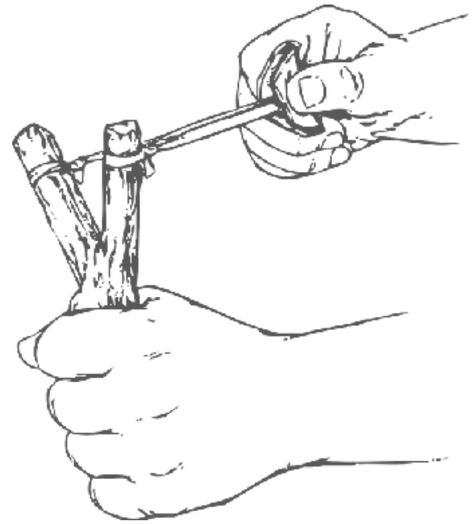
INTRODUCTION TO DATA ANALYSIS

LINEAR REGRESSION

PART I

OUTLINE

- fill me



case study

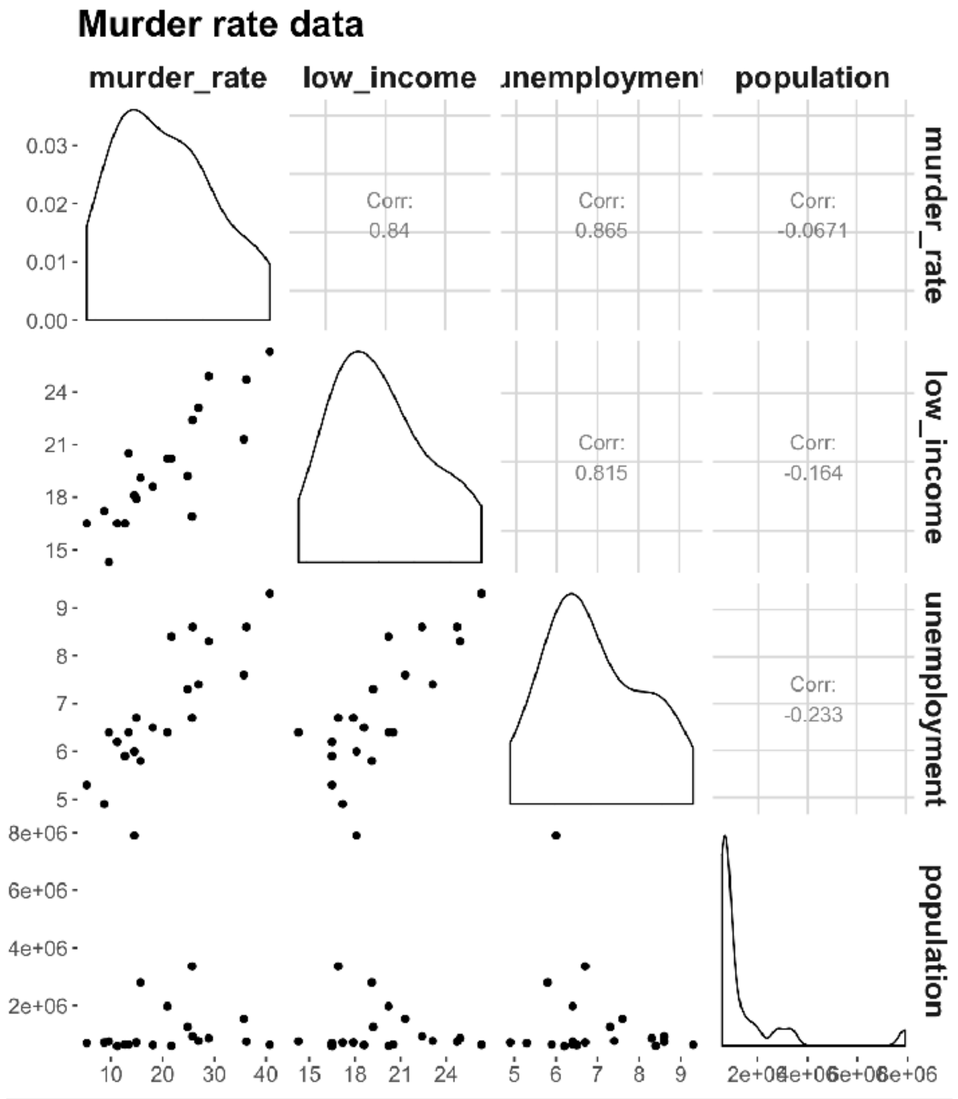
murder rates

MURDER RATES

```
murder_data
```

```
## # A tibble: 20 x 4
##    murder_rate low_income unemployment population
##          <dbl>      <dbl>        <dbl>      <dbl>
##  1        11.2       16.5          6.2     587000
##  2        13.4       20.5          6.4     643000
##  3        40.7       26.3          9.3     635000
##  4         5.3       16.5          5.3     692000
##  5        24.8       19.2          7.3    1248000
##  6        12.7       16.5          5.9     643000
##  7        20.9       20.2          6.4    1964000
##  8        35.7       21.3          7.6    1531000
##  9         8.7       17.2          4.9     713000
## 10         9.6       14.3          6.4     749000
## 11        14.5       18.1          6      7895000
## 12        26.9       23.1          7.4     762000
## 13        15.7       19.1          5.8    2793000
## 14        36.2       24.7          8.6     741000
## 15        18.1       18.6          6.5     625000
## 16        28.9       24.9          8.3     854000
## 17        14.9       17.9          6.7     716000
## 18        25.8       22.4          8.6     921000
## 19        21.7       20.2          8.4     595000
## 20        25.7       16.9          6.7    3353000
```

```
GGally::ggpairs(murder_data, title = "Murder rate data")
```



annual murders per million inhabitants

percentage inhabitants with low income

percentage inhabitants who are unemployed

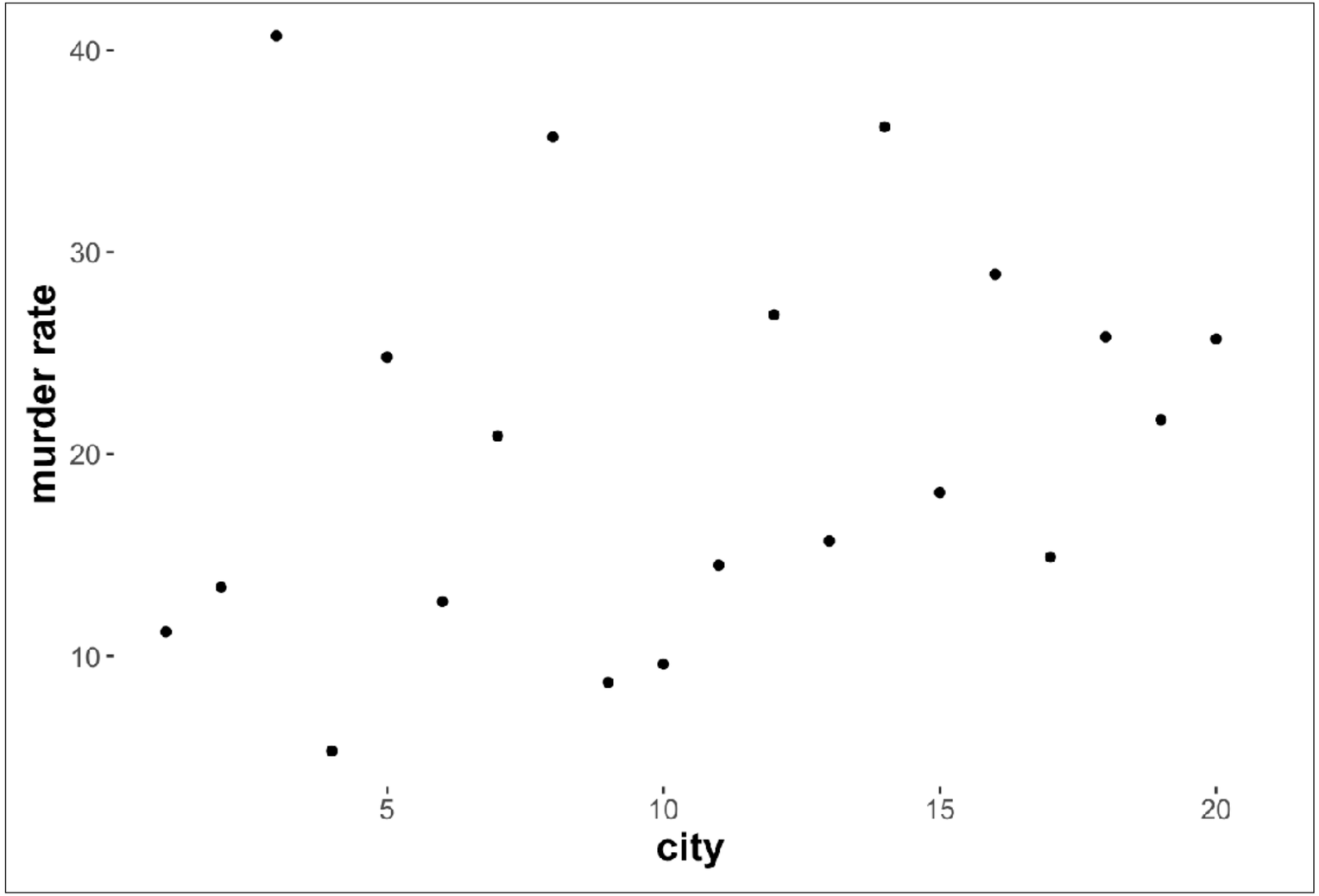
total population

linear regression

what? why? how?

PREDICTING MURDER RATES

MURDER RATE IN EACH CITY

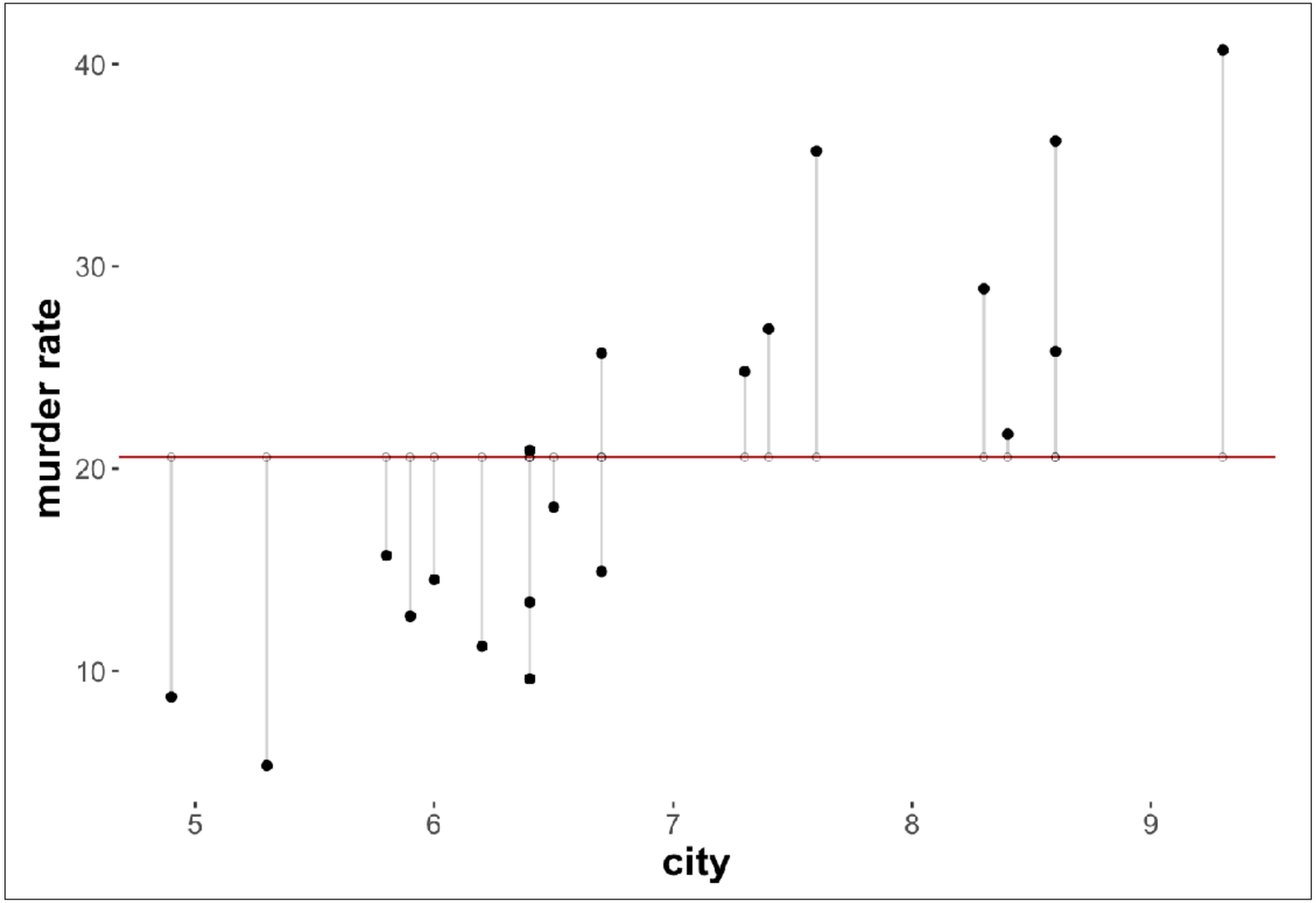


We know the murder rates of all cities. A city is randomly drawn from the data set (or the population from which the data is sampled). We are to predict the murder rate of that randomly drawn city. What's a best guess if we know nothing more about that city?

The mean of all murder rates!

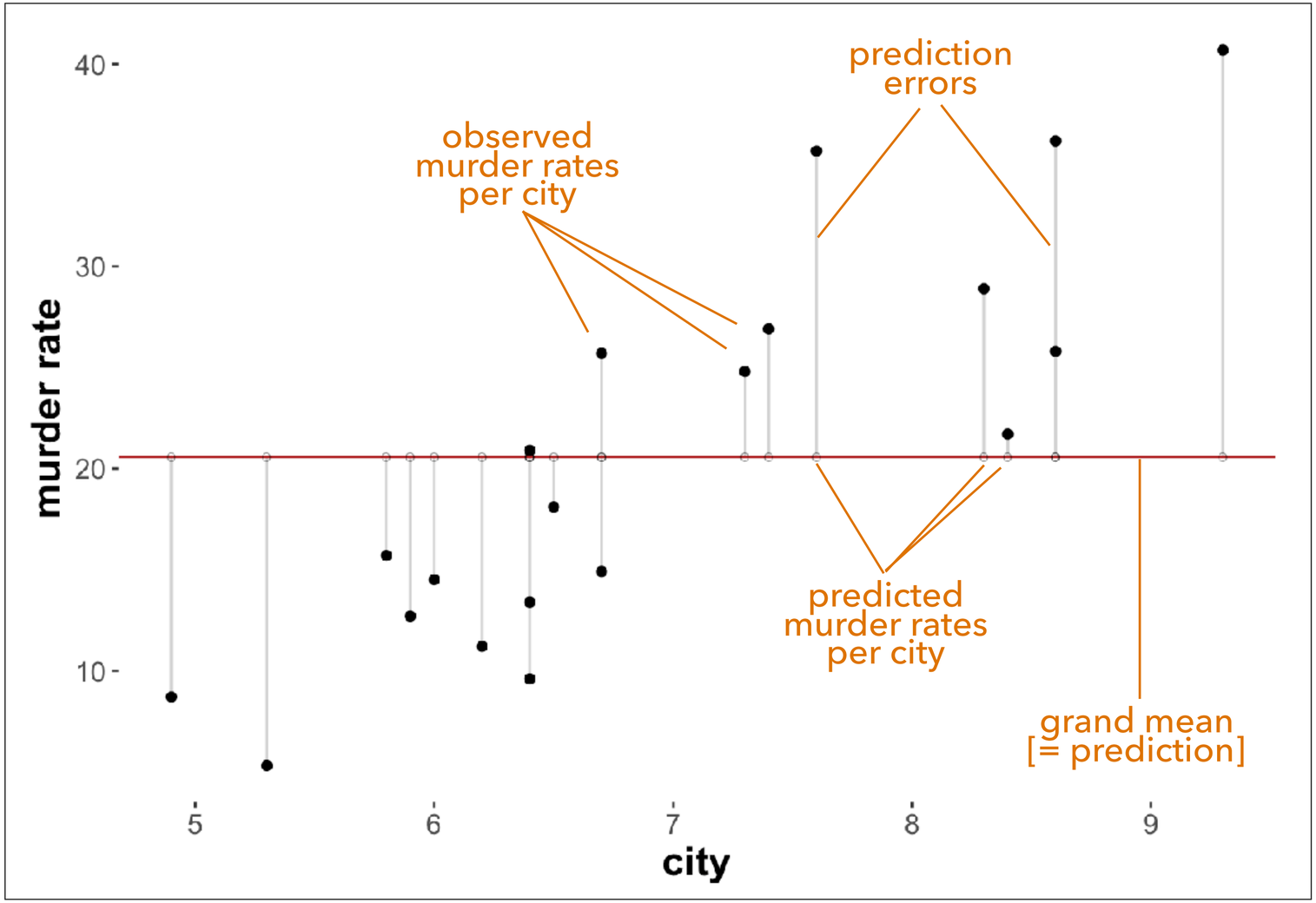
PREDICTING MURDER RATES

MURDER RATE IN EACH CITY WITH GRAND MEAN AS PREDICTOR



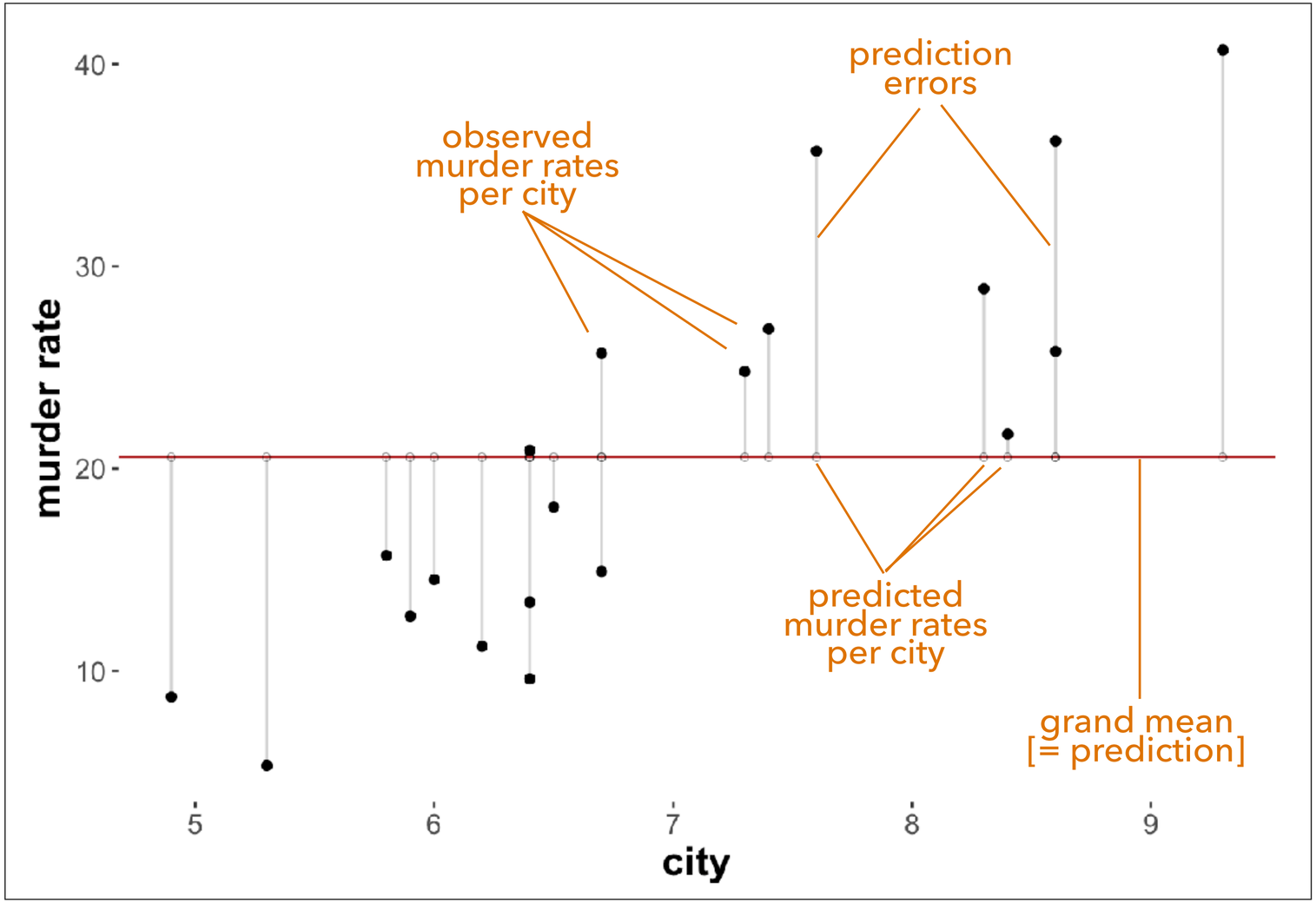
PREDICTING MURDER RATES

MURDER RATE IN EACH CITY WITH GRAND MEAN AS PREDICTOR



PREDICTING MURDER RATES

MURDER RATE IN EACH CITY WITH GRAND MEAN AS PREDICTOR



Overal prediction error:

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

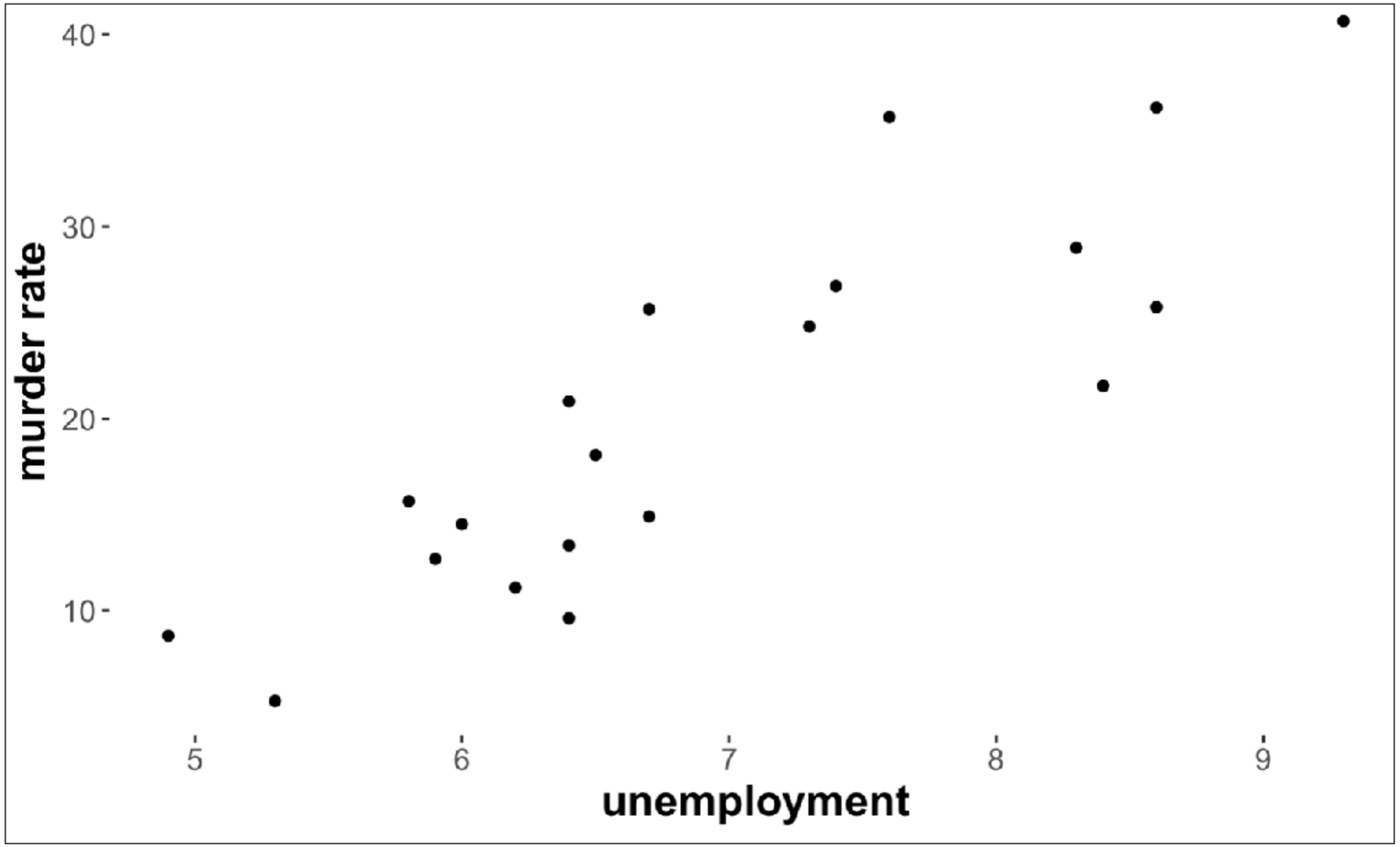
[total sum of squares]

```
y <- murder_data %>% pull(murder_rate)
n <- length(y)
tss_simple <- sum((y - mean(y))^2)
tss_simple
```

```
## [1] 1855.202
```

PREDICTING MURDER RATES

MURDER RATE IN EACH CITY AS A FUNCTION OF UNEMPLOYMENT RATE



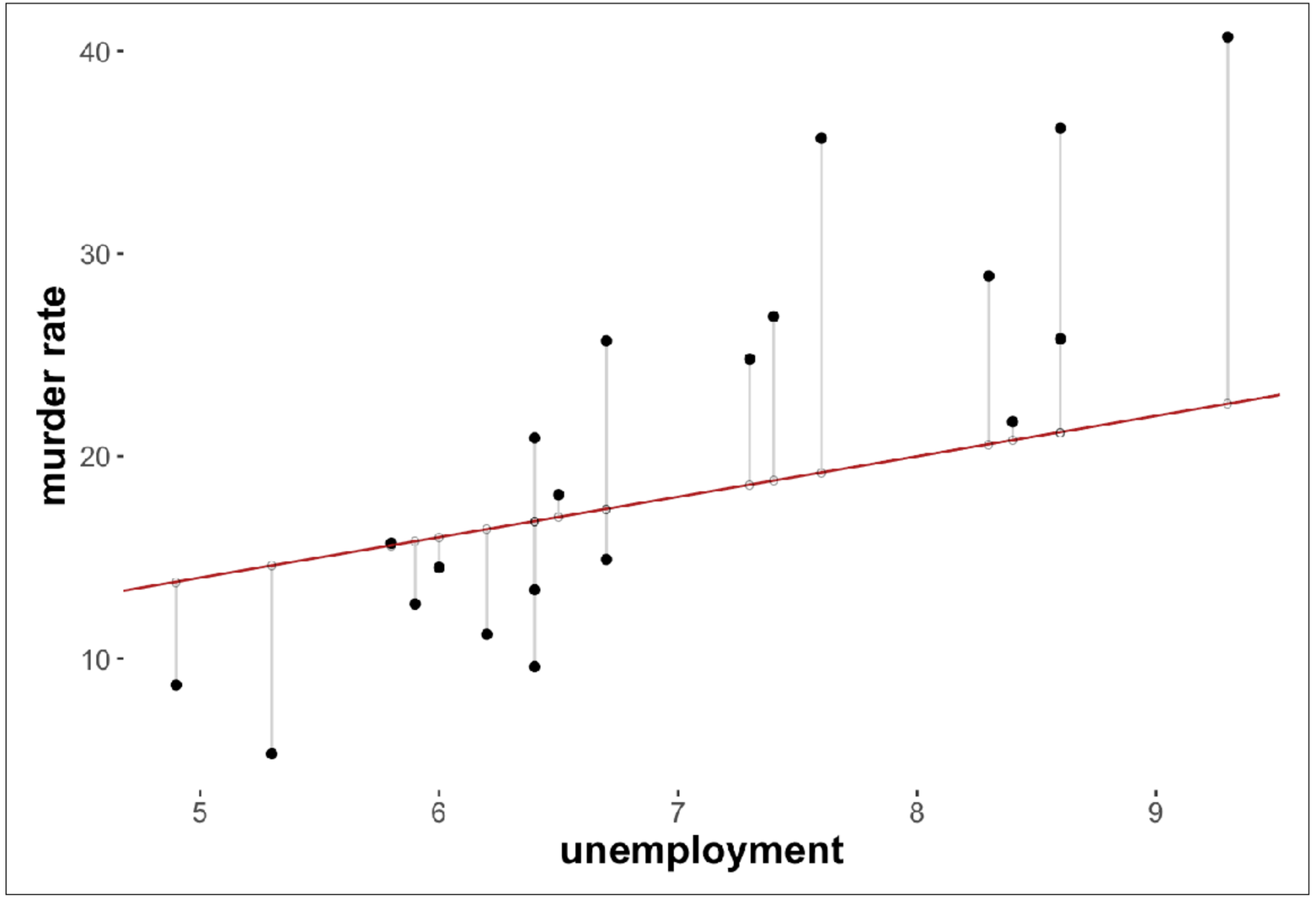
As before, we are to predict the murder rate of a randomly drawn city. But now **we also get to know that city's unemployment rate.** What's a best guess if we know nothing more about that city?

Let's just assume the following linear relationship:

$$\hat{y}_i = 2x_i + 4$$

PREDICTING MURDER RATES

MURDER RATE IN EACH CITY AS A FUNCTION OF UNEMPLOYMENT RATE



Overal prediction error:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y})^2$$

[residual sum of squares]

```
y <- murder_data %>% pull(murder_rate)
x <- murder_data %>% pull(unemployment)
predicted_y <- 2 * x + 4
n <- length(y)
rss_guesswork <- sum((y - predicted_y)^2)
rss_guesswork
```

```
## [1] 1327.74
```

SIMPLE LINEAR REGRESSION [ORDINARY LEAST-SQUARES REGRESSION]

linear prediction function

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

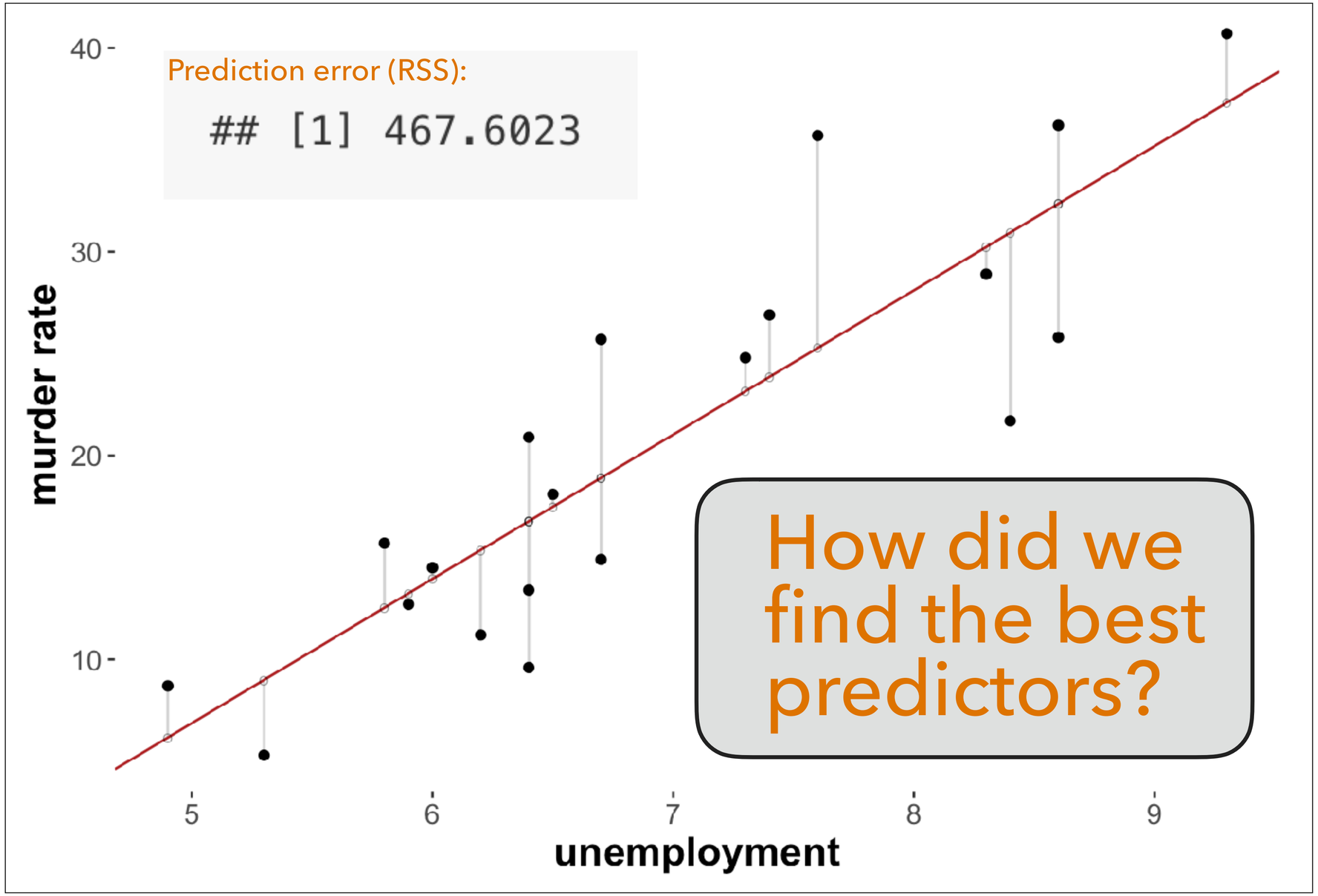
prediction error

$$\mathrm{RSS}_{\langle \beta_0, \beta_1 \rangle} = \sum_{i=1}^{k} (y_i - \hat{y}_i)^2$$

best-fitting parameters

$$\langle \hat{\beta}_0, \hat{\beta}_1 \rangle = \arg \min_{\langle \beta_0, \beta_1 \rangle} \mathrm{RSS}_{\langle \beta_0, \beta_1 \rangle}$$

BEST LINEAR PREDICTOR



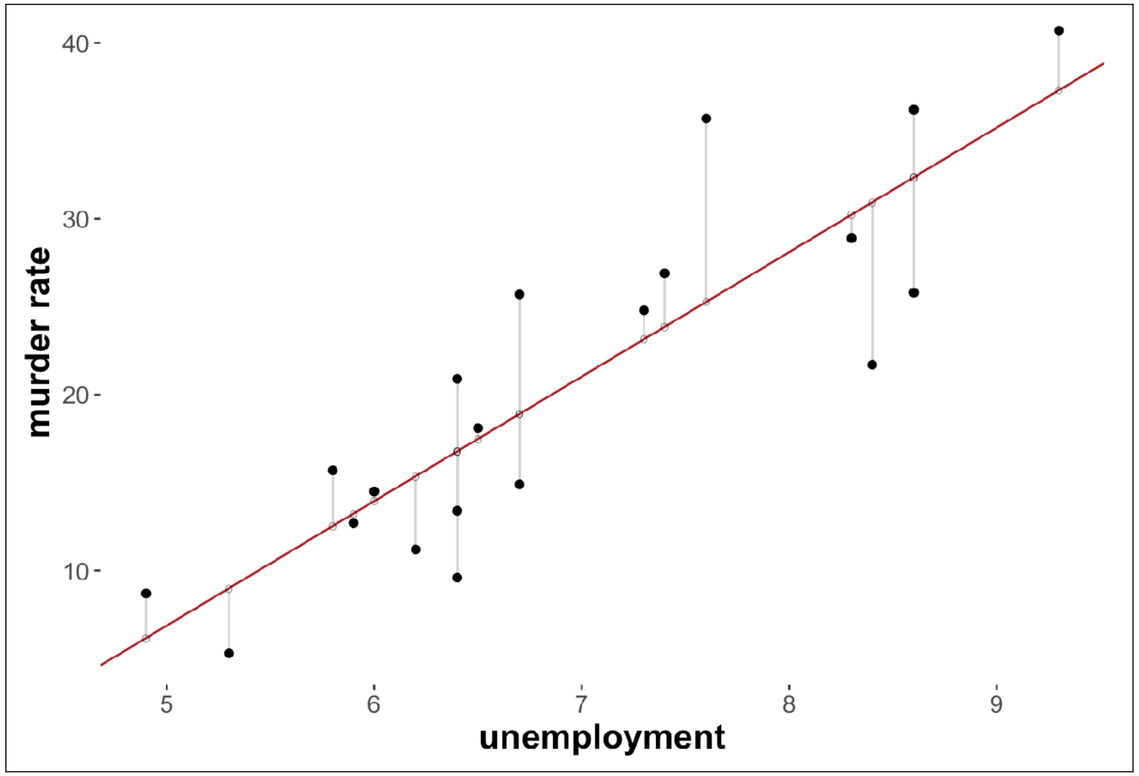
finding best-fits for

OLS regression

USING `OPTIM`

```
# data to be explained / predicted
y <- murder_data %>% pull(murder_rate)
# data to use for prediction / explanation
x <- murder_data %>% pull(unemployment)
# function to calculate residual sum of squares
get_rss = function(y, x, beta_0, beta_1) {
  yPred = beta_0 + x * beta_1
  sum((y-yPred)^2)
}
# finding best-fitting values for TSS
fit_rss = optim(par = c(0, 1),
  fn = function(par) {
    get_rss(y, x, par[1], par[2])
  }
)
# output the results
message(
  "Best fitting parameter values:",
  "\n\tIntercept: ", fit_rss$par[1] %>% signif(5),
  "\n\tSlope: ", fit_rss$par[2] %>%  signif(5),
  "\nRSS for best fit: ", fit_rss$value %>% signif(5)
)
```

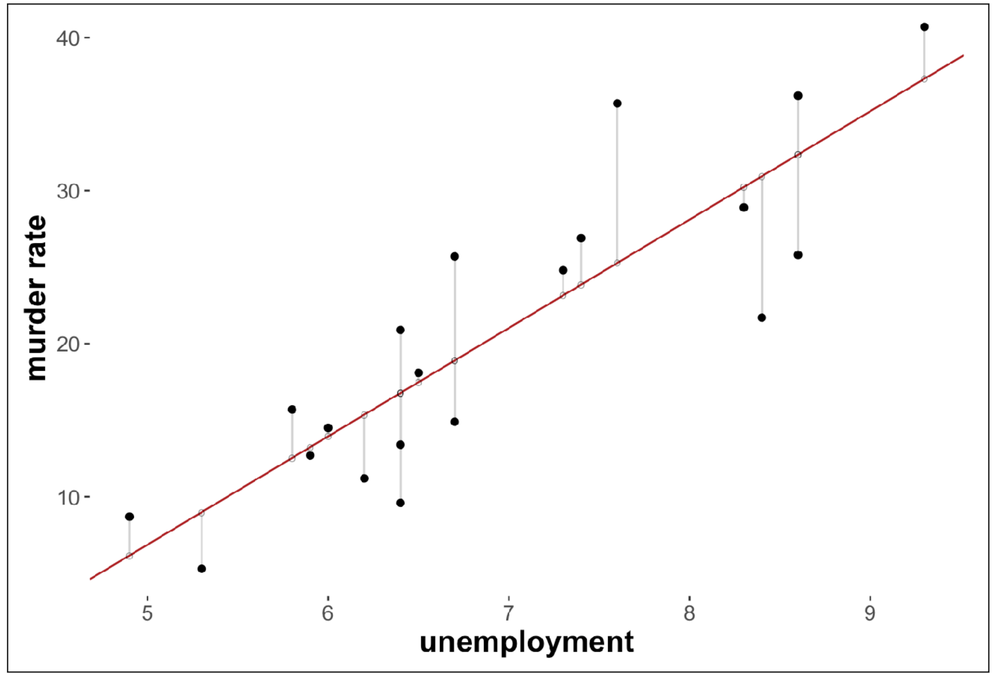
```
## Best fitting parameter values:
##  Intercept: -28.528
##  Slope: 7.0795
## RSS for best fit: 467.6
```



USING `LM`

```r
# fit an OLS regression
fit_lm <- lm(
  # the formula argument specifies dependent and independent variables
  formula = murder_rate ~ unemployment,
  # we also need to say where the data (columns) should come from
  data = murder_data
)
# output the fitted object
fit_lm
```

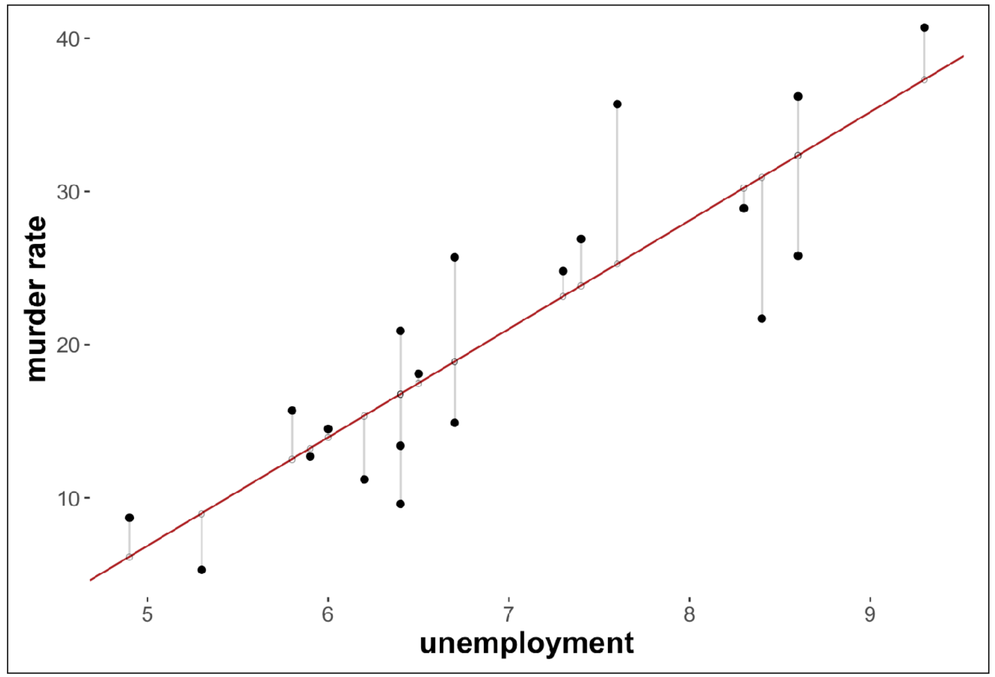
```
##
## Call:
## lm(formula = murder_rate ~ unemployment, data = murder_data)
##
## Coefficients:
##  (Intercept)  unemployment
##       -28.53          7.08
```



USING `LM`

```
summary(fit_lm)
```

```
##
## Call:
## lm(formula = murder_rate ~ unemployment, data = murder_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.2415 -3.7728  0.5795  3.2207 10.4221
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.5267     6.8137  -4.187 0.000554 ***
## unemployment   7.0796     0.9687   7.309 8.66e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.097 on 18 degrees of freedom
## Multiple R-squared:  0.748,  Adjusted R-squared:  0.7339
## F-statistic: 53.41 on 1 and 18 DF,  p-value: 8.663e-07
```



USING MATH

Theorem 3.1 (OLS solution) *For a simple linear regression model with just one predictor, the solution for:*

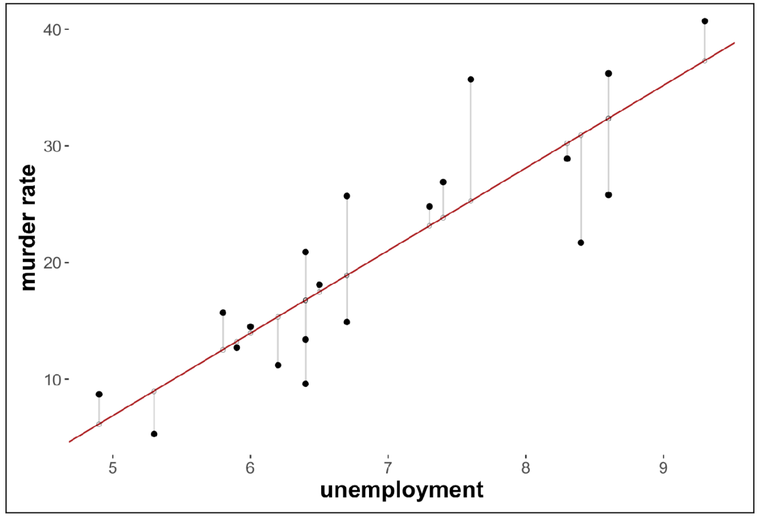
$$\arg\min_{\langle\beta_0,\beta_1\rangle} \sum_{i=1}^{k} (y_i - (\beta_0 + \beta_1 x_i))^2$$

is given by:

$$\hat{\beta}_1 = \frac{Cov(x,y)}{Var(x)} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

```
tibble(
  beta_1 = cov(x,y) / var(x),
  beta_0 = mean(y) - beta_1 * mean(x)
)
```

```
## # A tibble: 1 x 2
##   beta_1 beta_0
##    <dbl>  <dbl>
## 1   7.08  -28.5
```



finding best-fits for a

likelihood-based approach

FREQUENTIST MODEL [LIKELIHOOD-BASED APPROACH]

likelihood-based regression

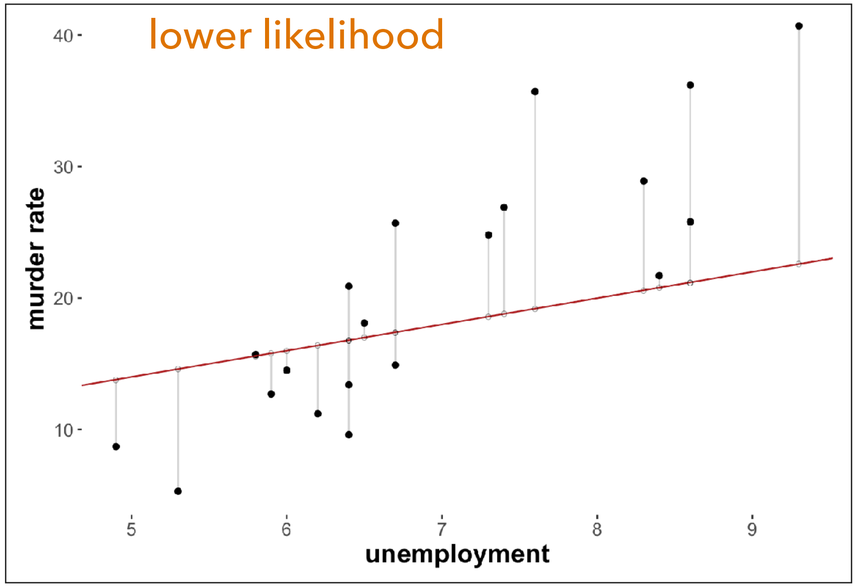
[explicit version]

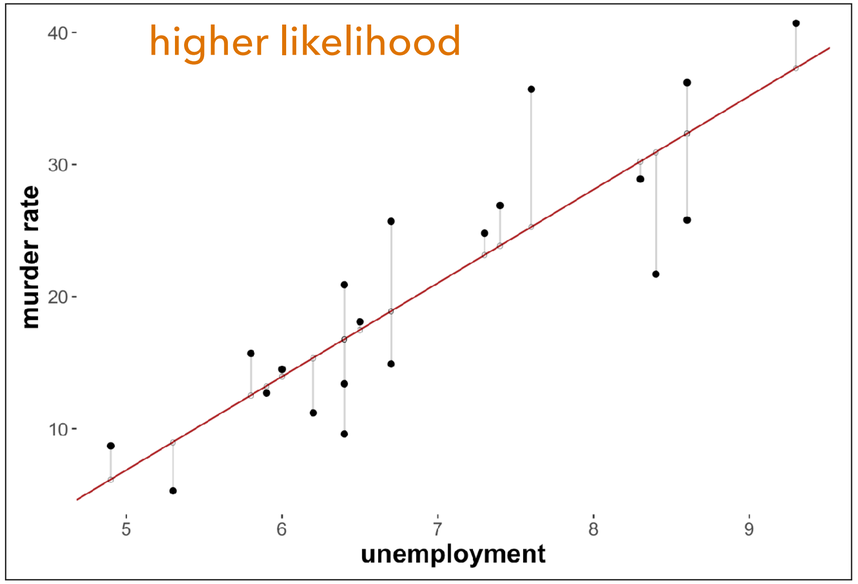
$$y_{\text{pred}} = \beta_0 + \beta_1 x$$
$$y_i = y_{\text{pred}} + \epsilon_i$$
$$\epsilon_i \sim \text{Normal}(0, \sigma)$$

likelihood-based regression

[compact version]

$$y_{\text{pred}} = \beta_0 + \beta_1 x$$
$$y \sim \text{Normal}(\mu = y_{\text{pred}}, \sigma)$$

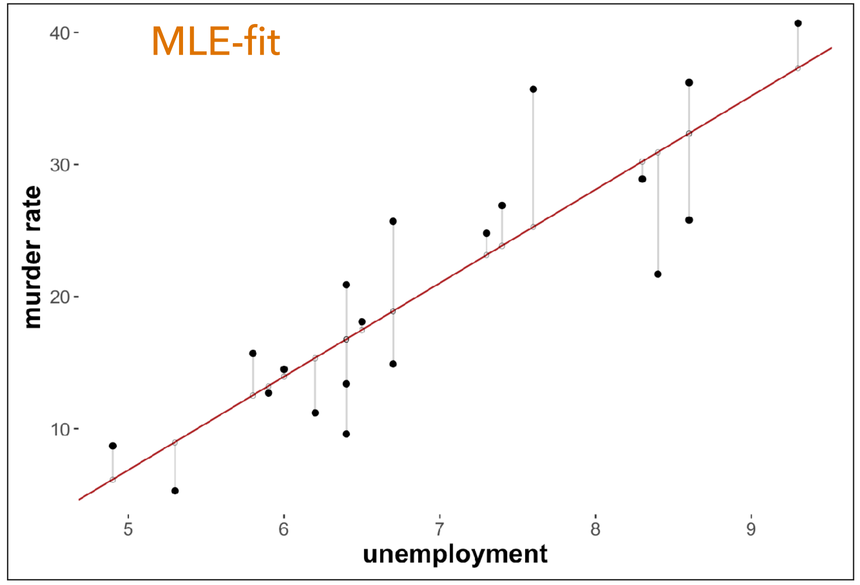




FITTING WITH `OPTIM`

```r
# data to be explained / predicted
y <- murder_data %>% pull(murder_rate)
# data to use for prediction / explanation
x <- murder_data %>% pull(unemployment)
# function to calculate negative log-likelihood
get_nll = function(y, x, beta_0, beta_1, sd) {
  if (sd <= 0) {return( Inf )}
  yPred = beta_0 + x * beta_1
  nll = -dnorm(y, mean=yPred, sd=sd, log = T)
  sum(nll)
}
# finding MLE
fit_lh = optim(par = c(0, 1, 1),
  fn = function(par) {
    get_nll(y, x, par[1], par[2], par[3])
  }
)
# output the results
message(
  "Best fitting parameter values:",
  "\n\tIntercept: ", fit_lh$par[1] %>% signif(5),
  "\n\tSlope: ", fit_lh$par[2] %>%  signif(5),
  "\nNegative log-likelihood for best fit: ", fit_lh$value %>% signif(5)
)
```

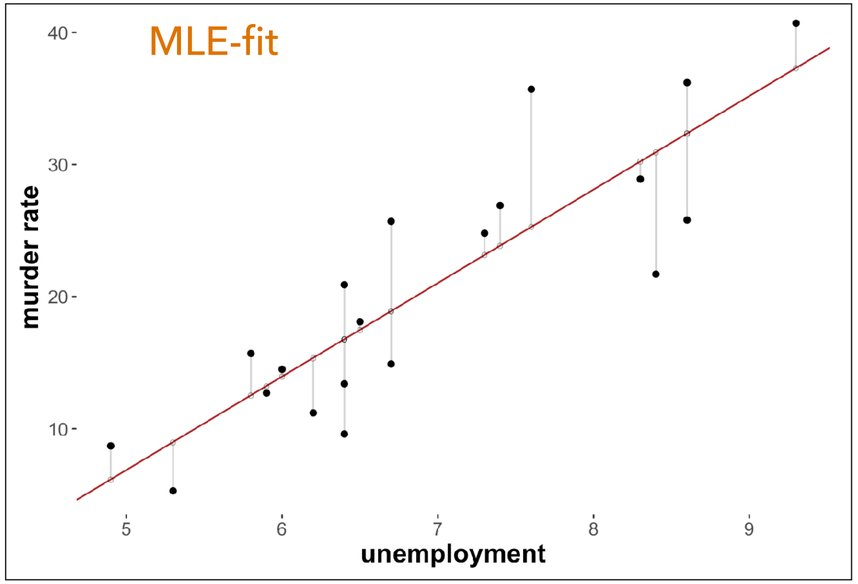
```
Best fitting parameter values:
        Intercept: -28.517
        Slope: 7.0783
Negative log-likelihood for best fit: 59.898
```



FITTING WITH `GLM`

```r
fit_glm <- glm(murder_rate ~ unemployment, data = murder_data)
fit_glm
```

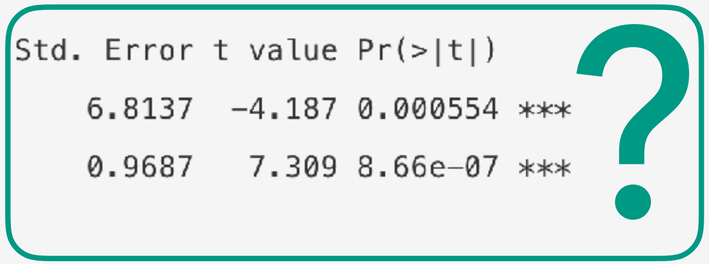
```
##
## Call:  glm(formula = murder_rate ~ unemployment, data = murder_data)
##
## Coefficients:
##  (Intercept)  unemployment
##       -28.53          7.08
##
## Degrees of Freedom: 19 Total (i.e. Null);  18 Residual
## Null Deviance:       1855
## Residual Deviance: 467.6     AIC: 125.8
```



FITTING WITH `GLM`

```
summary(fit_glm)
```

```
##
## Call:
## glm(formula = murder_rate ~ unemployment, data = murder_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.2415  -3.7728   0.5795   3.2207  10.4221
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.5267     6.8137  -4.187 0.000554 ***
## unemployment   7.0796     0.9687   7.309 8.66e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



CONNECTION OLS & MLE

Theorem 3.2 (MLE solution) *For a simple linear regression model with just one predictor, the solution for:*

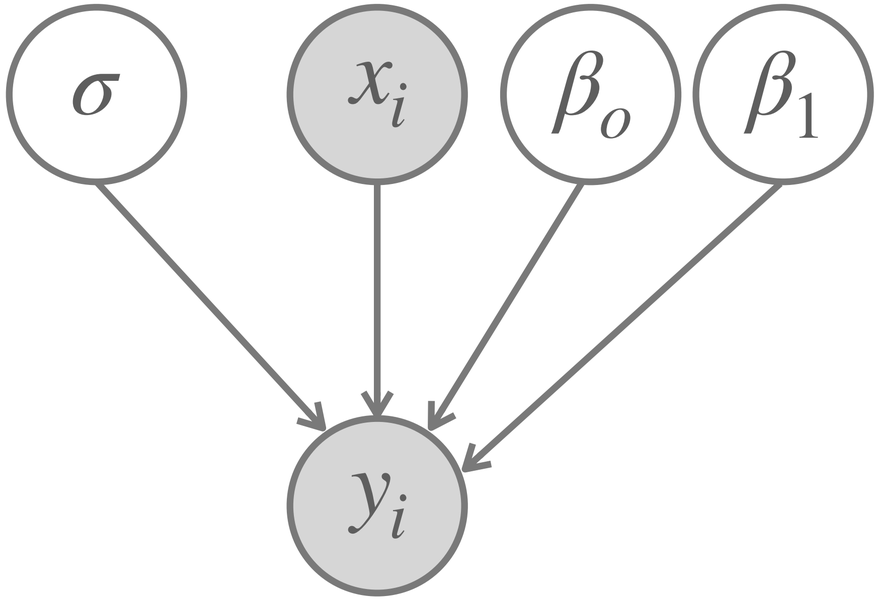
$$\arg \max_{\langle \beta_0, \beta_1, \sigma \rangle} \prod_{i=1}^{k} \text{Normal}(\mu = \beta_0 + \beta_1 x_i, \sigma)$$

is the same as for the OLS approach:

$$\hat{\beta}_1 = \frac{Cov(x, y)}{Var(x)} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Bayesian approach

BAYESIAN SIMPLE LINEAR REGRESSION MODEL



$$\sigma \sim \text{Trunc-Norm}(\ldots)$$

$$\beta_0 \sim \text{Student-t}(\ldots)$$

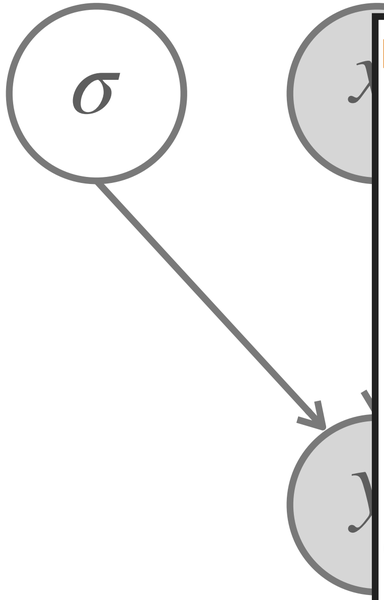
$$\beta_1 \sim \text{Student-t}(\ldots)$$

$$y_i \sim \text{Normal}(\beta_0 + \beta_1 x_i, \sigma)$$

greta

```
# data to be explained / predicted
y <- murder_data %>% pull(murder_rate)
# data to use for prediction / explanation
x <- murder_data %>% pull(unemployment)
y_greta     <- as_data(y)
x_greta     <- as_data(x)
# latent variables and priors
intercept <- student(df= 1, mu = 0, sigma = 10)
slope     <- student(df= 1, mu = 0, sigma = 10)
sigma     <- normal(0, 5, truncation = c(0, Inf))
# derived latent variable (linear model)
y_pred <- intercept + slope * x_greta
# likelihood
distribution(y) <- normal(y_pred, sigma)
# finalize model, register which parameters to monitor
murder_model <- model(intercept, slope, sigma)
```

BAYESIAN SIMPLE LINEAR REGRESSION MODEL



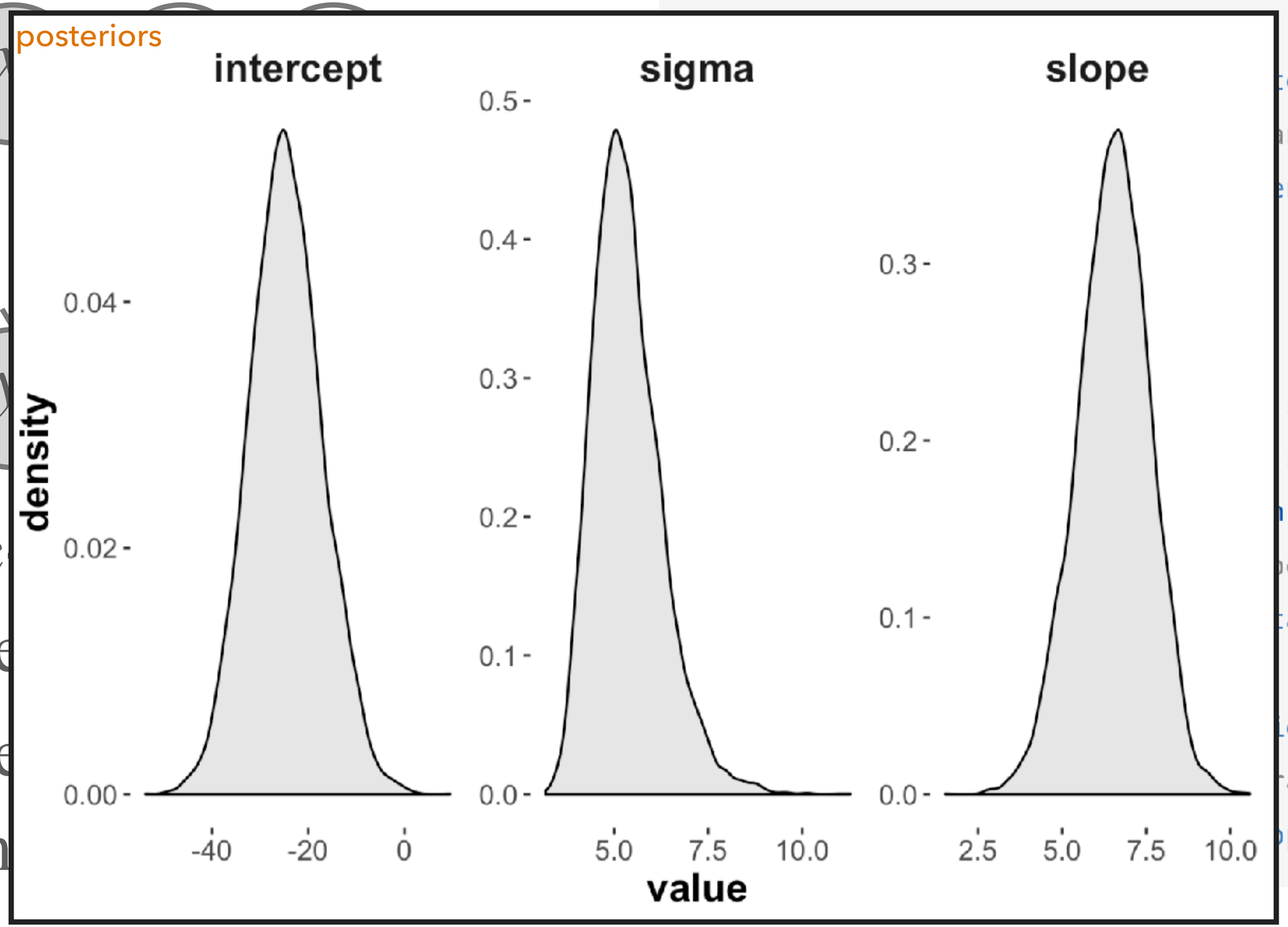
$\sigma \sim \text{Trunc}$

$\beta_0 \sim \text{Stude}$

$\beta_1 \sim \text{Stude}$

$y_i \sim \text{Norm}$

posteriors



greta

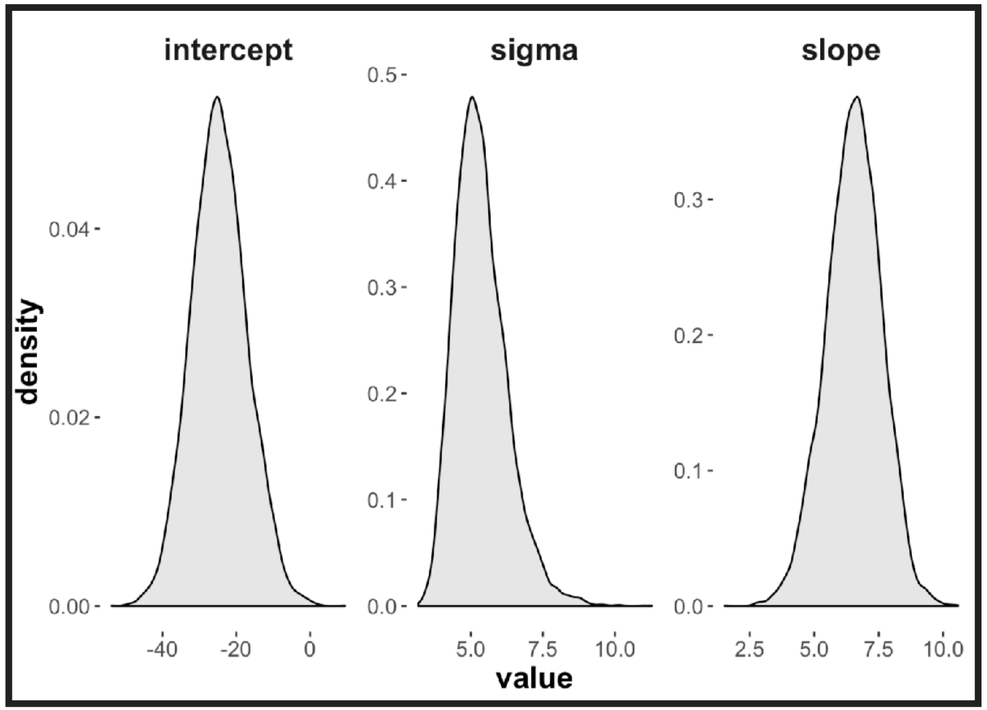
```
te)
nation
ent)


 sigma = 10)
 sigma = 10)
 = c(0, Inf))
odel)
ta


igma)
rameters to monitor
pe, sigma)
```

Hypothesis tests for regression coefficients

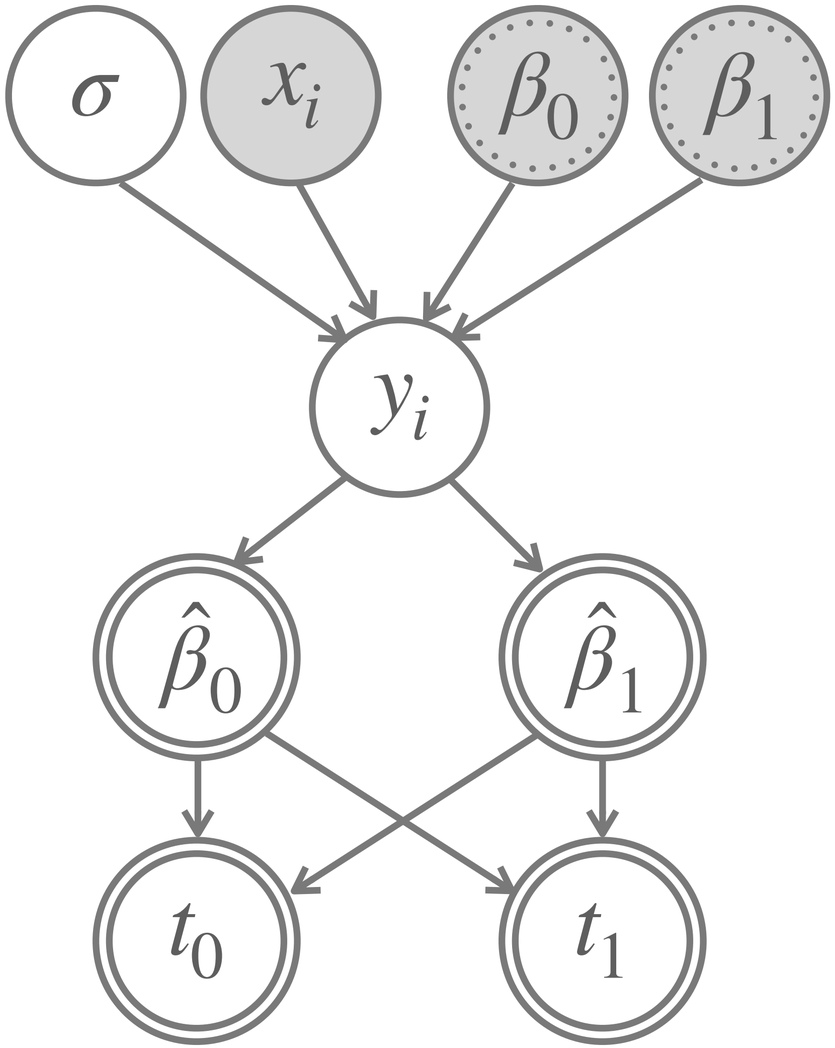
BAYESIAN APPROACH [ESTIMATION-BASED]

```r
# get means and 95% HDI
Bayes_estimates <- tidy_draws_murder_data %>%
  group_by(Parameter) %>%
  summarise(
    mean = mean(value),
    '|95%' = HDInterval::hdi(value)[1],
    '95|%' = HDInterval::hdi(value)[2]
  )
Bayes_estimates
```

```
## # A tibble: 3 x 4
##   Parameter   mean `|95%` `95|%`
##   <fct>      <dbl>  <dbl>  <dbl>
## 1 intercept -24.6  -39.1   -9.51
## 2 sigma       5.36   3.78   7.20
## 3 slope       6.53   4.34   8.53
```



FREQUENTIST T-TESTS FOR REGRESSION COEFFICIENTS



$$\hat{\beta}_1 = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(y)} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Null-hypothesis

$$\beta_0 = \beta_1 = 0$$

$$\mathrm{SE}_{\beta_1} = \sqrt{\frac{\sum_i (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2}{(N-2) \cdot \sum_i (x_i - \bar{x})^2}}$$

$$\mathrm{SE}_{\beta_0} = \sqrt{\sum_i (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 \cdot \left(\frac{1}{(N-2)} + \frac{\bar{x}^2}{\sum_i (x_i - \bar{x})^2} \right)}$$

$$t_1 = \frac{\hat{\beta}_1}{\mathrm{SE}_{\beta_1}} \qquad t_0 = \frac{\hat{\beta}_0}{\mathrm{SE}_{\beta_0}}$$

Sampling distribution:

$$t_0, t_1 \sim \text{Student-t}(\nu = N - 2)$$

FREQUENTIST T-TESTS FOR REGRESSION COEFFICIENTS

```r
# observed data
y_obs <- murder_data %>% pull(murder_rate)
x_obs <- murder_data %>% pull(unemployment)
n_obs <- length(y_obs)
# best-fitting coefficients
beta_1_hat <- cov(x_obs, y_obs) / var(x_obs)
beta_0_hat <- mean(y_obs) - beta_1_hat * mean(x_obs)
# calculating t-scores
MSE <- sum((y_obs - beta_0_hat - beta_1_hat * x_obs)^2) / (n_obs-2)
S_xx <- sum((x_obs - mean(x_obs))^2)
SE_beta_1_hat <- sqrt(MSE  / S_xx)
SE_beta_0_hat <- sqrt(MSE * (1/n_obs + mean(x_obs)^2 / S_xx))
t_slope = (beta_1_hat) / SE_beta_1_hat
t_intercept = beta_0_hat / SE_beta_0_hat
tibble(t_slope, t_intercept)
```

```
## # A tibble: 1 x 2
##   t_slope t_intercept
##     <dbl>       <dbl>
## 1    7.31       -4.19
```

```r
p_value_intercept = pt(t_intercept, df = n_obs -2) + 1-pt(-t_intercept, df = n_obs -2)
p_value_slope     = pt(-t_slope, df = n_obs -2) + 1-pt(t_slope, df = n_obs -2)
tibble(p_value_intercept, p_value_slope)
```

```
## # A tibble: 1 x 2
##   p_value_intercept p_value_slope
##               <dbl>         <dbl>
## 1          0.000554   0.000000866
```

```r
summary(glm(murder_rate ~ unemployment, data = murder_data))
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.5267     6.8137  -4.187 0.000554 ***
## unemployment   7.0796     0.9687   7.309 8.66e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```