Homework 11

Neal Kar (ink2105)

Announcement

Please do not add code folding (code_folding: hide) to your YAML Header or echo = FALSE to your RMD code chunk options. In order to accurately grade your HTML files we need to be able to see all of your code. Thank you!

Question 1

Please download the R download instructions and package_install.R script from courseworks. After following the instructions, you should be able to open RStudio and use R locally. Once you have RStudio working, run the package_install.R script to install the packages we commonly use for homework.

Once you have tried to get R/RStudio running, please write below whether you were successful. If you had any problems or ran into any issues, please let us know in this space as well.

I was successful.

Question 2

In this question we will go through fitting a polynomial regression to a new data set using k-fold cross-validation.

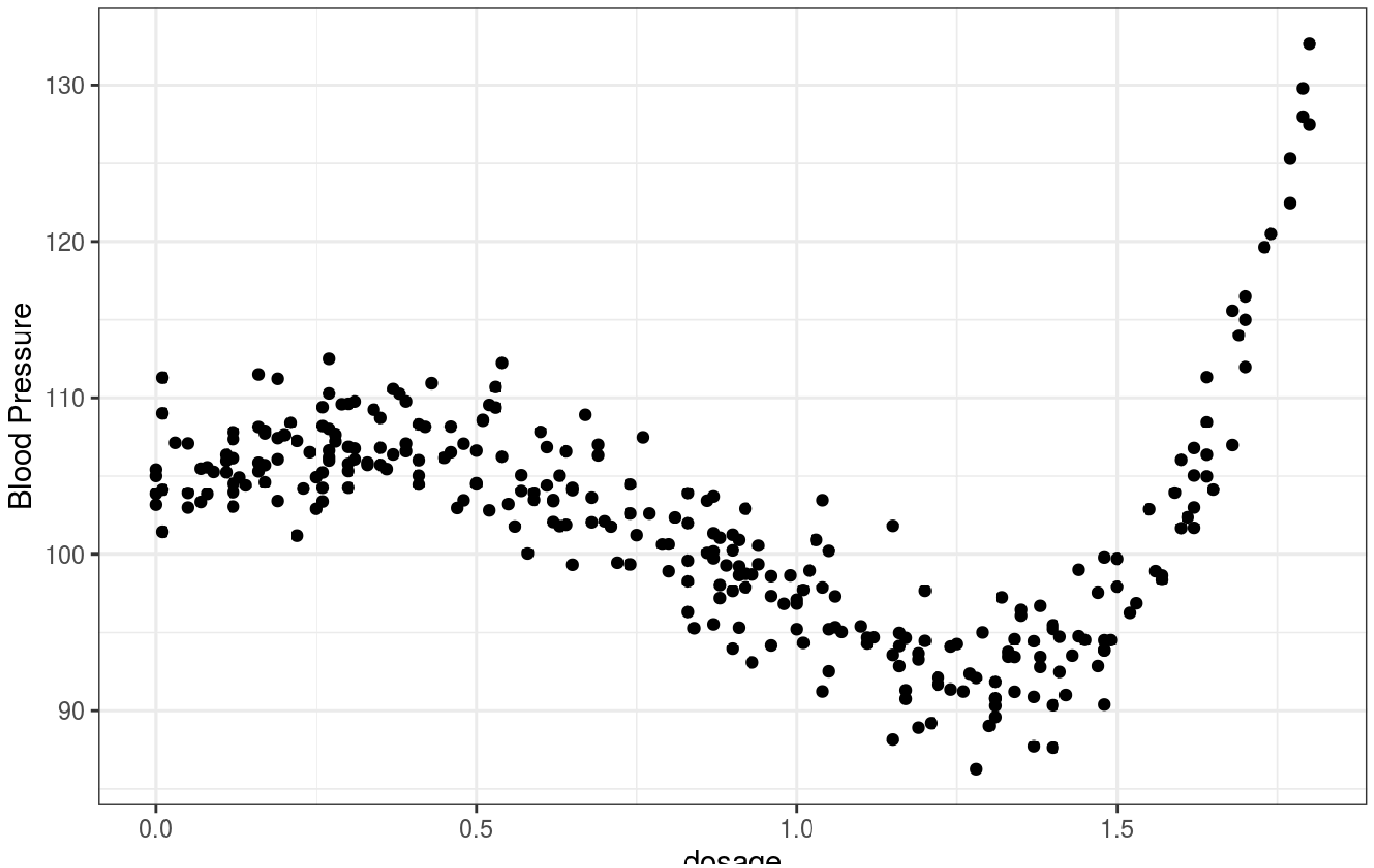
a)

Load the simulated `bp_dosage.csv` *file into R. For this exercise we will be trying to build a model that predicts blood pressure (* `bp` *) based on continuous dosage* `dosage` *.*

Make a graph of dosage and blood pressure. Does it look like there is a linear relationship between continuous dosage and blood pressure? Describe in a sentence how blood pressure seems to change as dosage increases.

```r
#Read in the data frame
dosage_df <- read_csv("data/bp_dosage.csv")

#Create the graph
ggplot(data = dosage_df) +
  geom_point(aes(x=dosage, y=bp)) +
  theme_bw() +
  labs(title="Relationship Between Dosage and Blood Pressure,
       x=Dosage", y="Blood Pressure")
```

Relationship Between Dosage and Blood Pressure,
x=Dosage



dosage

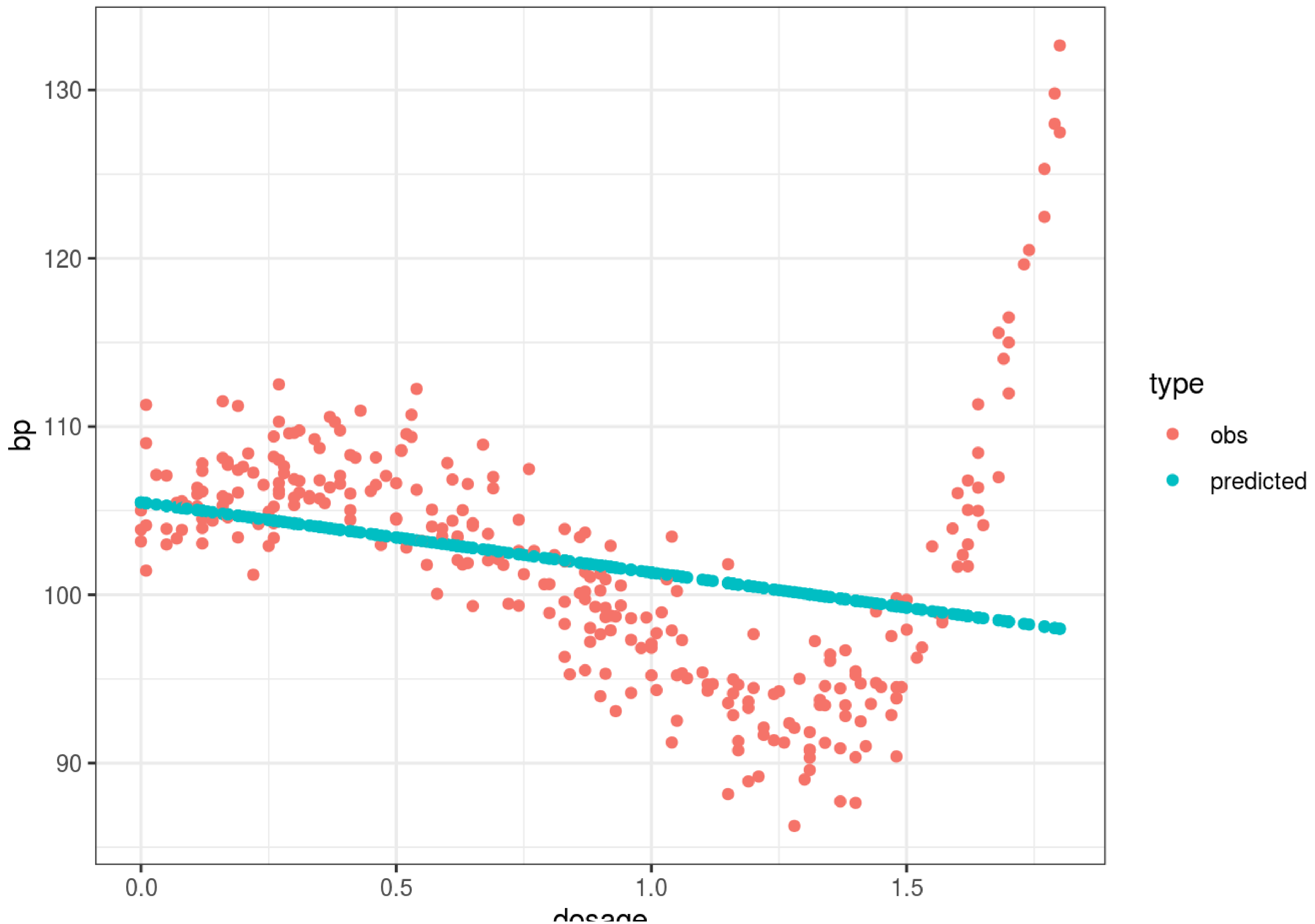
No, it doesn’t look like there is a linear relationship between the variables. The relationship may be descibed by a cubic or quartic function. Initially, dosage and bp are inversely related, but then beyond a dosage of 1.25 units, the relationship appears to become a direct relationship, with a very sharp increase in bp after 1.5 units of dosage.

b)

Adapt the code from lab to create a function that will fit a polynomial model to the blood pressure data. Your function should take take a data frame and a number (degree), and should output a fit polynomial model of that degree.

Test that your function works by fitting polynomial models of degree 1, 5, and 10 to the full blood pressure data.

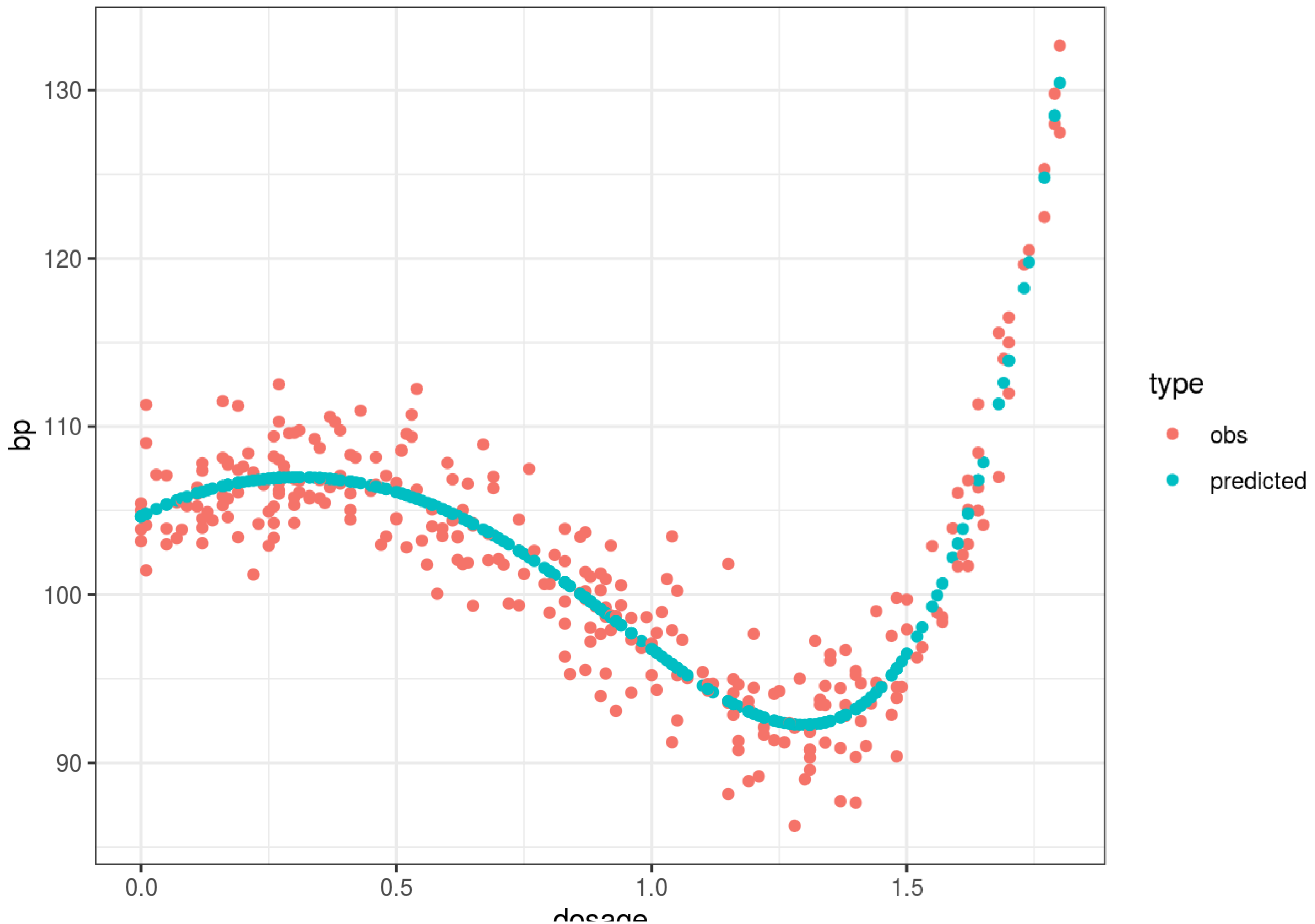
```r
#Create function
make_pred_graph <- function(df, model){
  preds <- augment(model, newdata = df) %>%
    select(dosage, obs = bp, predicted = .fitted) %>%
    pivot_longer(cols = 2:3,
                 values_to = "bp",
                 names_to = "type")
  plot <- ggplot(preds) +
    geom_point(aes(x = dosage, y = bp, color = type)) +
    theme_bw()
  return(plot)
}

### Test the function ###

#Degree = 1
deg_1 <- lm(bp ~ poly(dosage, 1), data = dosage_df)
make_pred_graph(dosage_df, deg_1)
```



130
120
110
100
90
bp
0.0
0.5
1.0
1.5
dosage
type
obs
predicted

dosage

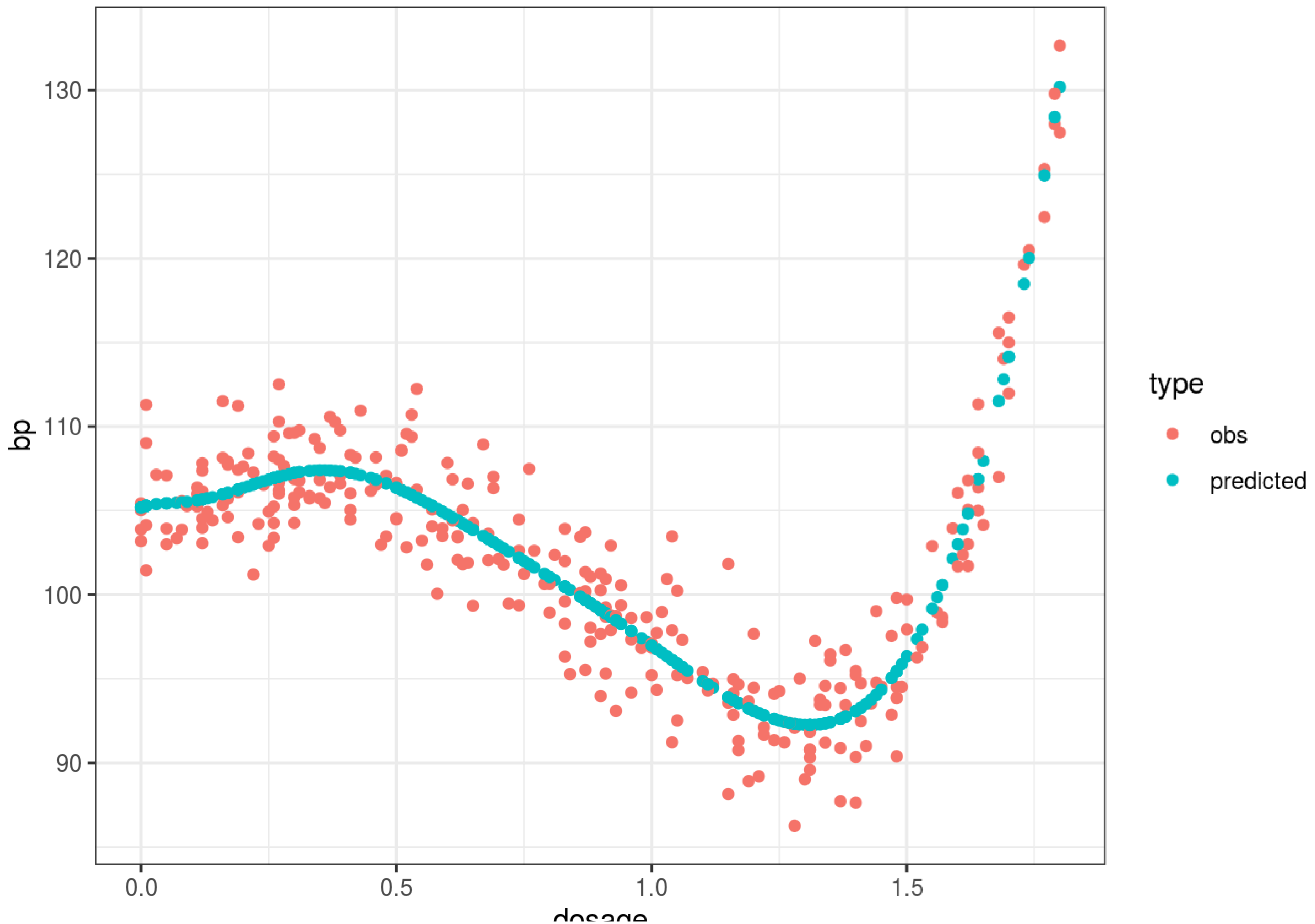
```r
#Degree = 5
deg_5 <- lm(bp ~ poly(dosage, 5), data = dosage_df)
make_pred_graph(dosage_df, deg_5)
```



130
120
110
100
90
bp
0.0
0.5
1.0
1.5
dosage
type
obs
predicted

dosage

```
#Degree = 10
deg_10 <- lm(bp ~ poly(dosage, 10), data = dosage_df)
make_pred_graph(dosage_df, deg_10)
```



bp
130
120
110
100
90
0.0
0.5
1.0
1.5
dosage
type
obs
predicted

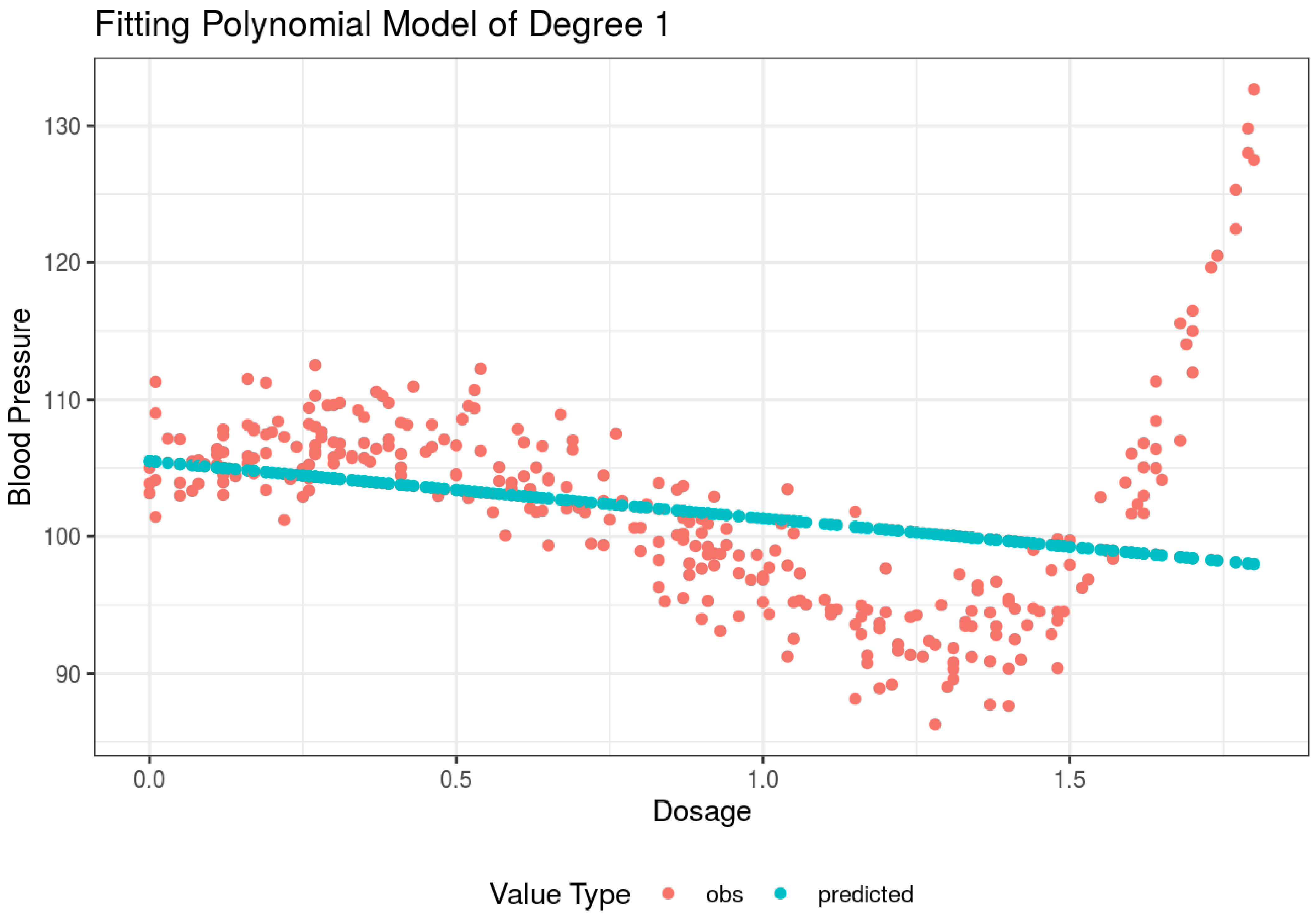
dosage

c)

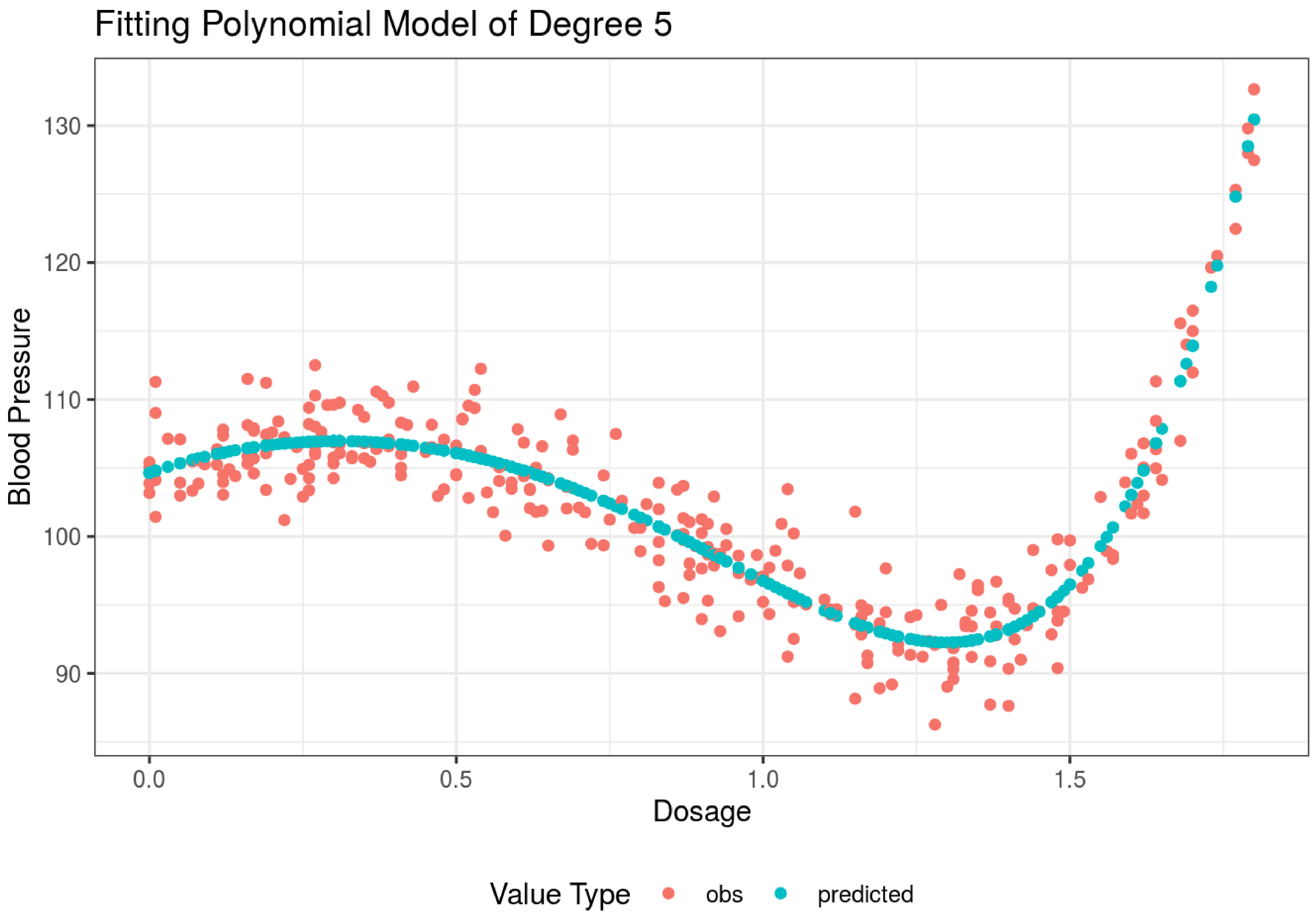
Next, adapt the function from lab that plots predicted values of a polynomial model. Make it so that it works with the variable names for the blood pressure data, and make it so that the title of the graph mentions the degree of the polynomial fit (you can use a new function argument for this). Use this function to make plots of the three models you fit on the full data set in part (b).

```
#Update function
make_pred_graph_v2 <- function(df, model, degree){

  preds <- augment(model, newdata = df) %>%
    select(dosage, obs = bp, predicted = .fitted) %>%
    pivot_longer(cols = 2:3,
                 values_to = "bp",
                 names_to = "type")
  plot <- ggplot(preds) +
    geom_point(aes(x = dosage, y = bp, color = type)) +
    theme_bw() +
    theme(legend.position = "bottom") +
    labs(title=paste("Fitting Polynomial Model of Degree",
                     degree), x="Dosage", y="Blood Pressure",
         color="Value Type")
  return(plot)
}

### Test the function ###
```

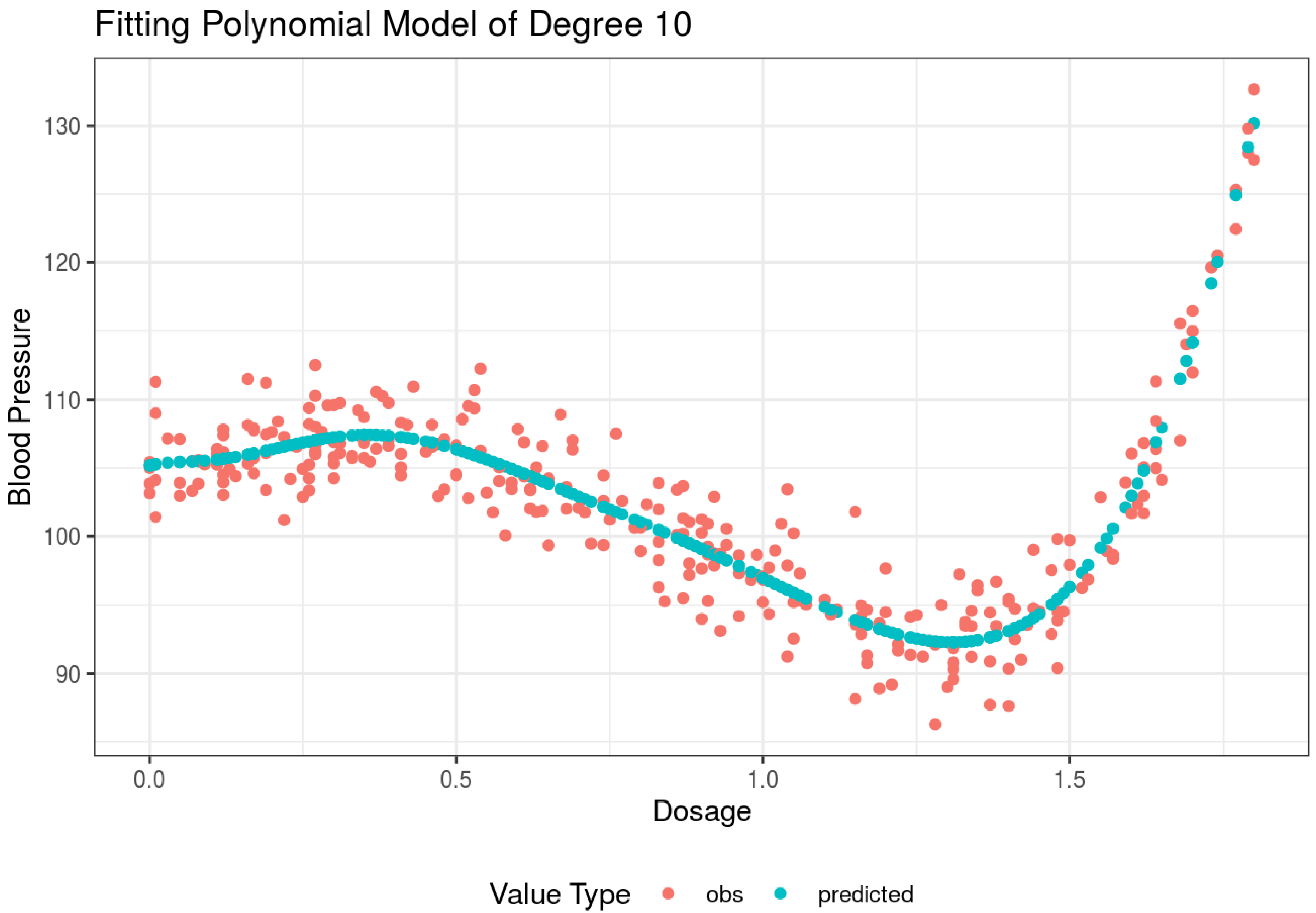
```
#Degree = 1
make_pred_graph_v2(dosage_df, deg_1, "1")
```


Fitting Polynomial Model of Degree 1
130
120
110
100
90
Blood Pressure
0.0
0.5
1.0
1.5
Dosage
Value Type
obs
predicted

```
#Degree = 5
make_pred_graph_v2(dosage_df, deg_5, "5")
```


Fitting Polynomial Model of Degree 5
Blood Pressure
130
120
110
100
90
0.0
0.5
1.0
1.5
Dosage
Value Type
obs
predicted

```
#Degree = 10
make_pred_graph_v2(dosage_df, deg_10, "10")
```


Fitting Polynomial Model of Degree 10
Blood Pressure
130
120
110
100
90
0.0
0.5
1.0
1.5
Dosage
Value Type
obs
predicted

d)

Now let’s begin the cross-validation and fitting process. First split the data into training and test sets. Make the training set a random sample of 80% of the original data, and make the test set all remaining observations (other 20%) from the original dataset.

```
## Separate Training Data Set and Test Set
dosage_training_set <- sample_frac(dosage_df, 0.80)

dosage_test_set <- dosage_df %>%
  anti_join(dosage_training_set, by = "patient_id")
```

e)

Now copy the `create_folds` function from Lab 10. Confirm that it works by making two new datasets out of the training set: one with 5 folds and one with 10 folds. Present a summary of the number of observations in each fold.

```r
## Create folds for cross-validation

create_folds <- function(df, k){
  df_k <- df %>%
    sample_frac() %>%
    mutate(fold = 1 + (row_number()-1) %/% (nrow(df)/k))
  return(df_k)
}

# Test that this works:

create_fold_5 <-
  create_folds(df = dosage_training_set, k = 5) %>%
  group_by(fold) %>%
  count()

kable(create_fold_5)
```

fold	n
1	48
2	48
3	48
4	48
5	48

```
create_fold_10 <-
  create_folds(df = dosage_training_set, k = 10) %>%
  group_by(fold) %>%
  count()

kable(create_fold_10)
```

fold	n
1	24
2	24
3	24
4	24
5	24
6	24
7	24
8	24
9	24
10	24

5-fold: 48 observations in each fold.

10-fold: 24 observations in each fold.

f)

Next, adapt the `get_RMSE` *function from Lab 10 so that it works correctly with the new data set. Additionally, copy the* `fit_and_assess` *function from Lab 10. Explain how the* `fit_and_assess` *function works: what is the purpose of the* `f` *argument? Which part of the data frame input* `df` *is used to fit the model, and which part of the data frame input* `df` *is used to evaluate the model using* `get_RMSE` *?*

```r
## Function to perform polynomial regression

fit_poly <- function(df, degree){
  pre_form <- str_c("bp ~ poly(dosage,", degree, ")")
  form <- as.formula(pre_form)
  model <- lm(form, data = df)
  return(model)
}

## Function to get RMSE

get_RMSE <- function(df, model){
  predicted_df <- augment(model, newdata = df) %>%
    select(dosage, bp, .fitted)
  rmse <- RMSE(predicted_df$.fitted, predicted_df$bp)
  return(rmse)
}
```

```r
# Fit model on all but one fold, test on last fold
fit_and_assess <- function(df, f, degree) {
  holdout <- df %>%
    filter(fold == f)
  train <- df %>%
    filter(fold != f)
  train_mod <- fit_poly(df = train, degree = degree)
  holdout_rmse <- get_RMSE(df = holdout, model = train_mod)
  return(holdout_rmse)
}
```

The purpose of the “f” argument is to designate which folds in the dataset should be set aside for training and for testing the model. The portion of the data frame set aside for training is used to fit the model. The whole data frame (training and testing portions) are used to evaluate the model.

g)

Now import the `perform_k_fold_cv` *function from Lab 10. Check that it is working by performing 10-fold cross-validation for degree 1, 5, and 10 polynomials. Report the cross-validated RMSE for each of these degrees in a sentence.*

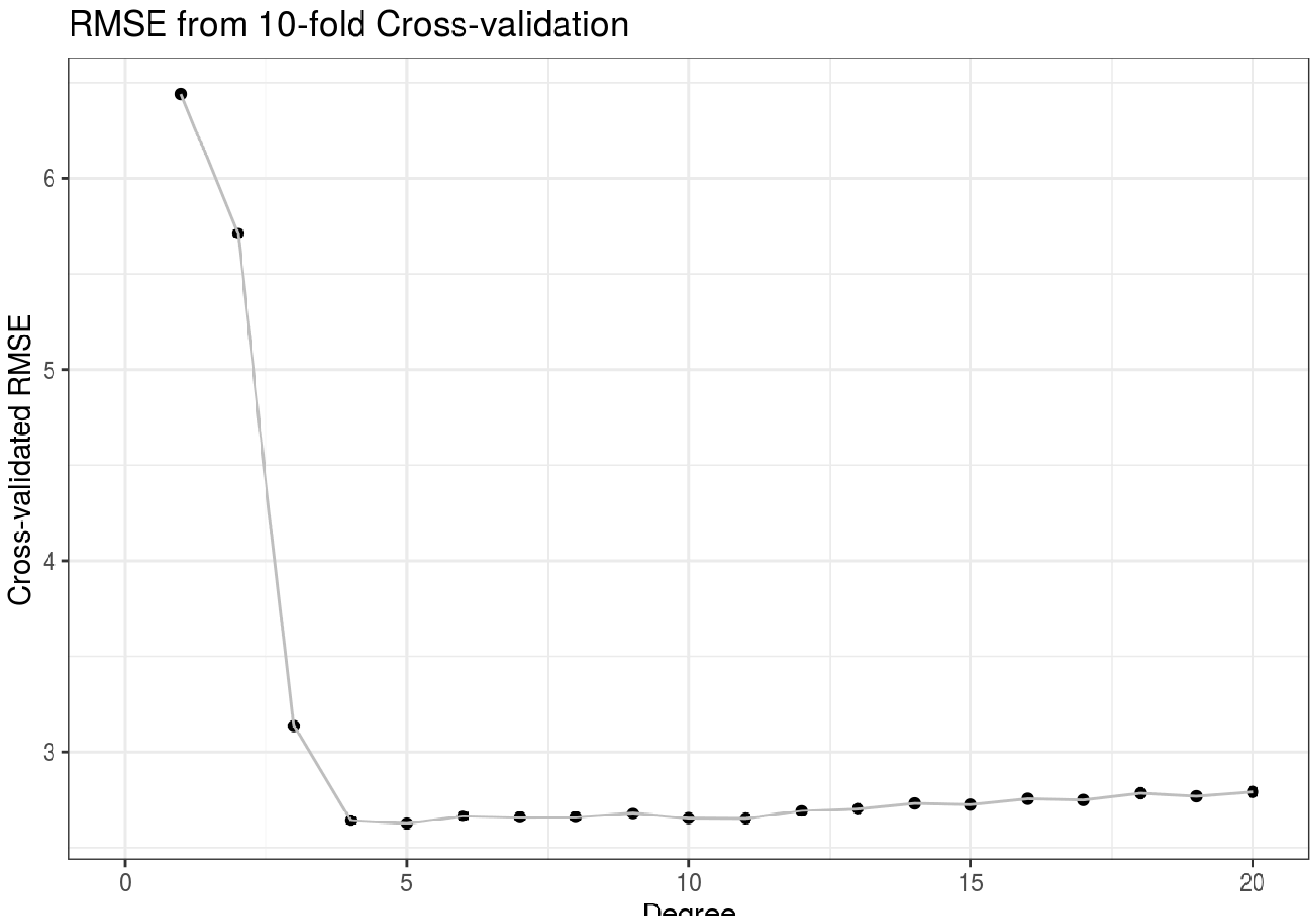
```r
# k-fold cross-validation

perform_k_fold_cv <- function(df, k, degree){
  folded_df <- create_folds(df = df, k = k)
  holdout_rmse_vals <- map_dbl(1:k, function(x) fit_and_assess(df = folded_df, x, degree = degree))
  mean_rmse <- mean(holdout_rmse_vals)
  return(mean_rmse)
}

rmse_1 <- perform_k_fold_cv(df = dosage_training_set, k = 10, degree = 1)

rmse_5 <- perform_k_fold_cv(df = dosage_training_set, k = 10, degree = 5)

rmse_10 <- perform_k_fold_cv(df = dosage_training_set, k = 10, degree = 10)
```

For degree 1, RMSE is 6.2935317, for degree 5, RMSE is 2.6438025, and for degree 10, RMSE is 2.6648804.

h)

Finally use `map_dbl()` *to obtain 10-fold cross-validated RMSE values for polynomial models from degree 1 to degree 20. Create a graph that shows the cross-validated RMSE values across different polynomial degrees.*

```r
# Perform the same action over varying degrees of polynomials (f
rom 1 to 20)

degree_vals <- map_dbl(1:20, function(x) perform_k_fold_cv(df =
dosage_training_set, k = 10, degree = x))

cv_res_tib <- tibble(degree = 1:20, cv_rmse = degree_vals)

ggplot(data = cv_res_tib) +
  geom_point(aes(x = degree, y = cv_rmse)) +
  geom_line(aes(x = degree, y = cv_rmse), color = "grey") +
  theme_bw() +
  scale_x_continuous(limits = c(0, 20)) +
  labs(title = "RMSE from 10-fold Cross-validation",
       x = "Degree", y = "Cross-validated RMSE")
```


RMSE from 10-fold Cross-validation
Cross-validated RMSE
6
5
4
3
0
5
10
15
20
Degree

i)

Based on your results and graph in question (h), which degree polynomial do you think performs best?

```
cv_res_tib <- cv_res_tib %>%
  arrange(cv_rmse)

kable(cv_res_tib)
```

degree	cv_rmse
5	2.628086
4	2.643646
11	2.654576
10	2.656299

degree	cv_rmse
7	2.661385
8	2.662015
6	2.667949
9	2.682271
12	2.696139
13	2.707703
15	2.730398
14	2.736712
17	2.754306
16	2.760371

degree	cv_rmse
19	2.773392
18	2.788962
20	2.795710
3	3.137799
2	5.714054
1	6.441663

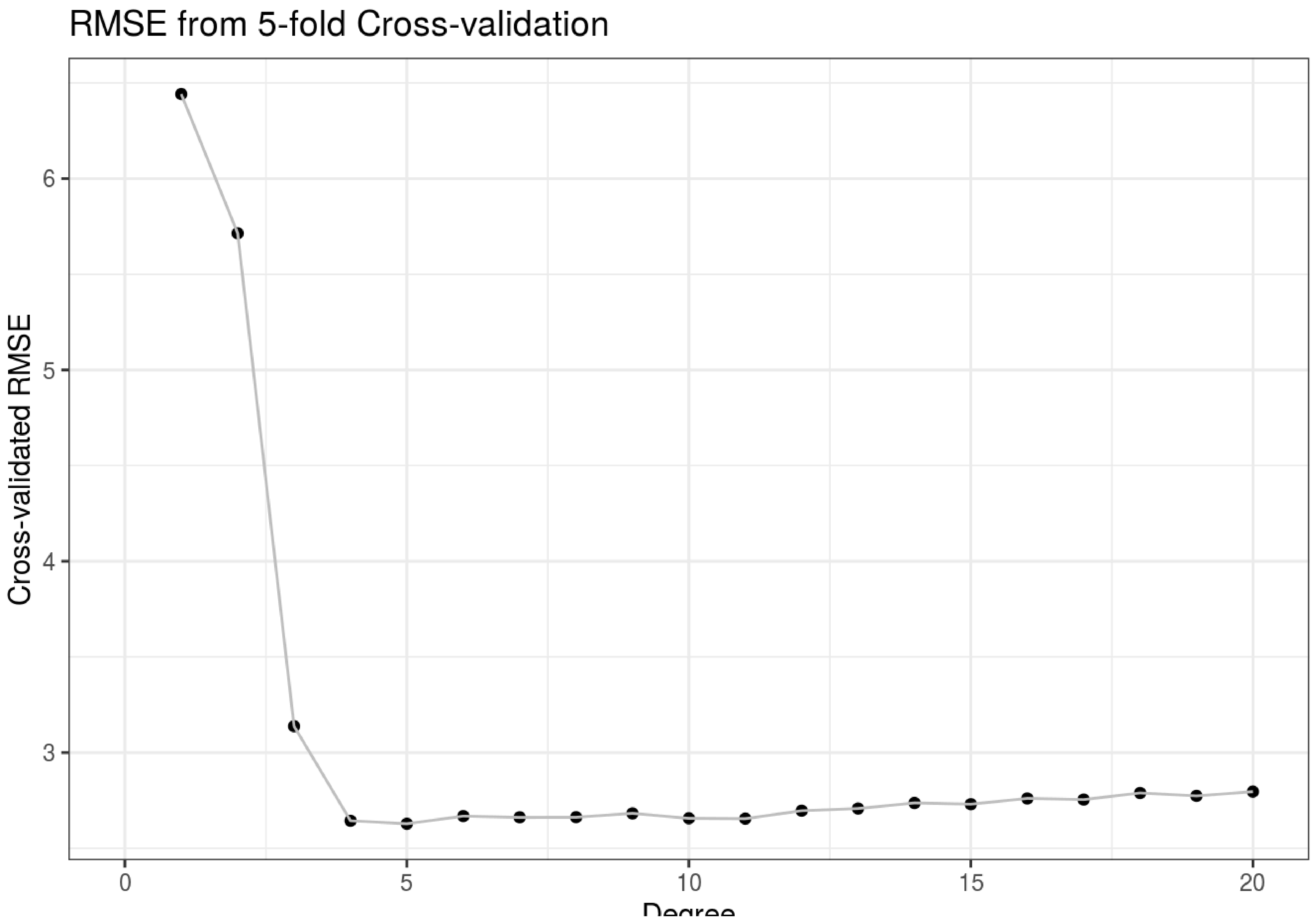
```r
#pull degree number
cv_res_tib_degr <- cv_res_tib$degree[1]
```

I think the polynomial of degree 5 performs best because it has the lowest RMSE value.

j)

The most common "k" chosen in k-fold cross-validation is 10, but with smaller datasets 5-fold cross-validation is also used. Reproduce the graph in (h) by performing 5-fold cross-validation to obtain RMSE values from degree 1 to 20. Does the graph look different? Would you have come to a different decision about which degree polynomial performs best?

```r
# Perform the same action over varying degrees of polynomials (f
rom 1 to 20) with 5-fold cross-validation

degree_vals_k5 <- map_dbl(1:20, function(x) perform_k_fold_cv(df
= dosage_training_set, k = 5, degree = x))

cv_res_tib_k5 <- tibble(degree = 1:20, cv_rmse = degree_vals)

ggplot(data = cv_res_tib_k5) +
  geom_point(aes(x = degree, y = cv_rmse)) +
  geom_line(aes(x = degree, y = cv_rmse), color = "grey") +
  theme_bw() +
  scale_x_continuous(limits = c(0, 20)) +
  labs(title = "RMSE from 5-fold Cross-validation",
       x = "Degree", y = "Cross-validated RMSE")
```



RMSE from 5-fold Cross-validation
Cross-validated RMSE
3
4
5
6
0
5
10
15
20
Degree

Degree

```
cv_res_tib_k5 <- cv_res_tib_k5 %>%
  arrange(cv_rmse)

kable(cv_res_tib_k5)
```

degree	cv_rmse
5	2.628086
4	2.643646
11	2.654576
10	2.656299
7	2.661385
8	2.662015
6	2.667949

degree	cv_rmse
9	2.682271
12	2.696139
13	2.707703
15	2.730398
14	2.736712
17	2.754306
16	2.760371
19	2.773392
18	2.788962
20	2.795710

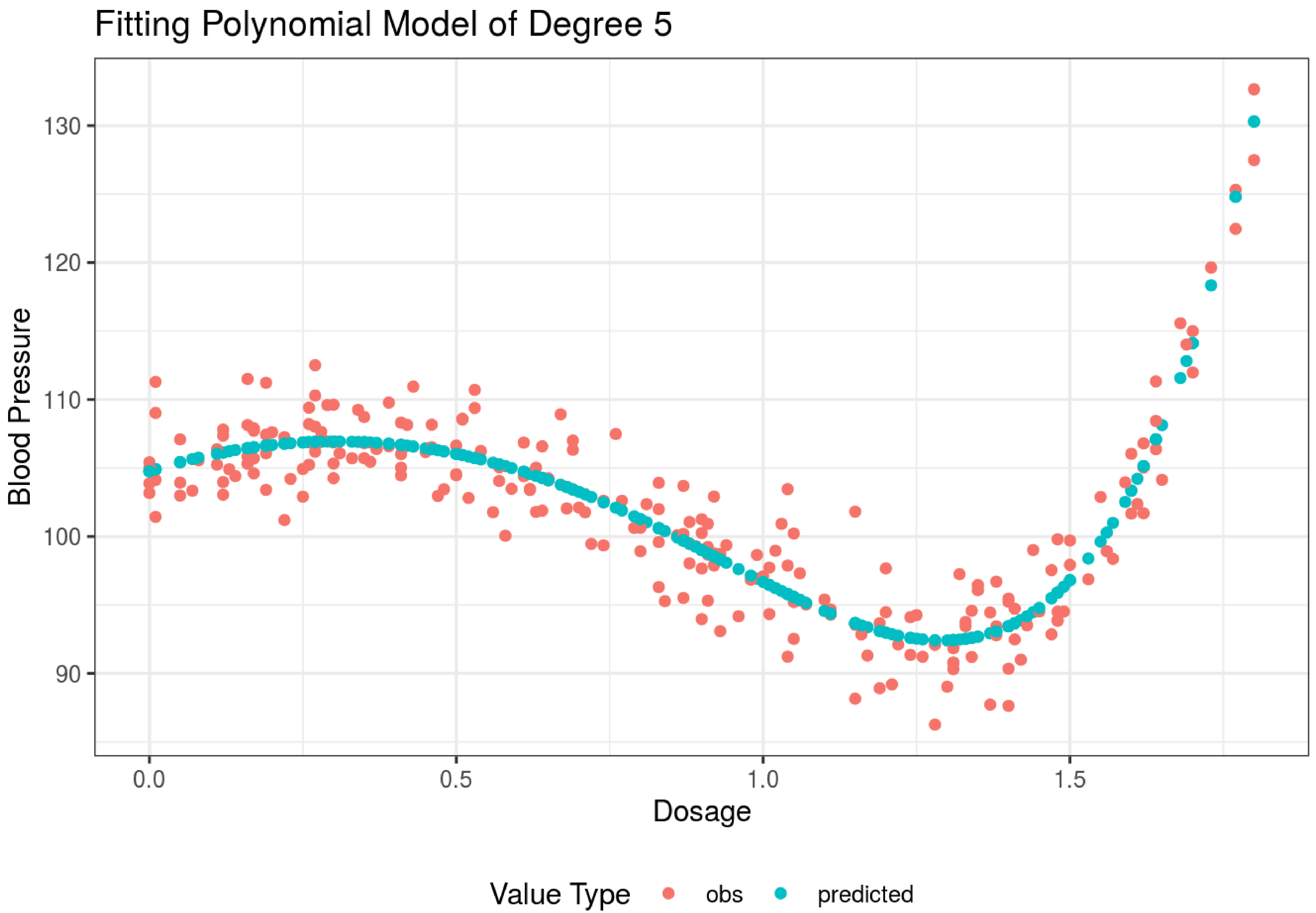
degree	cv_rmse
3	3.137799
2	5.714054
1	6.441663

The graph does not look different. I would not come to a different decision.

k)

Now, choose what you believe to be the best performing degree value and fit a polynomial model to the entire training data set. Create a graph showing predicted values for the training data set.

```
# Final model:

final_model <- fit_poly(df = dosage_training_set,
                        degree = cv_res_tib_degr)

make_pred_graph_v2(df = dosage_training_set, model = final_mode
l,
                   degree = as.character(cv_res_tib_degr))
```

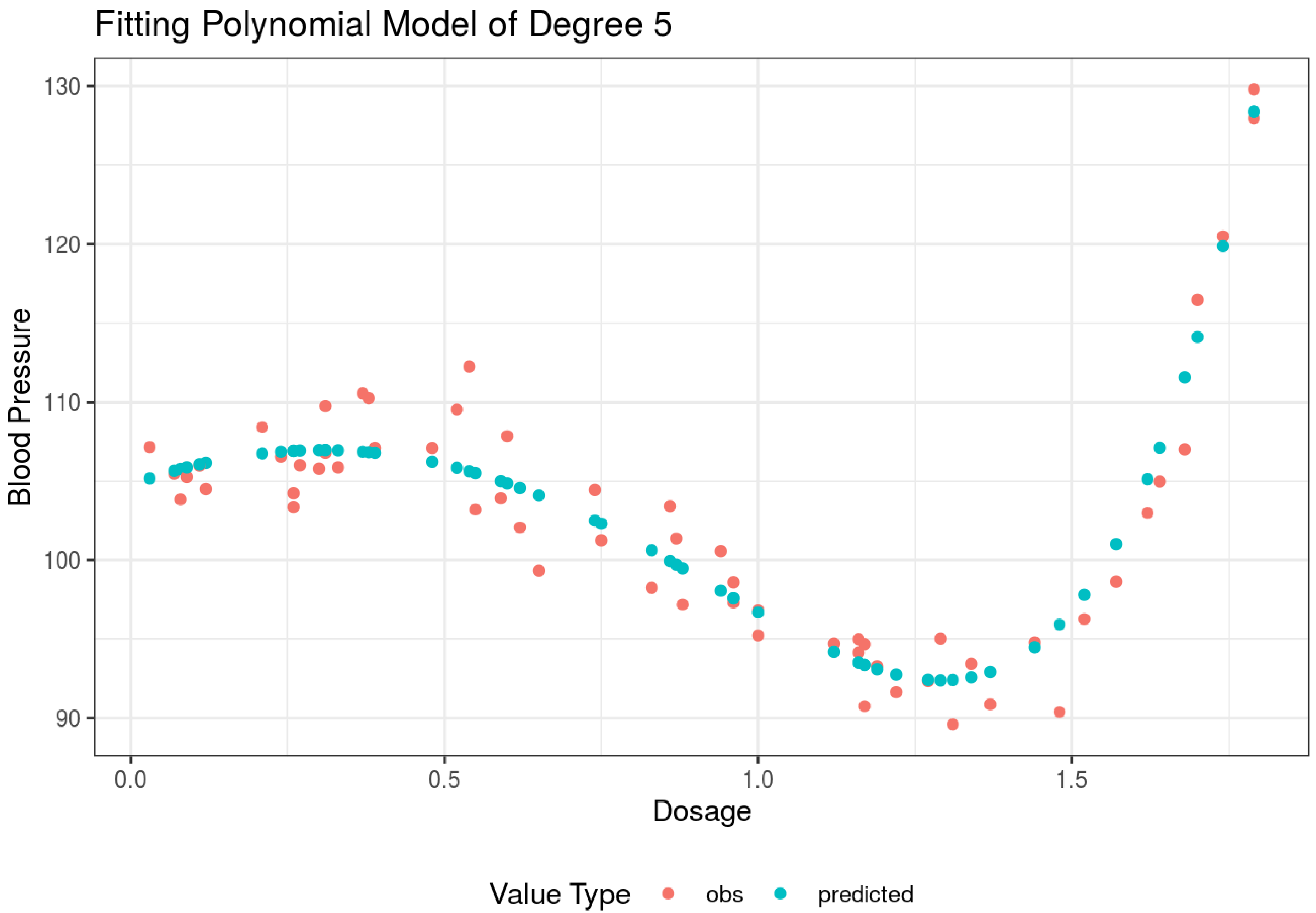


Fitting Polynomial Model of Degree 5
Blood Pressure
130
120
110
100
90
0.0
0.5
1.0
1.5
Dosage
Value Type
● obs
● predicted

l)

Finally, use the model you fit on the training set to obtain a measure of RMSE on the test set. Report this RMSE value in a sentence and also create a graph that shows your model predictions of the test data set.

```
### At the very end we fit the training data and see how our fin
al model performs on the test data that we have been holding out
this whole time:

make_pred_graph_v2(df = dosage_test_set, model = final_model,
                   degree = as.character(cv_res_tib_degr))
```


Fitting Polynomial Model of Degree 5
Blood Pressure
130
120
110
100
90
Dosage
0.0
0.5
1.0
1.5
Value Type
obs
predicted

```
final_rmse <- get_RMSE(dosage_test_set, model = final_model)
```

The RMSE value is 2.3145949.

m)

What do you think about the final model you selected? Does the graph in part (l) look like a good fit to the test data set?

I think the final model is the right choice. The graph looks like a good fit.

n)

OPTIONAL BONUS QUESTION: As we saw in class, one run of k-fold cross-validation can be unstable. For this question, perform ***repeated*** *10-fold cross-validation by obtaining cross-validated RMSE for each degree (1 to 20) polynomial* *model 5 times and then taking their average. Create a graph of these results with an appropriate title – are they different from the results you obtained from one run of 10-fold cross-validation in part (h).*

```
# Perform the same action over varying degrees of polynomials (f
rom 1 to 20) 5 times

#1st run
degree_vals_run1 <- map_dbl(1:20, function(x) perform_k_fold_cv
(df = dosage_training_set, k = 10, degree = x))

cv_res_tib_run1 <- tibble(degree = 1:20, run = 1,
                          cv_rmse = degree_vals_run1)

#2nd run
degree_vals_run2 <- map_dbl(1:20, function(x) perform_k_fold_cv
(df = dosage_training_set, k = 10, degree = x))

cv_res_tib_run2 <- tibble(degree = 1:20, run = 2,
                          cv_rmse = degree_vals_run2)

#3rd run
degree_vals_run3 <- map_dbl(1:20, function(x) perform_k_fold_cv
```

```r
(df = dosage_training_set, k = 10, degree = x))

cv_res_tib_run3 <- tibble(degree = 1:20, run = 3,
                          cv_rmse = degree_vals_run3)


#4th run
degree_vals_run4 <- map_dbl(1:20, function(x) perform_k_fold_cv
(df = dosage_training_set, k = 10, degree = x))

cv_res_tib_run4 <- tibble(degree = 1:20, run = 4,
                          cv_rmse = degree_vals_run4)


#5th run
degree_vals_run5 <- map_dbl(1:20, function(x) perform_k_fold_cv
(df = dosage_training_set, k = 10, degree = x))

cv_res_tib_run5 <- tibble(degree = 1:20, run = 5,
                          cv_rmse = degree_vals_run5)
```

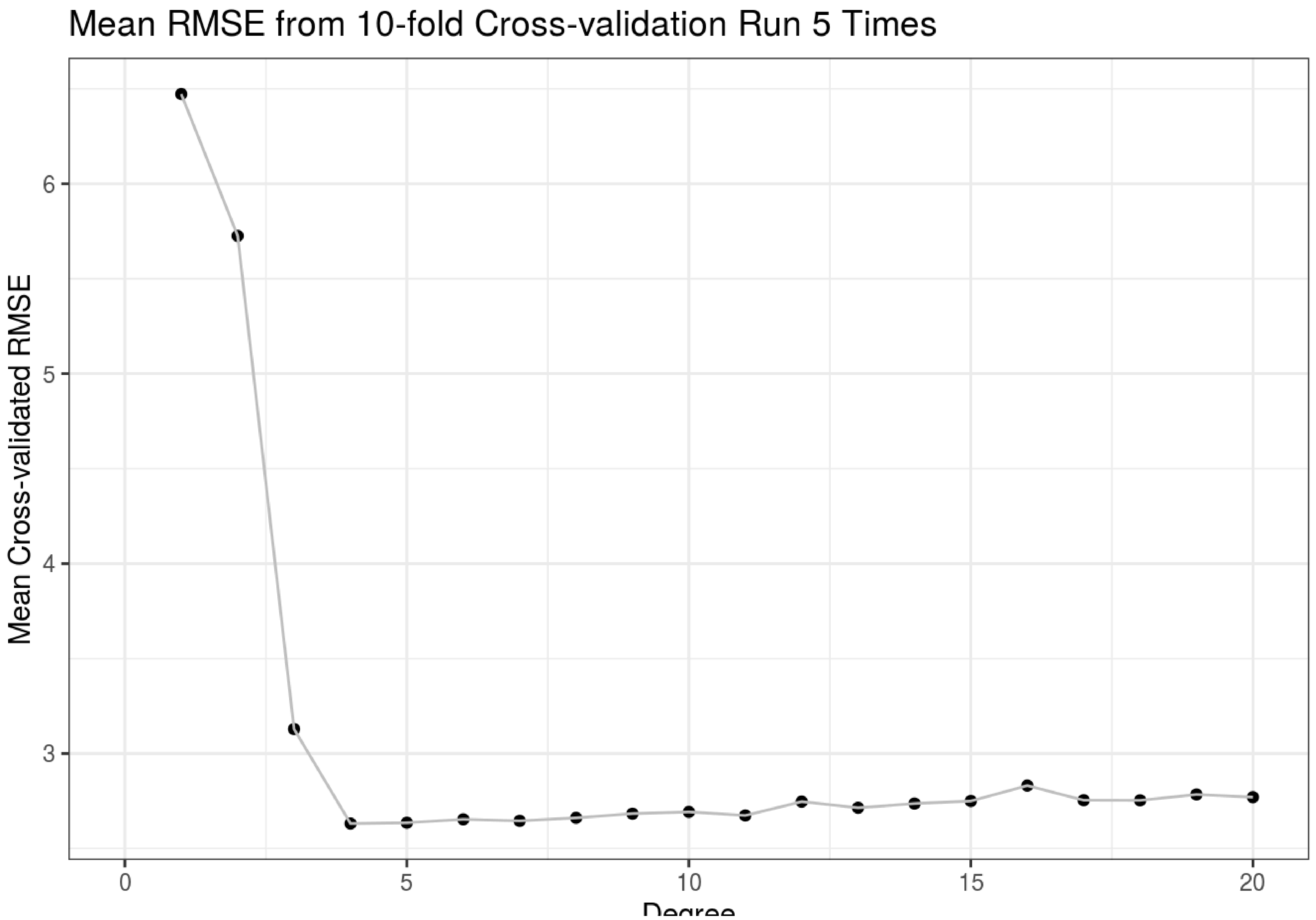
```
#Append df's together
appended_df <- bind_rows(cv_res_tib_run1, cv_res_tib_run2,
                         cv_res_tib_run3, cv_res_tib_run4,
                         cv_res_tib_run5)

#Compute mean RMSE value for each degree
appended_df_mean <- appended_df %>%
  arrange(degree, run) %>%
  group_by(degree) %>%
  summarize(mean_cv_rmse = mean(cv_rmse)) %>%
  arrange(mean_cv_rmse)

kable(appended_df_mean)
```

degree	mean_cv_rmse
4	2.631451
5	2.635510

degree	mean_cv_rmse
7	2.645534
6	2.653522
8	2.661244
11	2.673851
9	2.683506
10	2.692589
13	2.714434
14	2.736642
12	2.746664
15	2.749837

degree	mean_cv_rmse
18	2.753267
17	2.753966
20	2.770207
19	2.784215
16	2.831106
3	3.128781
2	5.725470
1	6.472259

```
#Create graph
ggplot(data = appended_df_mean) +
  geom_point(aes(x = degree, y = mean_cv_rmse)) +
  geom_line(aes(x = degree, y = mean_cv_rmse), color = "grey") +
  theme_bw() +
  scale_x_continuous(limits = c(0, 20)) +
  labs(title = "Mean RMSE from 10-fold Cross-validation Run 5 Ti
mes", x = "Degree", y = "Mean Cross-validated RMSE")
```


Mean RMSE from 10-fold Cross-validation Run 5 Times
Mean Cross-validated RMSE
6
5
4
3
0
5
10
15
20
Degree

Degree

The results appear slightly different from the prior single run of 10-fold cross-validation.

Question 3

In this problem we we will return to the COVID-19 dataset re-fit our linear model on new cases.

a)

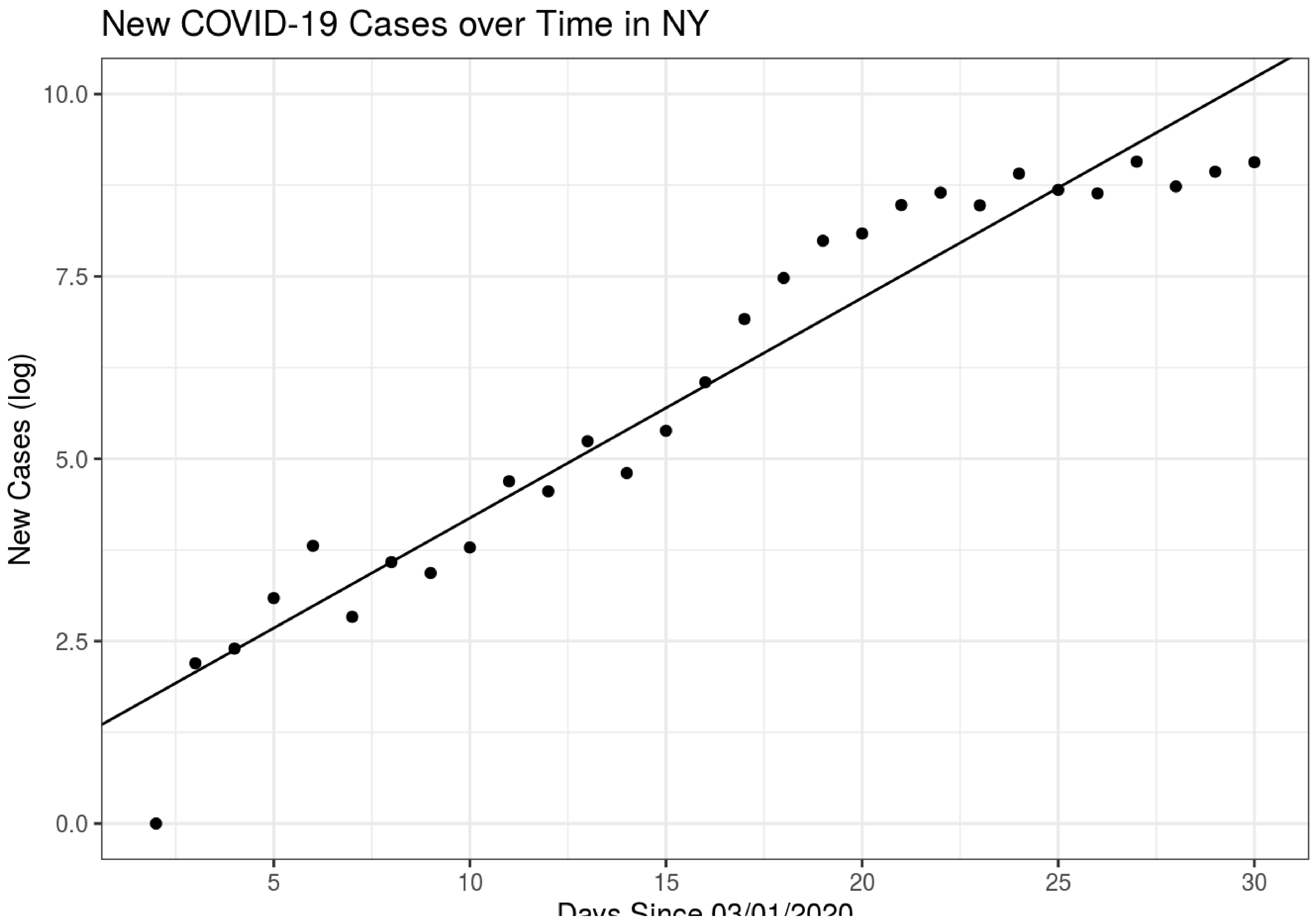
Download the latest US State-level COVID-19 dataset from the New York Times data portal from https://github.com/nytimes/covid-19-data (https://github.com/nytimes/covid-19-data).

```
state_covid_df <- read_csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv")
```

b)

Fit a linear model to log ***new*** *cases of COVID-19 in New York from March 1st, 2020 to March 31st, 2020 using days from March 1st, 2020 as a predictor. Create a graph of your modeled line overlaid on top of the observed data points. Please note that we are modeling log* ***new*** *cases of COVID-19, not log cumulative cases. Make sure you remove days with 0 new cases from the data set so that you do not have -Inf values once you perform the log transformation.*

```r
#Create new data frame
covid_NY_df <- state_covid_df %>%
  filter(state == "New York" & date >= "2020-03-01" &
           date <= "2020-03-31") %>%
  arrange(date) %>%
  mutate(new_cases = cases - lag(cases)) %>%
  mutate(log_new_cases = if_else(is.infinite(log(new_cases)),
                                  NA_real_, log(new_cases)),
         days = date - as.Date("2020-03-01")) %>%
  filter(new_cases != 0 & !is.na(new_cases))

#Create linear model of days -> log_new_cases
covid_NY_lm <- lm(log_new_cases ~ days, data = covid_NY_df)
#Put param estimates into a tibble
covid_NY_lm_tib <- tidy(covid_NY_lm)

#Plot observations and modeled line
ggplot(covid_NY_df) +
  geom_point(aes(x=days, y=log_new_cases)) +
  geom_abline(intercept = covid_NY_lm_tib$estimate[1],
              slope = covid_NY_lm_tib$estimate[2]) +
  theme_bw() +
  scale_x_continuous(breaks = seq(0, 30, by=5)) +
  scale_y_continuous(limits = c(0, 10)) +
  labs(title="New COVID-19 Cases over Time in NY",
       x="Days Since 03/01/2020", y="New Cases (log)")
```

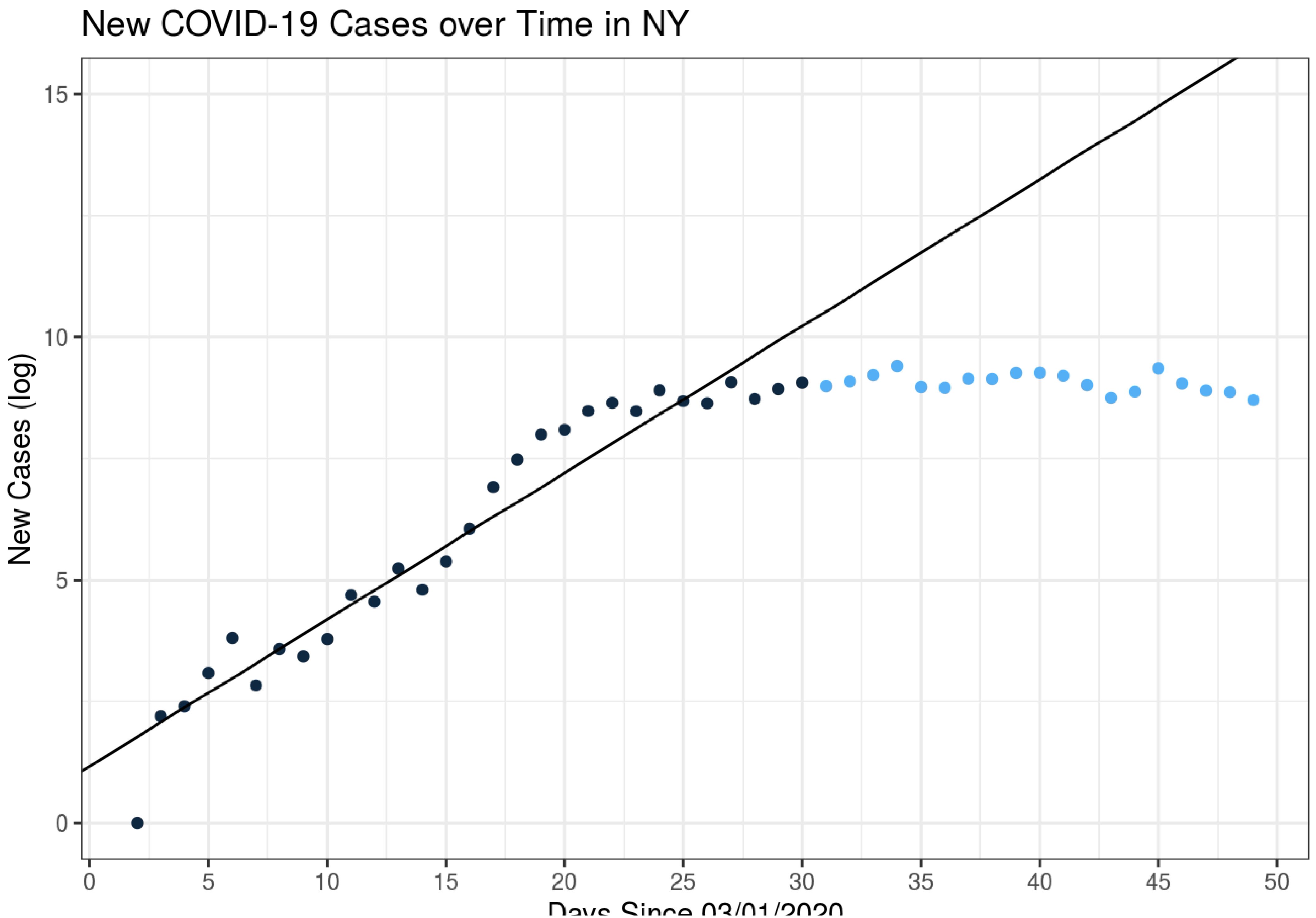

New COVID-19 Cases over Time in NY
New Cases (log)
10.0
7.5
5.0
2.5
0.0
5
10
15
20
25
30
Days Since 03/01/2020

Days Since 03/01/2020

c)

Update your graph in part (b) by including all the available data (after March 1st, 2020) from New York state. This updated graph should include your original regression line and the new data points (those after March 31st, 2020) in a different color.

```r
#Create new data frame
covid_NY_df_extended <- state_covid_df %>%
  filter(state == "New York" & date >= "2020-03-01") %>%
  arrange(date) %>%
  mutate(new_cases = cases - lag(cases)) %>%
  mutate(log_new_cases = if_else(is.infinite(log(new_cases)),
                            NA_real_, log(new_cases)),
         days = date - as.Date("2020-03-01"),
         date_flg = if_else(date > "2020-03-31", 1, 0)) %>%
  filter(new_cases != 0 & !is.na(new_cases))

#Create linear model of days -> log_new_cases
covid_NY_lm_extended <- lm(log_new_cases ~ days,
                           data = covid_NY_df_extended)
#Put param estimates into a tibble
covid_NY_lm_extended_tib <- tidy(covid_NY_lm_extended)

#Plot observations and modeled line
ggplot(covid_NY_df_extended) +
  geom_point(aes(x=days, y=log_new_cases, color=date_flg)) +
  geom_abline(intercept = covid_NY_lm_tib$estimate[1],
              slope = covid_NY_lm_tib$estimate[2]) +
  theme_bw() +
  theme(legend.position = "none") +
  scale_x_continuous(breaks = seq(0, 50, by=5)) +
  scale_y_continuous(limits = c(0, 15)) +
  labs(title="New COVID-19 Cases over Time in NY",
       x="Days Since 03/01/2020", y="New Cases (log)")
```


New COVID-19 Cases over Time in NY
New Cases (log)
15
10
5
0
0
5
10
15
20
25
30
35
40
45
50
Days Since 03/01/2020

Days Since 03/01/2020

d)

Adapt the code from Question 2 to create a new `get_RMSE` *function that will work for this data set. Use it to calculate the RMSE for the linear model fit in part (b) on the entire dataset (all data past March 1st, 2020).*

```r
## Function to get RMSE

get_RMSE_covid <- function(df, model){
  predicted_df <- augment(model, newdata = df) %>%
    select(days, log_new_cases, .fitted)
  rmse <- RMSE(predicted_df$.fitted, predicted_df$log_new_cases)
  return(rmse)
}

rmse_covid_NY <- get_RMSE_covid(covid_NY_df_extended,
                                covid_NY_lm_extended)
```

The RMSE value for the linear model is 1.2850283.

e)

We do not have enough data to use cross-validation to tune a polynomial model, but we can still “eyeball” a polynomial fit. Let’s construct a polynomial regression model (degree = 3) with days from March 1st, 2020 as the predictor and log new cases as the outcome in the full New York State dataset (all data past March 1st, 2020). Use your function from (d) to report the RMSE of this model on the full dataset. We’ll look at this model more next week and see how it performs as new data comes in!

```r
#Create 3rd degree polynomial model
covid_NY_deg3_mod <- lm(log_new_cases ~ poly(days, 3),
                         data = covid_NY_df_extended)

#Put parameter estimates into a tibble
covid_NY_deg3_mod_tib <- tidy(covid_NY_deg3_mod)

#Compute RMSE
rmse_covid_NY_deg3 <- get_RMSE_covid(covid_NY_df_extended,
                                     covid_NY_deg3_mod)
```

The RMSE value of the 3rd degree polynomial model is 0.4696048.