Quiz7-4

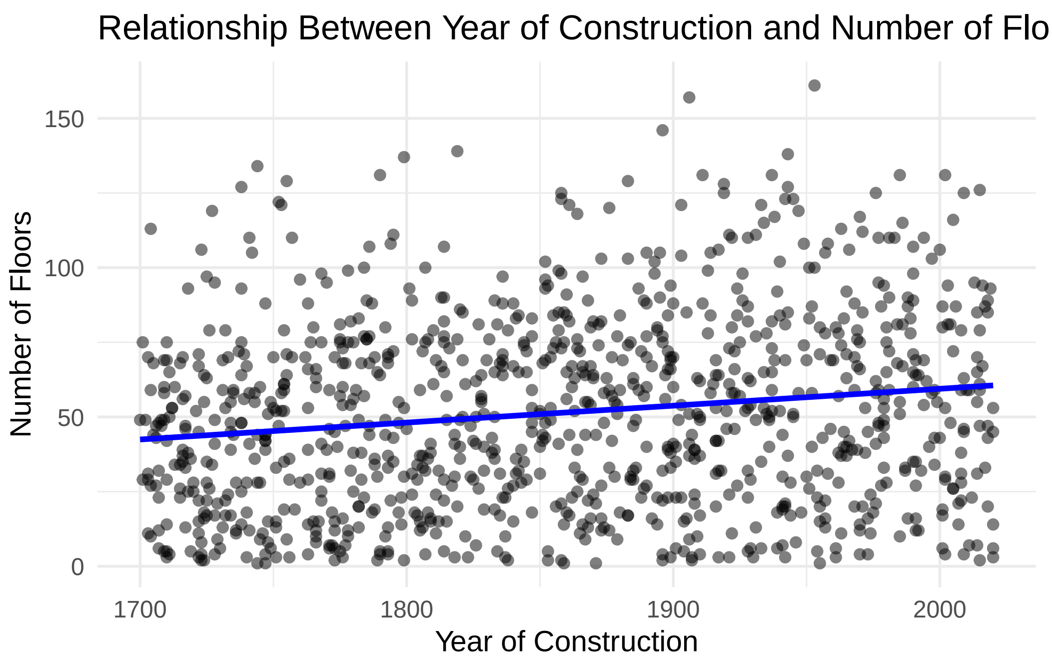
```r
library(ggplot2)
set.seed(123)
n <- 1000

year_of_construction <- sample(1700:2020, n, replace = TRUE)
location_zone <- sample(c("Central", "Suburban", "Outskirts"), n, replace = TRUE, prob = c(0
building_type <- sample(c("Residential", "Commercial", "Mixed-use"), n, replace = TRUE)

number_of_floors <- round(runif(n, min = 1, max = 100) *
                            (year_of_construction / 2020)^2 *
                            (ifelse(location_zone == "Central", 1.5, 1)) *
                            (ifelse(building_type == "Commercial", 1.2, 1))
                          )


buildings_df <- data.frame(year_of_construction, location_zone, building_type, number_of_flo


head(buildings_df)
```

```
  year_of_construction location_zone building_type number_of_floors
1                 1878      Suburban     Mixed-use               30
2                 1713       Central    Commercial               34
3                 1894      Suburban   Residential               23
4                 2005       Central     Mixed-use              116
5                 1817       Central    Commercial               27
6                 1998      Suburban   Residential               59
```

```r
ggplot(buildings_df, aes(x = year_of_construction, y = number_of_floors)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
```

```r
  labs(title = "Relationship Between Year of Construction and Number of Floors",
       x = "Year of Construction",
       y = "Number of Floors") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```r
if (!require("rstanarm")) install.packages("rstanarm")
```

```
Loading required package: rstanarm

Loading required package: Rcpp

This is rstanarm version 2.32.1

- See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

- Default priors may change, so it's safest to specify priors, even if equivalent to the def

- For execution on a local, multicore CPU with excess RAM we recommend calling
```

```
options(mc.cores = parallel::detectCores())
```

```r
library(rstanarm)


model <- stan_glm(number_of_floors ~ year_of_construction,
                  data = buildings_df,
                  family = gaussian, # Because we are predicting a continuous outcome
                  prior = normal(0, 2.5), # Assuming a normal prior with mean 0 and SD 2.5
                  seed = 123 # Set a seed for reproducibility
)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 2.7e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.036 seconds (Warm-up)
Chain 1:                0.095 seconds (Sampling)
Chain 1:                0.131 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 9e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
```

```
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.032 seconds (Warm-up)
Chain 2:                0.096 seconds (Sampling)
Chain 2:                0.128 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.07 seconds (Warm-up)
Chain 3:                0.093 seconds (Sampling)
```

```
Chain 3:                0.163 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4: 
Chain 4: 
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 0.039 seconds (Warm-up)
Chain 4:                0.09 seconds (Sampling)
Chain 4:                0.129 seconds (Total)
Chain 4: 
```

```r
# View model summary
print(summary(model))
```

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      number_of_floors ~ year_of_construction
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 1000
 predictors:   2
```

```
Estimates:
                       mean   sd    10%   50%   90%
(Intercept)           -53.8   20.1 -79.5 -54.0 -28.1
year_of_construction   0.1    0.0   0.0   0.1   0.1
sigma                  31.5    0.7  30.6  31.5  32.4

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 51.4    1.4 49.6  51.4  53.2

The mean_ppd is the sample average posterior predictive distribution of the outcome variable

MCMC diagnostics
                     mcse Rhat n_eff
(Intercept)          0.3  1.0  4285
year_of_construction 0.0  1.0  4283
sigma                0.0  1.0  3351
mean_PPD             0.0  1.0  3043
log-posterior        0.0  1.0  2120

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective
```

```r
# Visualize model results
plot(model)
```