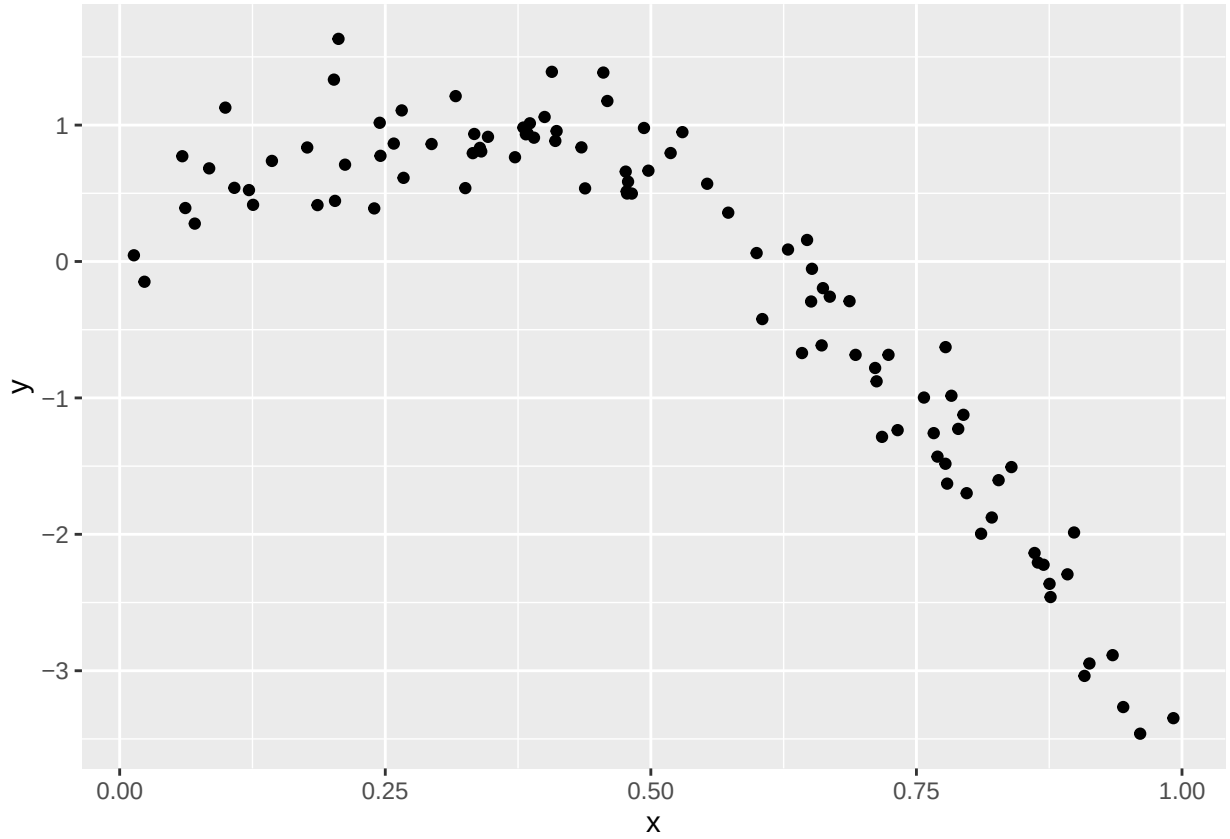
cross_validation

Jasmine Zhang

2023-11-14

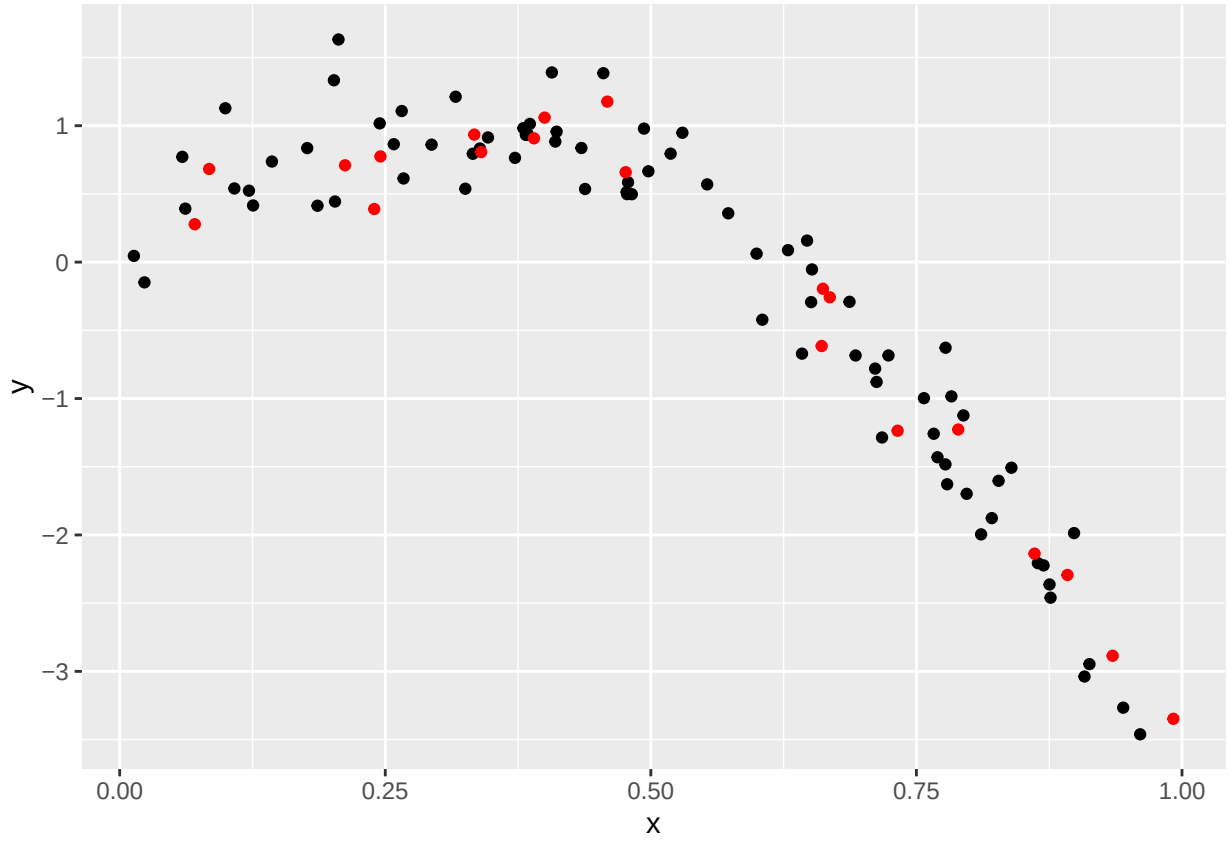
Nonlinear data and CV

```
nonlin_df = tibble(
  id = 1:100,
  x = runif(100, 0, 1),
  y = 1 - 10 * (x - .3) ^ 2 + rnorm(100, 0, .3))

nonlin_df |>
  ggplot(aes(x = x, y = y)) +
  geom_point()
```



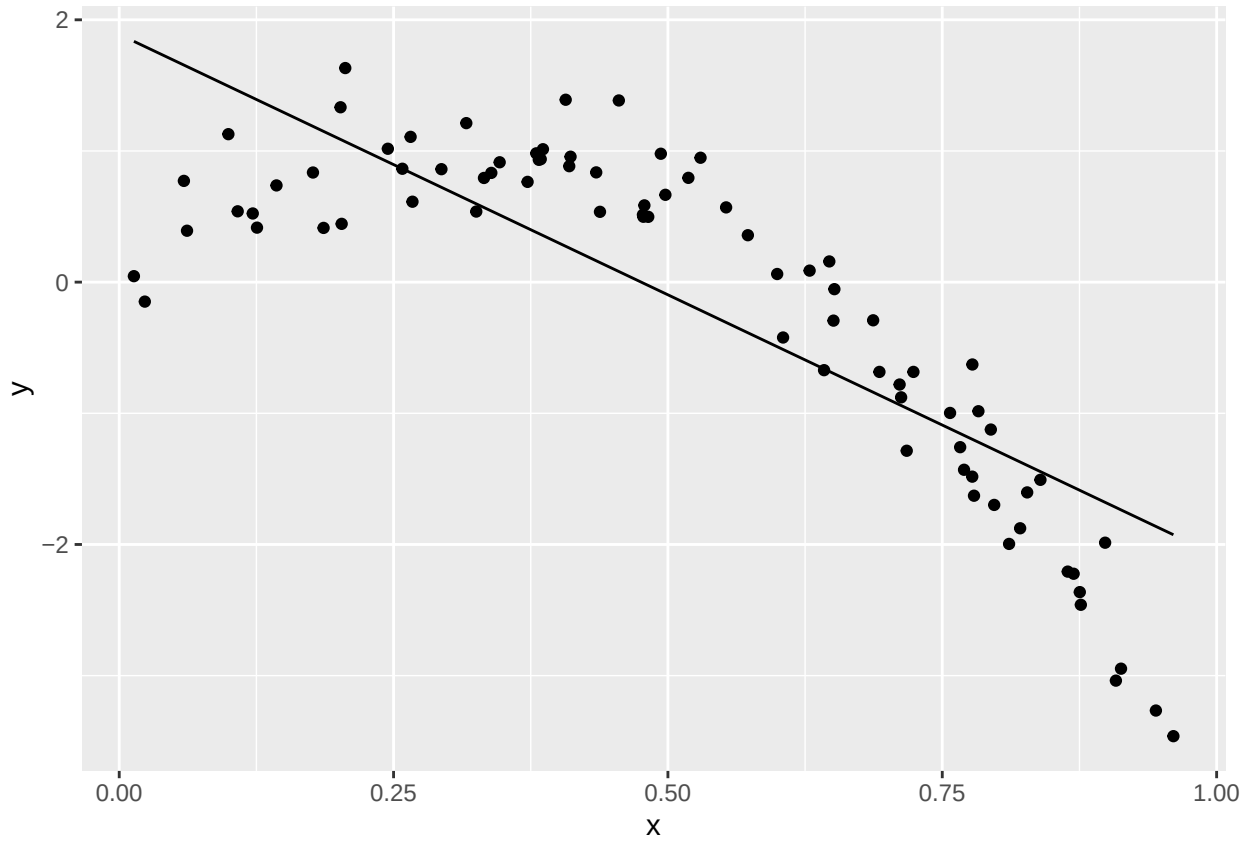
Do the train/test data split: by hand

```
train_df = sample_n(nonlin_df, 80)
test_df = anti_join(nonlin_df, train_df, by = "id") #data points not in train dataset

train_df |> ggplot(aes(x = x, y = y)) + geom_point() +
  geom_point(data = test_df, color = "red")
```

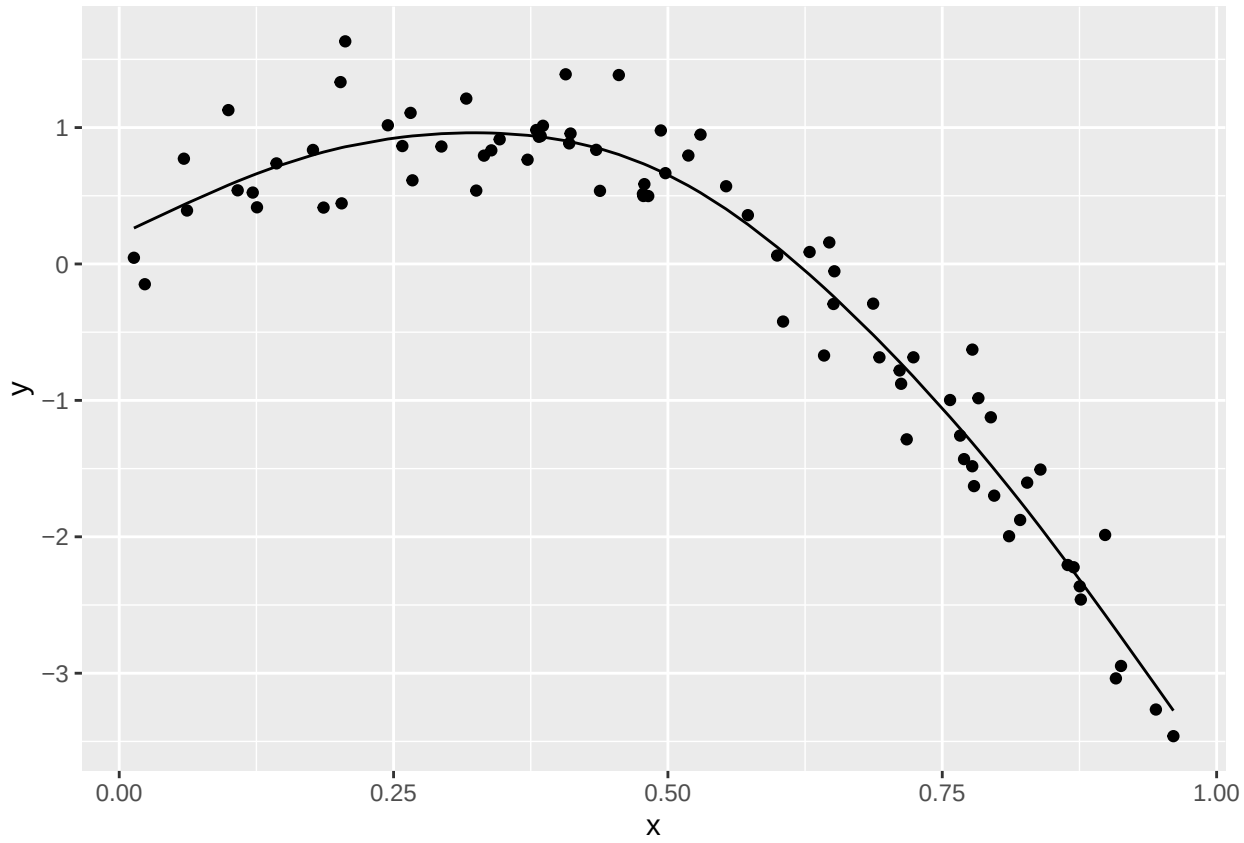



```
#can we fit curve based on black data and see how it predicts the red points
```

```
linear_mod = lm(y~x, data = train_df)
#quick visualization
train_df |>
  modelr::add_predictions(linear_mod) |> #get the fitted values
  ggplot(aes(x = x, y = y)) +
  geom_point() + geom_line(aes(y = pred))
```




```
smooth_mod = mgcv::gam(y ~ s(x), data = train_df)
train_df |>
  modelr::add_predictions(smooth_mod) |> #get the fitted values
  ggplot(aes(x = x, y = y)) +
  geom_point() + geom_line(aes(y = pred))
```




```
wiggly_mod = mgcv::gam(y ~ s(x, k = 30), sp = 10e-6, data = train_df) #wrong fit
```

RMSE on training and testing datasets

```
#see how the model fit is working on the dataset used to fit the model
rmse(linear_mod, train_df)
```

```
## [1] 0.7178747
```

```
rmse(smooth_mod, train_df)
```

```
## [1] 0.2874834
```

```
rmse(wiggly_mod, train_df) #wiggly model seems to be doing the best
```

```
## [1] 0.2498309
```

```
#see how the model fit on the testing data
rmse(linear_mod, test_df)
```

```
## [1] 0.7052956
```

```
rmse(smooth_mod, test_df)
```

```
## [1] 0.2221774
```

```
rmse(wiggly_mod, test_df)
```

```
## [1] 0.289051
```

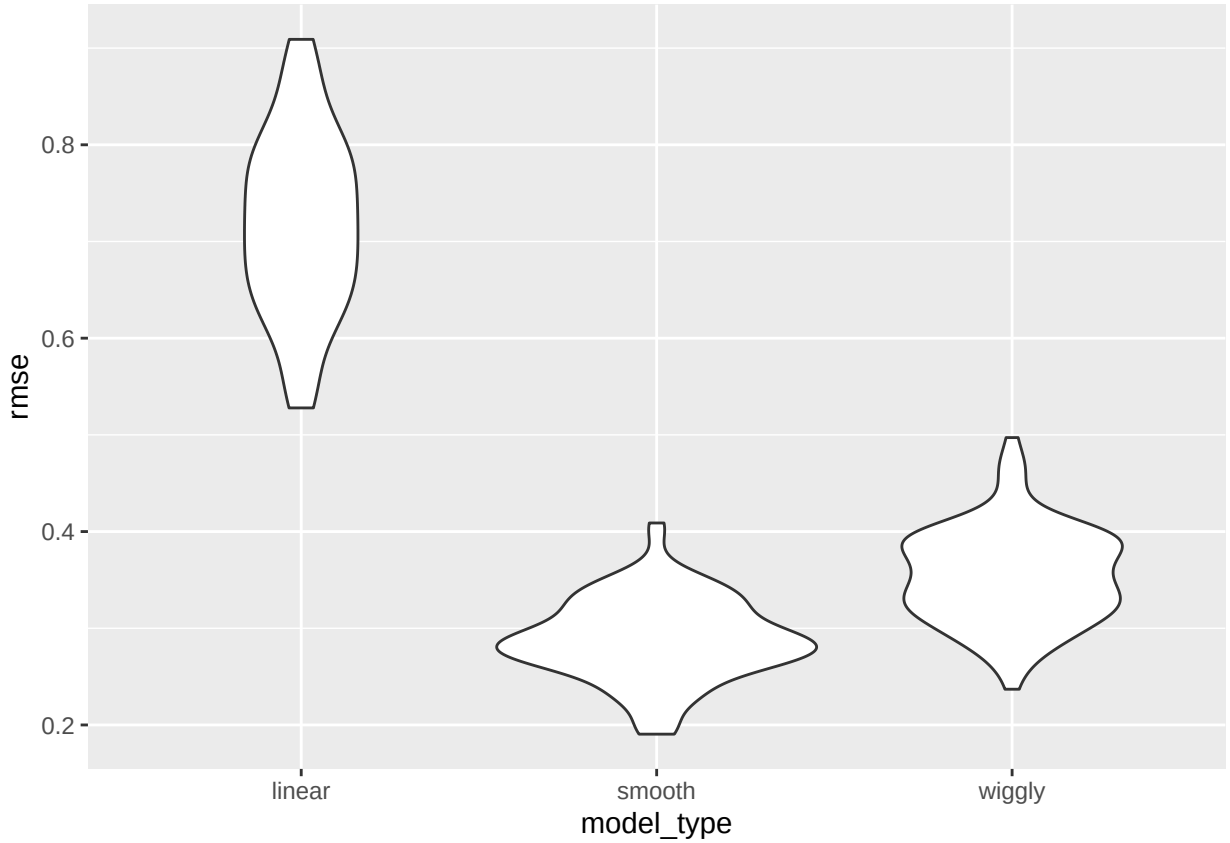
Use modelr for CV

```
cv_df = nonlin_df |>
  crossv_mc(n = 100) |>  #partition in train and test datasets for n times
  mutate(train = map(train, as_tibble),
         test = map(test, as_tibble)) #put the partitioned data into tibble for viewing
#cv_df |> pull(train) |> nth(2) |> as_tibble()
```

Apply each model to all training datasets, and evaluate on all testing datasets

```
cv_results = cv_df |>
  mutate(linear_mod = map(train, \(df)lm(y~x, data = df)),
         smooth_mod  = map(train, \(df) mgcv::gam(y ~ s(x), data = df)),
         wiggly_mod  = map(train, \(df) gam(y ~ s(x, k = 30), sp = 10e-6, data = df))) |>

#lin_mod_funtion = function (df) {lm(y~x, data = df)} shorthand function
  mutate(rmse_linear = map2_dbl(linear_mod, test, \(mod,df) rmse(mod, df)),
         rmse_smooth = map2_dbl(smooth_mod, test, \(mod, df) rmse(model = mod, data = df)),
         rmse_wiggly = map2_dbl(wiggly_mod, test, \(mod, df) rmse(model = mod, data = df))) #two columns
```

```
cv_results |>
  select(starts_with("rmse")) |>
  pivot_longer(
    everything(),
    names_to = "model_type",
    values_to = "rmse",
    names_prefix = "rmse_") |>
  ggplot(aes(x = model_type, y = rmse)) + geom_violin()
```




```
#group_by(model_type) |>
#summarize(m_rmse = mean(rmse))
```