Bayes Assignment 3 of 2025

Sean van der Merwe

2025-02-26

Activate the navigation pane (View menu) for ease of navigation through the document.

Instructions

The dataset is the 100 AI Companies of 2024 dataset from Kaggle. It is provided on the learning management system for convenience. The last 3 columns are the most interesting and what you will focus on.

Note that the data is raw, and slightly corrupted. You must first clean up the data and transform the variables. All such steps must be done using R code. You may not make any alterations to the data set. Your code must run correctly if someone downloads the data from the source again. For example, there are scores that Excel converted to dates at some point, transform them back intelligently with code.

Goal: To implement a robust Bayesian regression model on a dataset to test for basic trends and make a prediction with uncertainty.

After data cleaning, drop the two companies with missing scores. Then drop the company that corresponds to your position on the class list, leaving 97 companies.

Explain, based on statistics, whether you think the variables are related (before or after transformations). For the company that corresponds to your position on the class list, predict their revenue **distribution** for 2025 (one year older) assuming that their Glassdoor score drops by 0.5/5.

Your marks will be based how well you explain your approach and how sensible your reasoning is (all your steps, and especially your predicted distribution, must make sense given the constraints of the data).

Also, consider using transformed variables in your regressions, such as log annual revenue and log age, instead of the raw values.

For this assignment, submit Word, PDF, and Rmd/qmd files in one submission on the learning management system, in that order.

Memorandum

First the student must identify their position on the class list.

```
st <- 20

options(scipen = 12)
library(knitr)
library(tidyverse)
library(devEMF)
opts_chunk$set(dev='emf', fig.ext='emf')
```

Data cleaning

The data is read in unaltered.

```
d <- read.csv('Ai_companies.csv')
```

The key variables are transformed.

```
d$Glassdoor.Score[d$Glassdoor.Score == "5-Apr"] <- "4.0/5"
d$Score <- d$Glassdoor.Score |> substr(1, 3) |> as.numeric()
get_revenue <- \(r) {
  revenue_split <- r |> str_split_fixed("\\s", 2)
  revenue_raw <- revenue_split[,1] |> parse_number()
  ifelse(revenue_split[,2] |> startsWith("m"), revenue_raw*1000000,
revenue_raw*1000000000)
}
d$Revenue <- get_revenue(d$Annual.Revenue)
d$Log_Revenue <- d$Revenue |> log()
d$Age <- 2025 - d$Founded
d$Log_Age <- d$Age |> log()
```

Problem rows are removed. Target row is separated and adjusted as requested.

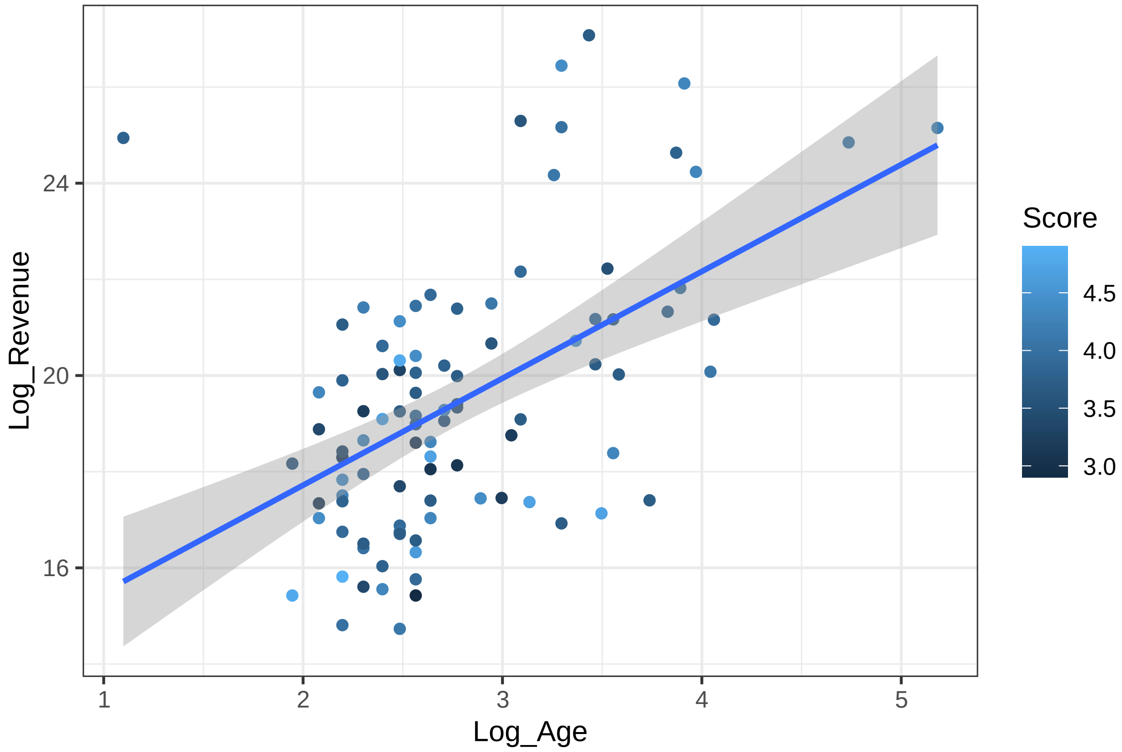
```
d <- d |> subset(!is.na(Score))
d_target <- d[st, ]
d <- d[-st, ]

d_target <- d_target |> mutate(
  Age = Age + 1,
  Log_Age = log(Age),
  Score = Score - 0.5
)
```

Data exploration

The data is explored visually to check for further anomalies and intricacies.

```
d |> ggplot(aes(x = Log_Age, y = Log_Revenue)) +
  geom_point(aes(colour = Score)) + theme_bw() +
  geom_smooth(method = 'lm', formula = 'y~x')
```



Let us highlight the youngest and oldest companies as they might be influential observations.

```
rbind(d[which.min(d$Log_Age),], d[which.max(d$Log_Age),]) |>
  select(1:6) |> kable()
```

	Company.Name	Description	Headquarters	Founded	Annual.Revenue	Glassdoor.Score
61	GE Vernova	Best for Wind Turbine Model	Cambridge, Massachusets	2022	$68 billion	3.8/5
62	Siemens	Best for Industrial Automation and Digitalization	Munich, Germany	1847	$83.65 billion	4.2/5

Correlation

It is interesting to consider the correlations between the variables before doing regression, at least in a data science setting where prediction is the focus. Note that deciding on the model based on the correlations can result in spurious (false positive) results and should be avoided unless you have already split the data and are only exploring the training portion.

Never attempt to test a model using the same data that you use to build the model.

Calculating correlations in R is straightforward.

```
d |> select(Log_Revenue, Log_Age, Score) |> cor()
```

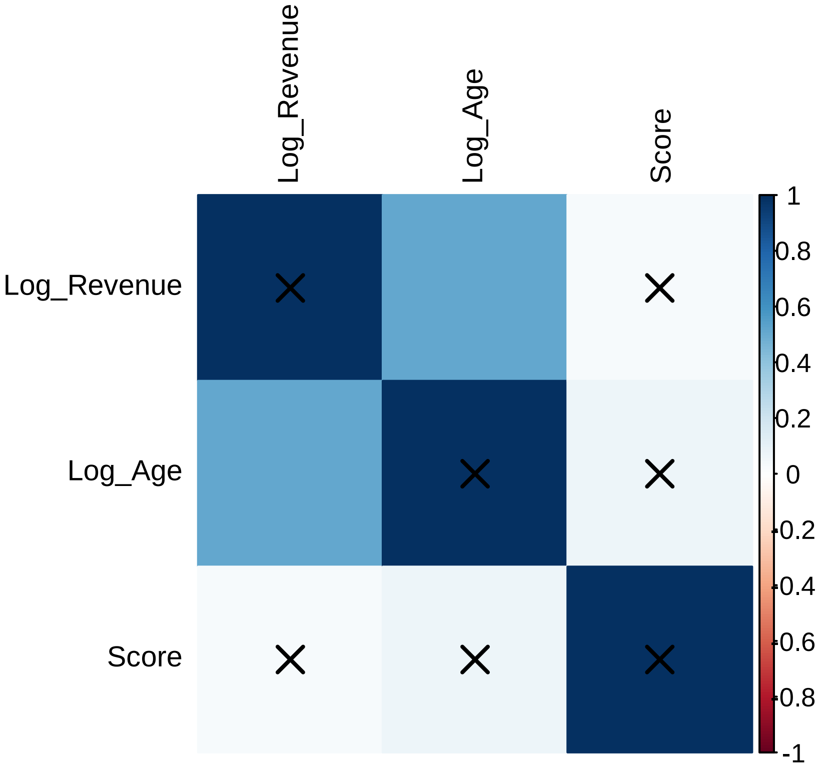
```
            Log_Revenue    Log_Age      Score
Log_Revenue  1.00000000 0.51678451 0.03526887
Log_Age      0.51678451 1.00000000 0.07942302
Score        0.03526887 0.07942302 1.00000000
```

But testing the correlations properly requires additional calculations, and illustrating them neatly requires a package such as *corrplot*.

```
pvalfunc <- function(sims, target = 0) { 2*min(mean(sims < target), mean(sims >
target)) }
corrplot_exact <- function(num_data_matrix, crosssize = 1.8, textsize = 0.9) {
  rho_post_sim <- function(r, n, n_sims = 10000) {
    y <- r*sqrt(rchisq(n_sims, n-2)/rchisq(n_sims, n-1)/(1-(r^2))) -
      rnorm(n_sims)/sqrt(rchisq(n_sims, n-1))
    y/sqrt(y^2 + 1)
  }
  if (any(class(num_data_matrix) %in% "data.frame")) {
    num_data_matrix <- as.matrix(num_data_matrix)
  }
  cormat <- cor(num_data_matrix, use="pairwise.complete.obs")
  rownames(cormat) <- colnames(cormat) <- colnames(num_data_matrix)
  nc <- ncol(num_data_matrix)
  p_values <- matrix(0, nc, nc)
  rownames(p_values) <- colnames(p_values) <- colnames(cormat)
  seq_len(nc-1) |> sapply(\(i) {
    seq((i+1), nc) |> sapply(\(j) {
      n <- sum(!(is.na(num_data_matrix[,i]) | is.na(num_data_matrix[,j])))
      p_values[i, j] <<- rho_post_sim(cormat[i, j], n) |> pvalfunc()
    })
  })
  pmat <- p_values + t(p_values) + diag(rep(1, nc))
  corrplot::corrplot(cormat, method = 'color', p.mat = pmat, insig = 'pch',
                     pch.cex = crosssize, tl.cex = textsize, tl.col='black')
  list(correlations = cormat, p_values = pmat) |> invisible()
}
```

In the diagram below the crosses indicate insignificant correlations (no evidence of deviation from the null hypothesis). However, the significance level has not been adjusted for multiple testing.

```
d |> select(Log_Revenue, Log_Age, Score) |> corrplot_exact()
```



We note that only the relationship between log revenue and log age appears significant in a univariate linear sense.

That said, we were asked to incorporate the score into our predictions, so we will include Score in the models, but not any interactions in this case (to avoid further over-fitting).

Ordinary regression

As a starting point for regression we implement ordinary least squares regression.

Ordinary regression is one of very few model types for which prediction intervals are directly available in R.

```
lm1 <- lm(Log_Revenue ~ Log_Age + Score, data = d)
lm1 |> summary()


Call:
lm(formula = Log_Revenue ~ Log_Age + Score, data = d)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.1867 -1.8946 -0.0484  1.2966  9.2275 

Coefficients:
            Estimate Std. Error t value     Pr(>|t|)    
(Intercept) 13.41038    2.36502   5.670 0.0000001562 ***
Log_Age      2.22517    0.38106   5.839 0.0000000745 ***
Score       -0.03676    0.56021  -0.066        0.948    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.404 on 94 degrees of freedom
```

```
Multiple R-squared:  0.2671,    Adjusted R-squared:  0.2515
F-statistic: 17.13 on 2 and 94 DF,  p-value: 0.0000004541
```

```
lm1 |> predict(newdata = d_target, interval = 'prediction')
```

```
        fit      lwr     upr
21 20.36077 15.50393 25.2176
```

Robust variant

Implementing robust regression is a good sensitivity analysis. By checking whether the relationships change we can assess the stability of our model.

```
lm2 <- MASS::rlm(Log_Revenue ~ Log_Age + Score, data = d)
lm2 |> summary()
```

```

Call: rlm(formula = Log_Revenue ~ Log_Age + Score, data = d)
Residuals:
     Min       1Q   Median       3Q      Max 
-4.19279 -1.67837  0.02641  1.47898  9.63170 

Coefficients:
            Value   Std. Error t value
(Intercept) 12.8898  2.2107     5.8305
Log_Age      2.3801  0.3562     6.6818
Score       -0.0509  0.5237    -0.0972

Residual standard error: 2.397 on 94 degrees of freedom
```

```
lm2 |> predict(newdata = d_target, interval = 'prediction')
```

```
Warning in predict.lm(lm2, newdata = d_target, interval = "prediction", : Assuming
constant prediction variance even though model fit is weighted
```

```
        fit      lwr      upr
21 20.28579 15.50606 25.06551
```

The robust fit is fairly similar, but we receive a warning that additional assumptions are made when attempting to create a prediction interval.

Bayesian regression

Now we do the same regression using a Bayesian simulation approach. The Bayesian regression results should be roughly in line with the ordinary results, but with added flexibility.

```
library(rstanarm)
mycores <- 3
options(mc.cores = mycores)
```

```
lm3 <- stan_glm(Log_Revenue ~ Log_Age + Score, data = d)
lm3 |> summary(digits = 2)
```

```

Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      Log_Revenue ~ Log_Age + Score
```

```
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 97
 predictors:   3

Estimates:
              mean   sd    10%   50%   90%
(Intercept) 13.44   2.38 10.39 13.42 16.49
Log_Age      2.23   0.38  1.74  2.23  2.71
Score       -0.05   0.57 -0.78 -0.04  0.68
sigma        2.42   0.18  2.20  2.41  2.65

Fit Diagnostics:
           mean   sd    10%   50%   90%
mean_PPD 19.44   0.35 18.99 19.45 19.89

The mean_ppd is the sample average posterior predictive distribution of the outcome
variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.03 1.00 5138
Log_Age       0.01 1.00 4705
Score         0.01 1.00 4722
sigma         0.00 1.00 4768
mean_PPD      0.01 1.00 4511
log-posterior 0.04 1.01 1580

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of
effective sample size, and Rhat is the potential scale reduction factor on split
chains (at convergence Rhat=1).
```

```
lm4 <- stan_lm(Log_Revenue ~ Log_Age + Score, data = d,
               prior = R2(summary(lm1)$r.squared, what = 'mean'))
```

```
Warning: There were 39 divergent transitions after warmup. See
https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.

Warning: Examine the pairs() plot to diagnose sampling problems
```

```
lm4 |> summary(digits = 2)
```

```


Model Info:
 function:     stan_lm
 family:       gaussian [identity]
 formula:      Log_Revenue ~ Log_Age + Score
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 97
 predictors:   3

Estimates:
                mean   sd    10%   50%   90%
```

```
(Intercept)   13.70   2.34 10.68 13.75 16.67
Log_Age        2.11   0.36  1.64  2.11  2.58
Score         -0.03   0.56 -0.73 -0.04  0.70
sigma          2.42   0.18  2.21  2.41  2.65
log-fit_ratio  0.00   0.07 -0.09  0.00  0.09
R2             0.25   0.07  0.16  0.25  0.33

Fit Diagnostics:
           mean   sd    10%   50%   90%
mean_PPD 19.45   0.35 19.00 19.45 19.89

The mean_ppd is the sample average posterior predictive distribution of the outcome
variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.07 1.00 1153
Log_Age       0.01 1.00 1602
Score         0.02 1.00 1301
sigma         0.00 1.00 2557
log-fit_ratio 0.00 1.00 2179
R2            0.00 1.00 1703
mean_PPD      0.01 1.00 4162
log-posterior 0.06 1.01  952

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of
effective sample size, and Rhat is the potential scale reduction factor on split
chains (at convergence Rhat=1).
```

Robust Bayesian regression with t-residuals

The *brms* package allows for a lot more distributions to be used, including Student-t. This has the effect of downweighting extreme residuals automatically, while maintaining accurate prediction intervals.

```
library(brms)

Warning: package 'brms' was built under R version 4.4.2

Loading 'brms' package (version 2.22.0). Useful instructions
can be found by typing help('brms'). A more detailed introduction
to the package is available through vignette('brms_overview').


Attaching package: 'brms'

The following objects are masked from 'package:rstanarm':

    dirichlet, exponential, get_y, lasso, ngrps

The following object is masked from 'package:stats':

    ar

lm5 <- brm(Log_Revenue ~ Log_Age + Score, data = d, family = 'student')

Compiling Stan program...
```

```
Start sampling
```

```
lm5 |> summary()
```

```
 Family: student 
  Links: mu = identity; sigma = identity; nu = identity 
Formula: Log_Revenue ~ Log_Age + Score 
   Data: d (Number of observations: 97) 
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Regression Coefficients:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept    12.98      2.26     8.54    17.28 1.00     4864     2843
Log_Age       2.37      0.38     1.62     3.10 1.00     4839     2812
Score        -0.06      0.53    -1.07     1.01 1.00     4662     2918

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     2.15      0.22     1.73     2.59 1.00     2936     2528
nu       15.46     10.65     3.93    44.77 1.00     3544     2947

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

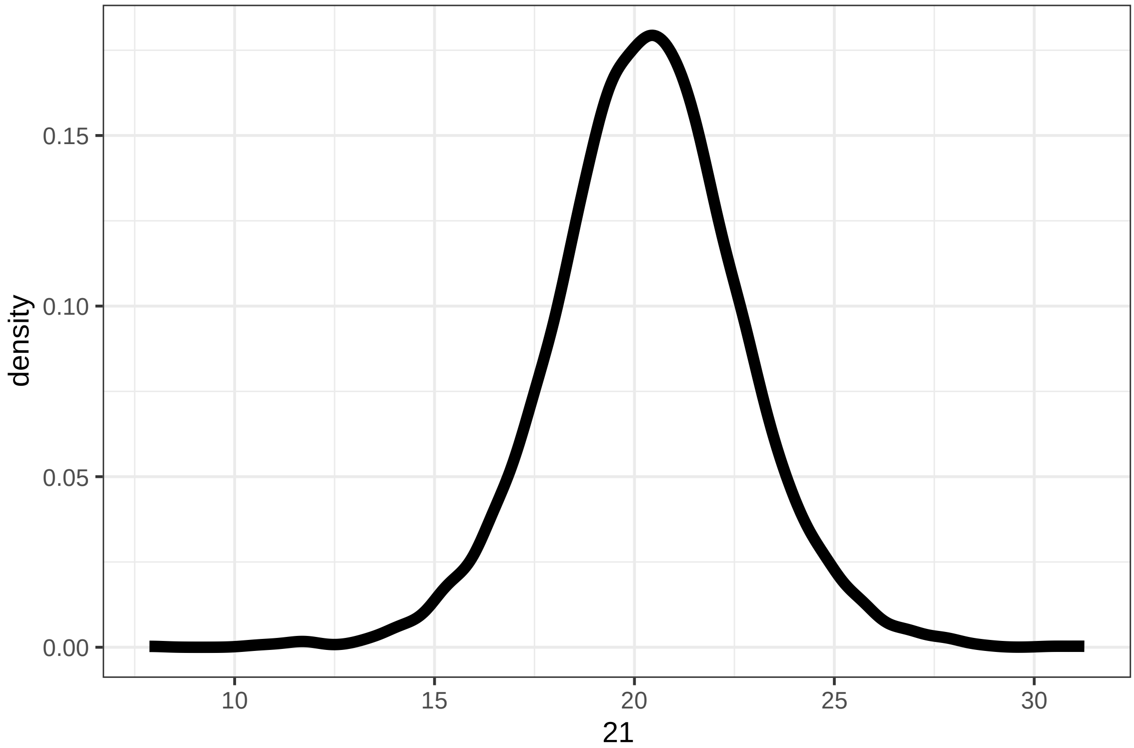
Prediction

As a last step, we will simulate the posterior predictive distribution of the final regression fit for the target observation.

```
post_sims <- lm5 |> as.data.frame()
new_data <- model.matrix(Log_Revenue ~ Log_Age + Score, data = d_target)
preds <- as.matrix(post_sims[,1:3]) %*% t(new_data) +
          rt(nrow(post_sims), post_sims$nu) * post_sims$sigma
```

First we give the predictive distribution on the log scale and see whether it agrees with the previous results.

```
preds |> ggplot(aes(x = `21`)) + geom_density(linewidth = 2) + theme_bw()
```




```
preds |> quantile(c(0.025, 0.975))

    2.5%    97.5%
15.55410 25.01376
```

Calculating the shortest interval might be superior to the symmetric interval.

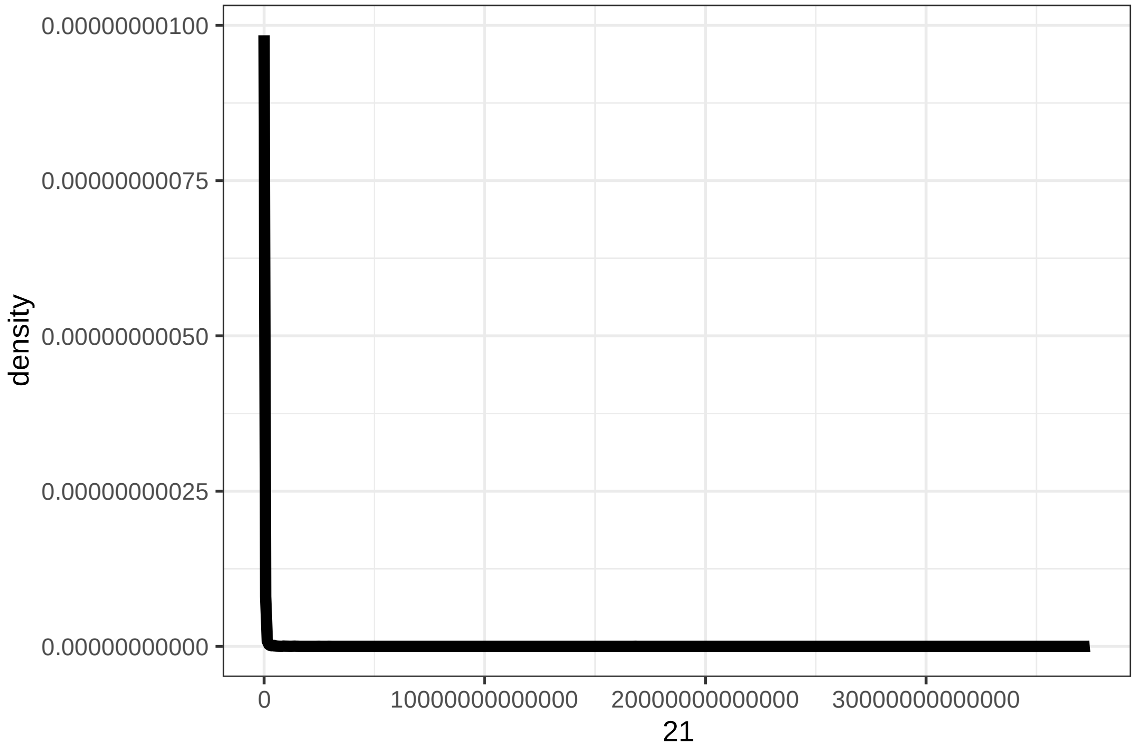
```
shortestinterval <- function(postsims, width=0.95) { # Coded by Sean van der Merwe, UFS
  sort(postsims) -> sorted.postsims
  round(length(postsims)*width) -> gap
  which.min(diff(sorted.postsims, gap)) -> pos
  sorted.postsims[c(pos, pos + gap)] }

preds |> shortestinterval()

[1] 15.46477 24.92022
```

Then we give the distribution and interval on the original scale.

```
preds |> exp() |> ggplot(aes(x = `21`)) + geom_density(linewidth = 2) + theme_bw()
```




```
preds |> exp() |> quantile(c(0.025, 0.975))
```

```
     2.5%       97.5%
  5689290 73002441224
```

```
preds |> exp() |> shortestinterval()
```

```
[1]        2614.926 32311459824.830
```

While it might seem useful to report on the original scale, and it usually is, in this case the results are completely useless.