Bootstrapping

James Caldwell

2024-02-26

Feb 2024

This R script demonstrates the concept of bootstrapping as a method for quantifying uncertainty in a fitted curve. It generates synthetic data using a defined data-generating process, fits a polynomial model to the data, and visualizes the true model alongside bootstrap samples and their confidence intervals. The script begins by defining functions for simulating data and the true mean function. It then generates synthetic data, creates scatterplots, fits polynomial models, and plots the true model and bootstrapped samples. The bootstrap resampling process involves randomly sampling observations with replacement from the original dataset, fitting polynomial models to each bootstrap sample, and calculating confidence intervals for the fitted curves.

```
library(tidymodels)# for optional tidymodels solutions
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.1.1 ──
```

```
## ✓ broom        1.0.5     ✓ recipes      1.0.9
## ✓ dials        1.2.0     ✓ rsample      1.2.0
## ✓ dplyr        1.1.4     ✓ tibble       3.2.1
## ✓ ggplot2      3.4.4     ✓ tidyr        1.3.0
## ✓ infer        1.0.5     ✓ tune         1.1.2
## ✓ modeldata    1.2.0     ✓ workflows    1.1.3
## ✓ parsnip      1.1.1     ✓ workflowsets 1.0.1
## ✓ purrr        1.0.2     ✓ yardstick    1.2.0
```

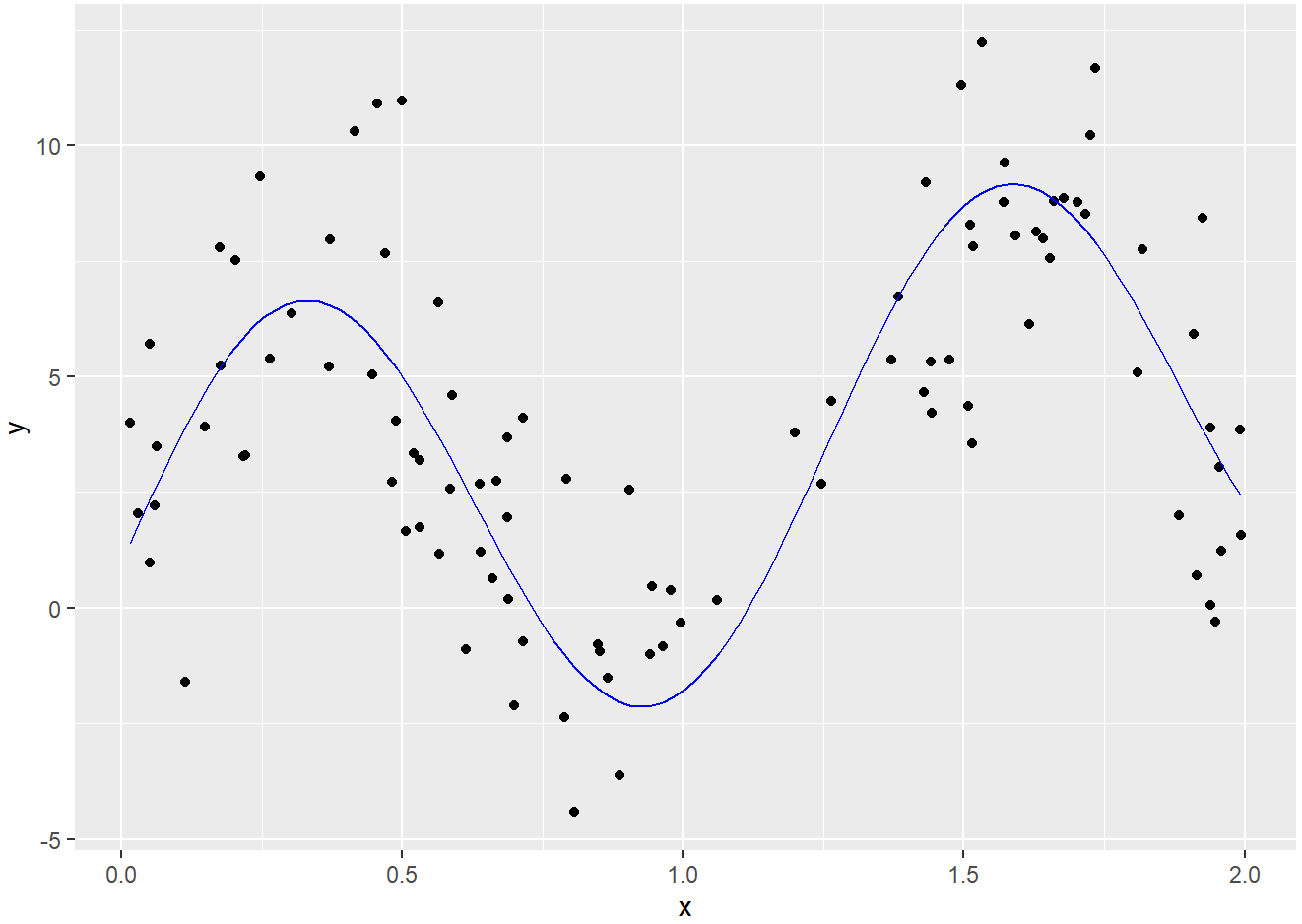
```
## ── Conflicts ────────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::discard() masks scales::discard()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ✖ recipes::step()  masks stats::step()
## • Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(tidyverse) # functions for data manipulation
```

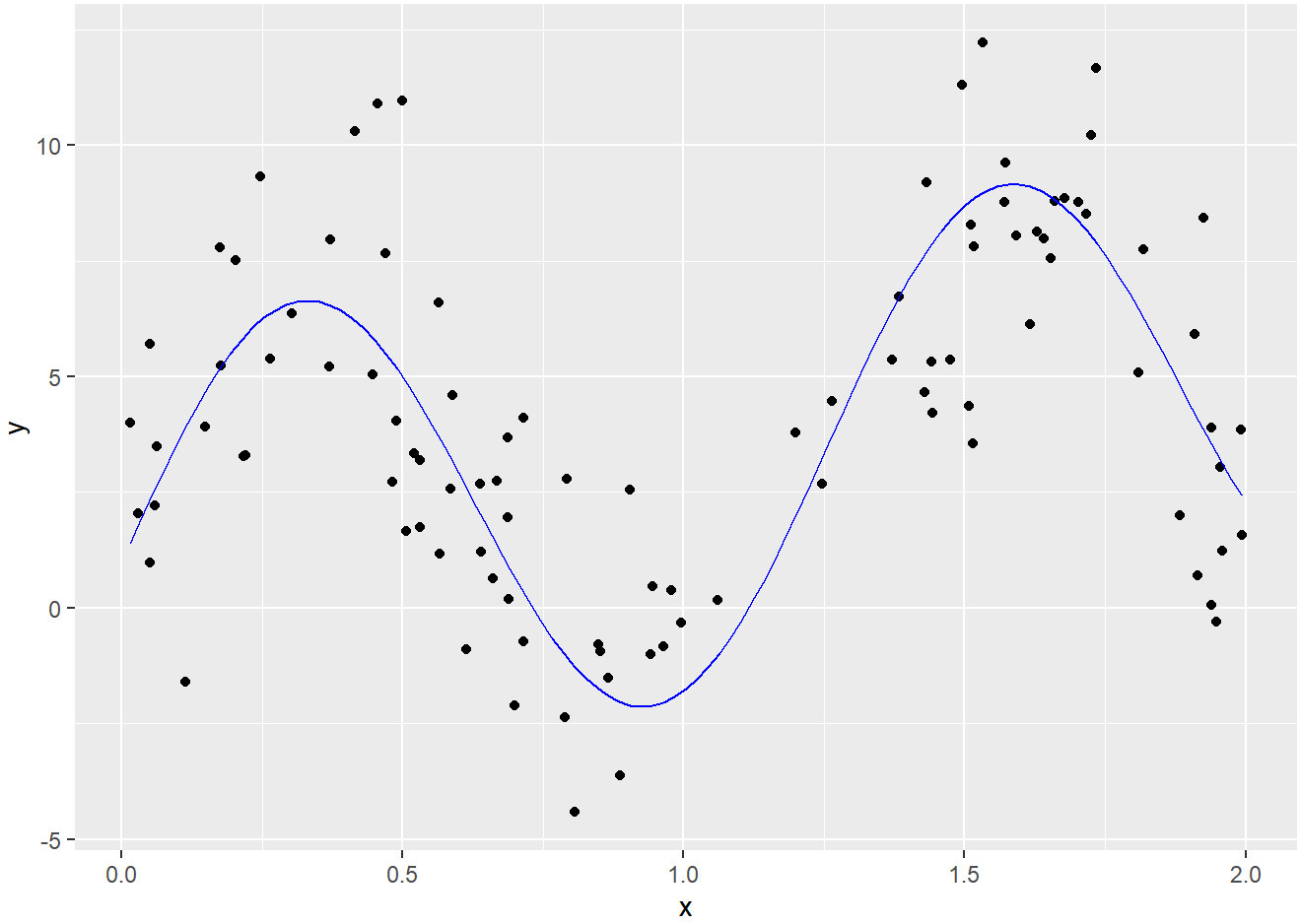
```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.5
## ✓ lubridate 1.9.3     ✓ stringr   1.5.1
```

```
## ── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ purrr::discard()    masks scales::discard()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ readr::spec()       masks yardstick::spec()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

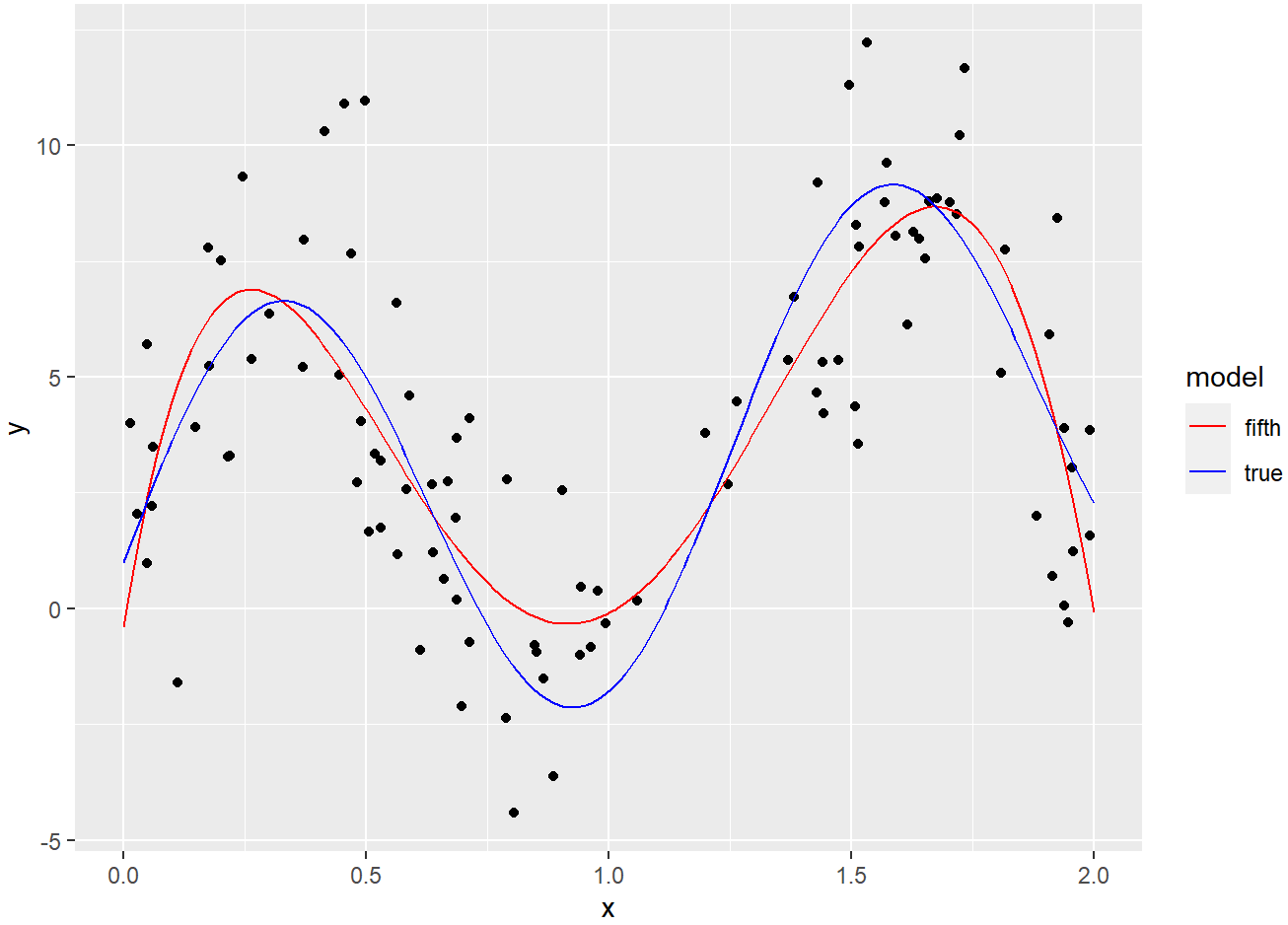
```
library(broom)
library(splines)

#Bootstrapping

# Bootstrap resampling can be used to quantify the uncertainty in a fitted curve.

## Data Generating Process

sim_x <- function(n) runif(n,min = 0, max = 2) # U[0,2]
f <- function(x) 1 + 2*x + 5*sin(5*x) # true mean function
sim_y <- function(x){ # generate Y|X from N{f(x),sd}
  n = length(x)
  f(x) + rnorm(n, mean = 0, sd = 2.5)

}

#-- Settings
n = 100       # number of observations

#-- Generate Data
set.seed(211)
x = sim_x(n)
y = sim_y(x)

data_train = tibble(x,y)

#-- Scatterplot: Tidyverse
ggplot(tibble(x,y), aes(x,y)) +
  geom_point() +
  geom_function(fun=f, color="blue")
```




```
#-- Settings
n = 100       # number of observations

#-- Generate Data
set.seed(211)
x = sim_x(n)
y = sim_y(x)

data_train = tibble(x,y)

#-- Scatterplot: Tidyverse
ggplot(tibble(x,y), aes(x,y)) +
  geom_point() +
  geom_function(fun=f, color="blue")
```



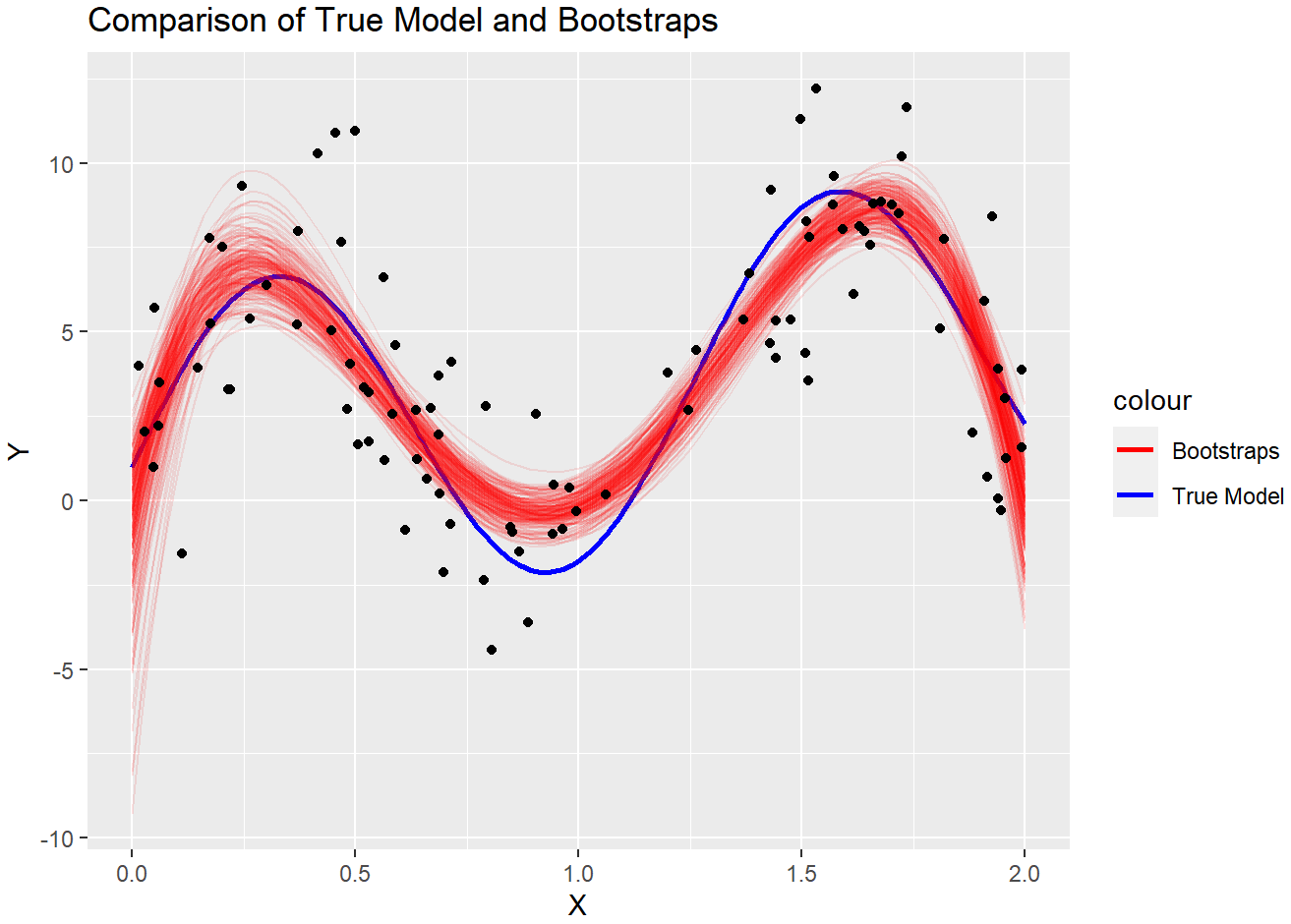

```
xseq = seq(0, 2, length=200)
fit5 = lm(y~poly(x,5))   # 5th deg polyfit
yhat5 = predict(fit5, tibble(x=xseq))

#: make data for plotting; convert to long format
pred.data = tibble(x=xseq, fifth=yhat5, true=f(xseq)) %>%
  pivot_longer(cols=-x, names_to="model", values_to="y")

#: plot
ggplot(tibble(x,y), aes(x,y)) +
  geom_point() +
  geom_line(data=pred.data, aes(color=model)) +
  scale_color_manual(values=c(true = "blue", fifth="red"))
```


10
5
0
-5
y
0.0
0.5
1.0
1.5
2.0
x
model
fifth
true

```
M = 200 # number of bootstrap samples

eval_pts = tibble(x=seq(0, 2, length=100))

yhat5 = matrix(NA, length(eval_pts), M) # initialize matrix for fitted values
yhat5 = matrix(NA, 100, M) # initialize matrix for fitted values

set.seed(212)

for(m in 1:M){
  # sample indices/rows from empirical distribution (with replacement)
  ind = sample(n, replace=TRUE)

  data_boot = data_train[ind,]
  m_boot = lm(y~poly(x,5), data=data_boot) #

  yhat5[,m] = predict(m_boot, eval_pts)

}

# : Convert to tibble and plot
data_fit = as_tibble(yhat5, .name_repair = "unique_quiet") %>% # convert matrix to tibble
  bind_cols(eval_pts) %>% # add the eval points
  pivot_longer(-x, names_to="simulation", values_to="y") # convert to long format

# print(head(yhat5,15))
print(head(data_fit,30))
```

```
## # A tibble: 30 × 3
##        x simulation       y
##    <dbl> <chr>        <dbl>
##  1     0 ...1         0.521
##  2     0 ...2        -2.61
##  3     0 ...3         0.539
##  4     0 ...4         2.69
##  5     0 ...5        -1.06
##  6     0 ...6        -0.0254
##  7     0 ...7         0.201
##  8     0 ...8         1.08
##  9     0 ...9        -0.317
## 10     0 ...10       -1.94
## # ℹ 20 more rows
```

```
true_data <- pred.data %>%
  filter(model == "true")

ggplot(data_train, aes(x, y)) +
  geom_line(data = true_data, aes(color = "True Model"), linetype = "solid", linewidth = 1) +
  geom_line(data = data_fit, aes(color = "Bootstraps", group = simulation), alpha = 0.1, linewid
th = 0.5) +
  geom_point() +
  scale_color_manual(values = c("True Model" = "blue", "Bootstraps" = "red"), labels = c("True M
odel" = "True Model", "Bootstraps" = "Bootstraps")) +
  labs(title = "Comparison of True Model and Bootstraps",
       x = "X",
       y = "Y")
```




```
print(dim(yhat5))
```

```
## [1] 100 200
```

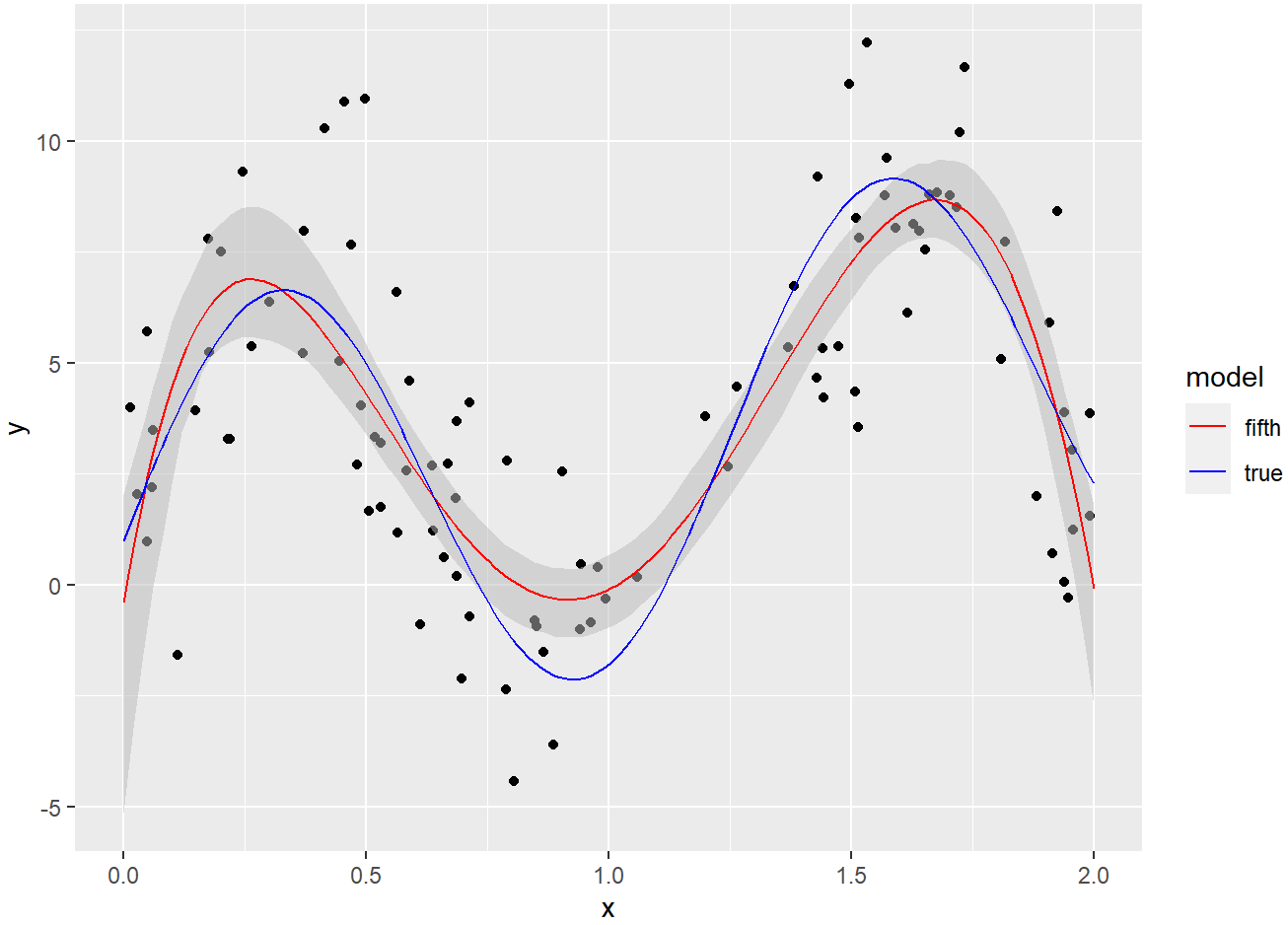
```
# Calculate pointwise 95% confidence intervals
upper_ci <- apply(yhat5, 1, quantile, probs = 0.975)
lower_ci <- apply(yhat5, 1, quantile, probs = 0.025)

#-- Generate Data
set.seed(211)
x = sim_x(n)
y = sim_y(x)
ci_x <- seq(0, 2, length.out = 100)

# Create data frame for upper and lower confidence intervals
# ci_data <- data.frame(x = xseq, upper_ci = upper_ci, lower_ci = lower_ci)
ci_data <- data.frame(x = x, y = y, upper_ci = upper_ci, lower_ci = lower_ci,ci_x = ci_x)

print(head(ci_data))
```

```
##           x         y upper_ci    lower_ci       ci_x
## 1 0.4695968 7.6568711 2.035368 -5.14275394 0.00000000
## 2 1.8104933 5.0821303 2.836006 -3.22463377 0.02020202
## 3 1.9394062 3.8859839 3.605344 -1.54940480 0.04040404
## 4 0.6884168 0.1899454 4.467958 -0.04702497 0.06060606
## 5 0.5311731 1.7439802 5.197669  1.10548940 0.08080808
## 6 0.0293828 2.0464239 5.923988  2.26503192 0.10101010
```

```
print(dim(ci_data))
```

```
## [1] 100   5
```

```
ggplot(ci_data,
       aes(x,y)) +
  geom_point() +
  geom_ribbon(data = ci_data, aes(x = ci_x, ymin = lower_ci, ymax = upper_ci), fill = "gray", al
pha = 0.5) +
  geom_line(data=pred.data, aes(color=model)) +
  scale_color_manual(values=c(true = "blue", fifth="red"))
```


10
5
0
-5
y
0.0
0.5
1.0
1.5
2.0
x
model
fifth
true