least squares method**.
- Minimize the **sum of the squared residuals** (SSR):

$$\text{SSR} = \sum_{i=1}^{n} (\text{prediction error}_i)^2 = \sum_{i=1}^{n} (Y_i - a - b \cdot X_i)^2$$

- Finds the line that minimizes the magnitude of the prediction errors!

3/ Linear regression in R

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.

Linear regression in R

- R will calculate least squares line for a data set using `lm( )`.
    - Syntax: `lm(y ~ x, data = mydata)`

Linear regression in R

- R will calculate least squares line for a data set using `lm( )`.
    - Syntax: `lm(y ~ x, data = mydata)`
    - `y` is the name of the dependent variance

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.
  - Syntax: `lm(y ~ x, data = mydata)`
  - `y` is the name of the dependent variance
  - `x` is the name of the independent variable

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.
    - Syntax: `lm(y ~ x, data = mydata)`
    - `y` is the name of the dependent variance
    - `x` is the name of the independent variable
    - `mydata` is the data.frame where they live

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.
    - Syntax: `lm(y ~ x, data = mydata)`
    - `y` is the name of the dependent variance
    - `x` is the name of the independent variable
    - `mydata` is the data.frame where they live

```
fit <- lm(weight ~ steps_lag, data = health)
fit
```

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.
    - Syntax: `lm(y ~ x, data = mydata)`
    - `y` is the name of the dependent variance
    - `x` is the name of the independent variable
    - `mydata` is the data.frame where they live

```
fit <- lm(weight ~ steps_lag, data = health)
fit
```

```
##
## Call:
## lm(formula = weight ~ steps_lag, data = health)
##
## Coefficients:
## (Intercept)    steps_lag
##     170.675       -0.231
```

Linear regression in R

- R will calculate least squares line for a data set using `lm()`.
  - Syntax: `lm(y ~ x, data = mydata)`
  - `y` is the name of the dependent variance
  - `x` is the name of the independent variable
  - `mydata` is the data.frame where they live

```
fit <- lm(weight ~ steps_lag, data = health)
fit
```

```
##
## Call:
## lm(formula = weight ~ steps_lag, data = health)
##
## Coefficients:
## (Intercept)    steps_lag
##     170.675       -0.231
```

Coefficients

Use `coef()` to extract estimated coefficients:

```
coef(fit)
```

```
## (Intercept)   steps_lag
##     170.675      -0.231
```

Coefficients

Use `coef()` to extract estimated coefficients:

```
coef(fit)
```

```
## (Intercept)   steps_lag
##     170.675      -0.231
```

Interpretation: a 1-unit increase in X (1,000 steps) is associated with a decrease in the average weight of 0.231 pounds.

Coefficients

Use `coef()` to extract estimated coefficients:

```
coef(fit)
```

```
## (Intercept)   steps_lag
##     170.675      -0.231
```

Interpretation: a 1-unit increase in X (1,000 steps) is associated with a decrease in the average weight of 0.231 pounds.

Question: what would this model predict about the change in average weight for a 10,000 step increase in steps?

broom package

The broom package can provide nice summaries of the regression output.

`augment()` can show fitted values, residuals and other unit-level statistics:

```
library(broom)
augment(fit) |> head()
```

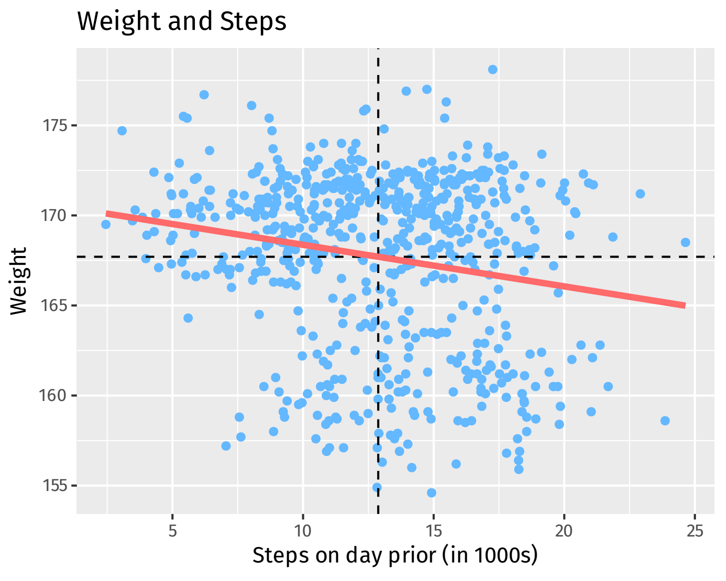
```
## # A tibble: 6 x 8
##   weight steps_lag .fitted  .resid    .hat .sigma    .cooksd
##    <dbl>     <dbl>   <dbl>   <dbl>   <dbl>  <dbl>      <dbl>
## 1   169.      17.5    167.  2.46   0.00369   4.68    5.13e-4
## 2   168       18.4    166.  1.57   0.00463   4.68    2.64e-4
## 3   167.      19.6    166.  1.05   0.00609   4.68    1.54e-4
## 4   168.      10.4    168. -0.0750 0.00217   4.68    2.80e-7
## 5   168.      18.7    166.  1.44   0.00496   4.68    2.38e-4
## 6   166.       9.14   169. -2.27   0.00296   4.68    3.49e-4
## # i 1 more variable: .std.resid <dbl>
```

Properties of least squares

Least squares line always goes through $(\overline{X}, \overline{Y})$.

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1") +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_hline(yintercept = mean(health$weight), linetype = "dashed") +
  geom_vline(xintercept = mean(health$steps_lag), linetype = "dashed") +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```

Least squares line always goes through $(\overline{X}, \overline{Y})$.



Properties of least squares line

Estimated slope is related to correlation:

$$\hat{\beta} = (\text{correlation of } X \text{ and } Y) \times \frac{\text{SD of } Y}{\text{SD of } X}$$

Properties of least squares line

Estimated slope is related to correlation:

$$\hat{\beta} = (\text{correlation of } X \text{ and } Y) \times \frac{\text{SD of } Y}{\text{SD of } X}$$

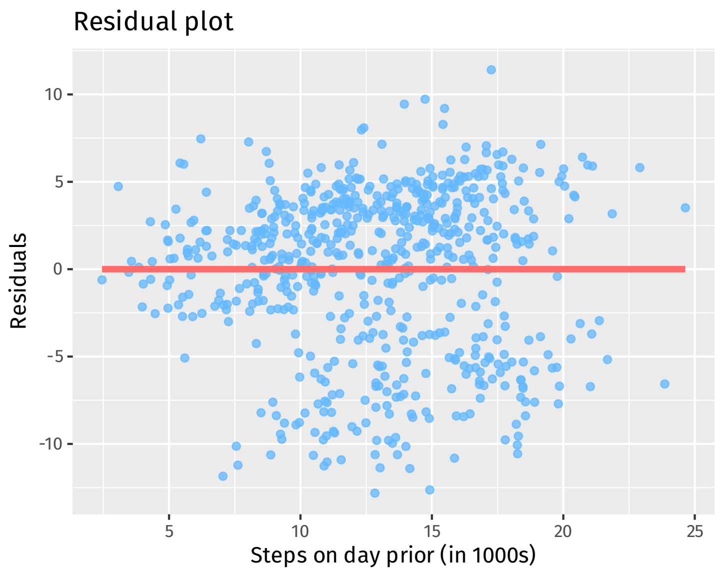
Mean of residuals is always 0.

```
augment(fit) |>
  summarize(mean(.resid))
```

```
## # A tibble: 1 x 1
##   `mean(.resid)`
##            <dbl>
## 1      -1.27e-13
```

Plotting the residuals

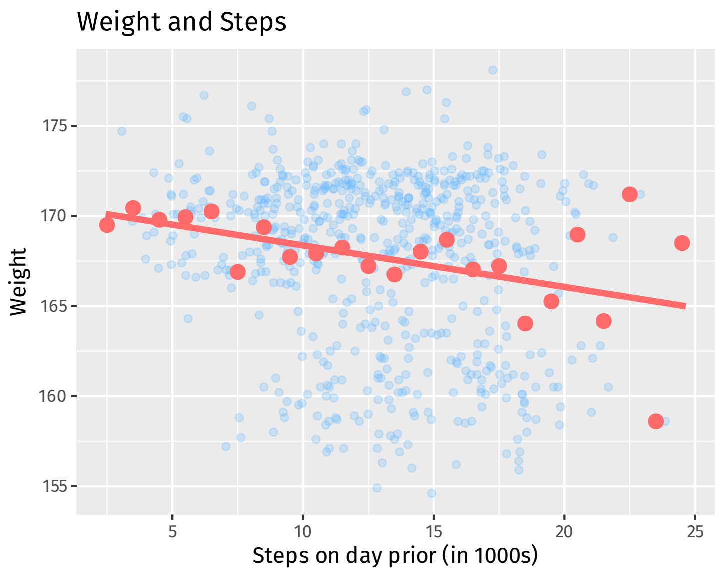
```r
augment(fit) |>
  ggplot(aes(x = steps_lag, y = .resid)) +
  geom_point(color = "steelblue1", alpha = 0.75) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Residuals",
    title = "Residual plot"
  ) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```


Residual plot
Residuals
10
5
0
-5
-10
5
10
15
20
25
Steps on day prior (in 1000s)

Smoothed graph of averages

Another way to think of the regression line is a smoothed version of the binned means plot:

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 3,
                   geom = "point", binwidth = 1) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)