Modelos para diferentes cantones

Jimena Murillo

2022-06-06

```r
library(keras) # for deep learning
library(tidyverse) # general utility functions
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret) # machine learning utility functions
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tibble)
library(readr)
library(ggplot2)
library(tensorflow)
```

```
##
## Attaching package: 'tensorflow'
```

```
## The following object is masked from 'package:caret':
##
##     train
```

```r
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute
```

Datos

```r
load("C:/Users/usuario1/Desktop/CIMPA/Github_CIMPA/PRACTICA_CIMPA/base_cantones.RData")


basecanton = basecanton  %>%

  dplyr::select(Canton, Year,Month,Nino12SSTA, Nino3SSTA, Nino4SSTA,Nino34SSTA,Nino34SSTA1, Nino34SSTA2

  arrange(Canton,Year,Month) %>% ungroup() %>% mutate(Month=as.numeric(Month))


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

basecanton2 = basecanton %>% group_by(basecanton$Canton) %>%
  mutate_if(is.numeric, normalize)
```

```
## `mutate_if()` ignored the following grouping variables:
## * Column `basecanton$Canton`
```

```r
basecanton2 = basecanton2[,-35]


#Train y test

data_train = as.data.frame(basecanton2) %>% filter(Year < 1)#PARA ENTRENAR HASTA 2018
data_test = as.data.frame(basecanton2) %>% filter(Year >= 1)

X_train = data_train[,-ncol(data_train)]
y_train = as.data.frame(data_train[,c("Canton","RR")])

X_test = as.data.frame(data_test[,-ncol(data_test)])
y_test = as.data.frame(data_test[,c("Canton","RR")])

#Almacen de eval de los modelos


NRMSE = NULL
NIS_95 = NULL
```

Arquitectura y programación del modelo

Generar un Wrapper para el learning dropout

```r
# R6 wrapper class, a subclass of KerasWrapper
ConcreteDropout <- R6::R6Class("ConcreteDropout",

  inherit = KerasWrapper,

  public = list(
    weight_regularizer = NULL,
    dropout_regularizer = NULL,
    init_min = NULL,
    init_max = NULL,
    is_mc_dropout = NULL,
    supports_masking = TRUE,
    p_logit = NULL,
    p = NULL,

    initialize = function(weight_regularizer,
                          dropout_regularizer,
                          init_min,
                          init_max,
                          is_mc_dropout) {
      self$weight_regularizer <- weight_regularizer
      self$dropout_regularizer <- dropout_regularizer
      self$is_mc_dropout <- is_mc_dropout
      self$init_min <- k_log(init_min) - k_log(1 - init_min)
      self$init_max <- k_log(init_max) - k_log(1 - init_max)
    },

    build = function(input_shape) {
      super$build(input_shape)

      self$p_logit <- super$add_weight(
        name = "p_logit",
        shape = shape(1),
        initializer = initializer_random_uniform(self$init_min, self$init_max),
        trainable = TRUE
      )

      self$p <- k_sigmoid(self$p_logit)

      input_dim <- input_shape[[2]]

      weight <- private$py_wrapper$layer$kernel

      kernel_regularizer <- self$weight_regularizer *
                            k_sum(k_square(weight)) /
                            (1 - self$p)

      dropout_regularizer <- self$p * k_log(self$p)
      dropout_regularizer <- dropout_regularizer +
```

```
                              (1 - self$p) * k_log(1 - self$p)
      dropout_regularizer <- dropout_regularizer *
                              self$dropout_regularizer *
                              k_cast(input_dim, k_floatx())

      regularizer <- k_sum(kernel_regularizer + dropout_regularizer)
      super$add_loss(regularizer)
    },

    concrete_dropout = function(x) {
      eps <- k_cast_to_floatx(k_epsilon())
      temp <- 0.1

      unif_noise <- k_random_uniform(shape = k_shape(x))

      drop_prob <- k_log(self$p + eps) -
                   k_log(1 - self$p + eps) +
                   k_log(unif_noise + eps) -
                   k_log(1 - unif_noise + eps)
      drop_prob <- k_sigmoid(drop_prob / temp)

      random_tensor <- 1 - drop_prob

      retain_prob <- 1 - self$p
      x <- x * random_tensor
      x <- x / retain_prob
      x
    },

    call = function(x, mask = NULL, training = NULL) {
      if (self$is_mc_dropout) {
        super$call(self$concrete_dropout(x))
      } else {
        k_in_train_phase(
          function()
            super$call(self$concrete_dropout(x)),
          super$call(x),
          training = training
        )
      }
    }
  )
)

# function for instantiating custom wrapper
layer_concrete_dropout <- function(object,
                                   layer,
                                   weight_regularizer = 1e-6,
                                   dropout_regularizer = 1e-5,
                                   init_min = 0.1,
                                   init_max = 0.1,
                                   is_mc_dropout = TRUE,
                                   name = NULL,
```

```
                                    trainable = TRUE) {
  create_wrapper(ConcreteDropout, object, list(
    layer = layer,
    weight_regularizer = weight_regularizer,
    dropout_regularizer = dropout_regularizer,
    init_min = init_min,
    init_max = init_max,
    is_mc_dropout = is_mc_dropout,
    name = name,
    trainable = trainable
  ))
}

# sample size (training data)
n_train <- 232
# sample size (validation data)
n_val <- 3
# prior length-scale
l <- 1e-4
# initial value for weight regularizer
wd <- l^2/232
# initial value for dropout regularizer
dd <- 2/3
```

Arquitectura del modelo:

```
  input_dim <- 32
  output_dim <- 1

  input <- layer_input(shape = input_dim)
```

```
## Loaded Tensorflow version 2.8.0
```

```
  output <- input %>% layer_concrete_dropout(
    layer = layer_dense(units = 100, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 50, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 50, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 50, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
```

```
    layer = layer_dense(units = 25, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 25, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 25, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 12, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 12, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 6, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    ) %>% layer_concrete_dropout(
    layer = layer_dense(units = 6, activation = "relu"),
    weight_regularizer = wd,
    dropout_regularizer = dd
    )
```

Entrenamiento de modelos para cada cantón

Entrenar al modelo y predecir

```
Cantones = unique(basecanton$Canton)
Evaluacion3 = matrix(NA, ncol = 2, nrow = length(Cantones))
plot_list3 = list()

plot_list_c = list()
Evaluacion_c = matrix(NA, ncol = 2, nrow = length(Cantones))

for (i in 1:length(Cantones)) {

  X_trainc = X_train %>% filter(Canton == Cantones[i])
  X_trainc = as.matrix(X_trainc[,-1])
  y_trainc = y_train %>% filter(Canton == Cantones[i])
  y_trainc = as.matrix(y_trainc[,-1])

  X_testc = X_test %>% filter(Canton == Cantones[i])
  X_testc = as.matrix(X_testc[,-1])
  y_testc = y_test %>% filter(Canton == Cantones[i])
  y_testc = as.matrix(y_testc[,-1])
```

```r
base = as.data.frame(basecanton %>% filter(Canton == Cantones[i]) %>% dplyr::select(RR))



## Output del Modelo


mean <- output %>% layer_concrete_dropout(
  layer = layer_dense(units = output_dim),
  weight_regularizer = wd,
  dropout_regularizer = dd
)

log_var <- output %>% layer_concrete_dropout(
  layer_dense(units = output_dim),
  weight_regularizer = wd,
  dropout_regularizer = dd
)

output <- layer_concatenate(list(mean, log_var))

model <- keras_model(input, output)


model %>% compile(
optimizer = "adam",
loss = "mse",
metrics = "mae")


print(Cantones[i])

history <- model %>% fit(
  X_trainc,
  y_trainc,
  epochs = 50,
  batch_size = 18,
  validation_split = 0.1
)


## MonteCarlo sampling 3 meses


denorm <- function(x) {
  return (x*(max(base$RR) - min(base$RR))+min(base$RR))
}

num_MC_samples <- 100

MC_samples <- array(0, dim = c(num_MC_samples, nrow(X_testc), 2 * output_dim))
for (k in 1:num_MC_samples) {
  MC_samples[k, , ] <- denorm((model %>% predict(X_testc)))
```

```r
}


## Generar intervalo de confianza 3 meses


# First, we determine the predictive mean as an average of the MC samples' mean output:

# the means are in the first output column
means = NULL
means <- MC_samples[, , 1:output_dim]

# average over the MC samples
predictive_mean <- apply(means, 2, mean)



# To calculate epistemic uncertainty, we again use the mean output, but this time we're interested in

epistemic_uncertainty <- apply(means, 2, var)



# Then aleatoric uncertainty is the average over the MC samples of the variance output. 1 .

logvar = NULL
logvar <- MC_samples[, , (output_dim + 1):(output_dim * 2)]
aleatoric_uncertainty <- exp(colMeans(logvar))


# Note how this procedure gives us uncertainty estimates individually for every prediction. How do they

#Obtener RR originales

y_testc = denorm(y_testc)




df <- data.frame(
  x = 1:nrow(X_testc),
  y_testc = y_testc,
  y_pred = predictive_mean,
  e_u_lower = predictive_mean - sqrt(epistemic_uncertainty),
  e_u_upper = predictive_mean + sqrt(epistemic_uncertainty),
  a_u_lower = predictive_mean - sqrt(aleatoric_uncertainty),
  a_u_upper = predictive_mean + sqrt(aleatoric_uncertainty),
  u_overall_lower = predictive_mean -
                  sqrt(epistemic_uncertainty) -
                  sqrt(aleatoric_uncertainty),
  u_overall_upper = predictive_mean +
                  sqrt(epistemic_uncertainty) +
                  sqrt(aleatoric_uncertainty)
```

```
)

#Here, first, is epistemic uncertainty, with shaded bands indicating one standard deviation above resp.

  metricas <- function(df){
    NRMSE <- mean((df$y_pred-y_testc)^2)/mean(y_testc)
    NIS_95 <- mean((df$e_u_upper-df$e_u_lower)+
                   (2/0.05)*(df$e_u_lower-y_testc)*(y_testc<df$e_u_lower)+
                   (2/0.05)*(y_testc-df$e_u_upper)*(y_testc>df$e_u_upper))/mean(y_testc)
    return(data.frame(NRMSE,NIS_95))
  }

  Evaluacion3[i,1:2] = as.numeric(metricas(df))

  title = paste("Tendencia", Cantones[i], sep = " ")

  p = ggplot(df, aes(x, y_pred)) +
    geom_line(colour = "blue") +
    geom_line( aes(x, y = y_testc, colour = "red"))+
    geom_ribbon(aes(ymin = e_u_lower, ymax = e_u_upper), alpha = 0.3)+
    ggtitle(title)

  plot_list3[[i]] = p

  #### VALORES APROXIMADOS ####

  X_test_all = basecanton2 %>% filter(Canton == Cantones[i])
  X_test_all = as.matrix(X_test_all[,-c(1,33)])
  y_test_all = basecanton %>% filter(Canton == Cantones[i])
  y_test_all = as.matrix(y_test_all$RR)

  denorm <- function(x) {
    return (x*(max(base$RR) - min(base$RR))+min(base$RR))
  }

  num_MC_samples <- 100

  MC_samples <- array(0, dim = c(num_MC_samples, 235, 2 * output_dim))
  for (k in 1:num_MC_samples) {
    MC_samples[k, , ] <- denorm((model %>% predict(X_test_all)))
  }
```

```r
## Generar intervalo de confianza 3 meses


# First, we determine the predictive mean as an average of the MC samples' mean output:

# the means are in the first output column
means = NULL
means <- MC_samples[, , 1:output_dim]

# average over the MC samples
predictive_mean <- apply(means, 2, mean)



# To calculate epistemic uncertainty, we again use the mean output, but this time we're interested in

epistemic_uncertainty <- apply(means, 2, var)



# Then aleatoric uncertainty is the average over the MC samples of the variance output. 1 .

logvar = NULL
logvar <- MC_samples[, , (output_dim + 1):(output_dim * 2)]
aleatoric_uncertainty <- exp(colMeans(logvar))


# Note how this procedure gives us uncertainty estimates individually for every prediction. How do they

#Obtener RR originales



df <- data.frame(
  x = 1:nrow(X_test_all),
  y_test_all = y_test_all,
  y_pred = predictive_mean,
  e_u_lower = predictive_mean - sqrt(epistemic_uncertainty),
  e_u_upper = predictive_mean + sqrt(epistemic_uncertainty),
  a_u_lower = predictive_mean - sqrt(aleatoric_uncertainty),
  a_u_upper = predictive_mean + sqrt(aleatoric_uncertainty),
  u_overall_lower = predictive_mean -
                  sqrt(epistemic_uncertainty) -
                  sqrt(aleatoric_uncertainty),
  u_overall_upper = predictive_mean +
                  sqrt(epistemic_uncertainty) +
                  sqrt(aleatoric_uncertainty)
)
```

```
#Here, first, is epistemic uncertainty, with shaded bands indicating one standard deviation above resp.



  metricas <- function(df){
    NRMSE <- mean((df$y_pred-y_test_all)^2)/mean(y_test_all)
    NIS_95 <- mean((df$e_u_upper-df$e_u_lower)+
                   (2/0.05)*(df$e_u_lower-y_test_all)*(y_test_all<df$e_u_lower)+
                   (2/0.05)*(y_test_all-df$e_u_upper)*(y_test_all>df$e_u_upper))/mean(y_test_all)
    return(data.frame(NRMSE,NIS_95))
  }



  Evaluacion_c[i,1:2] = as.numeric(metricas(df))

  title = paste("Tendencia", Cantones[i], sep = " ")



  p1 = ggplot(df, aes(x, y_pred)) +
    geom_line(colour = "blue") +
    geom_line( aes(x, y = y_test_all, colour = "red"))+
    geom_ribbon(aes(ymin = e_u_lower, ymax = e_u_upper), alpha = 0.3)+
    ggtitle(title)

  plot_list_c[[i]] = p1



}
```

```
## [1] "Alajuela"
## [1] "Alajuelita"
## [1] "Atenas"
## [1] "Cañas"
## [1] "Carrillo"
## [1] "Corredores"
## [1] "Desamparados"
## [1] "Esparza"
## [1] "Garabito"
## [1] "Golfito"
## [1] "Guacimo"
## [1] "La Cruz"
## [1] "Liberia"
## [1] "Limon"
## [1] "Matina"
## [1] "Montes de Oro"
## [1] "Nicoya"
## [1] "Orotina"
## [1] "Osa"
## [1] "Parrita"
## [1] "Perez Zeledón"
```

```
## [1] "Pococí"
## [1] "Puntarenas"
## [1] "Quepos"
## [1] "San Jose"
## [1] "Santa Ana"
## [1] "SantaCruz"
## [1] "Sarapiquí"
## [1] "Siquirres"
## [1] "Talamanca"
## [1] "Turrialba"
## [1] "Upala"
```

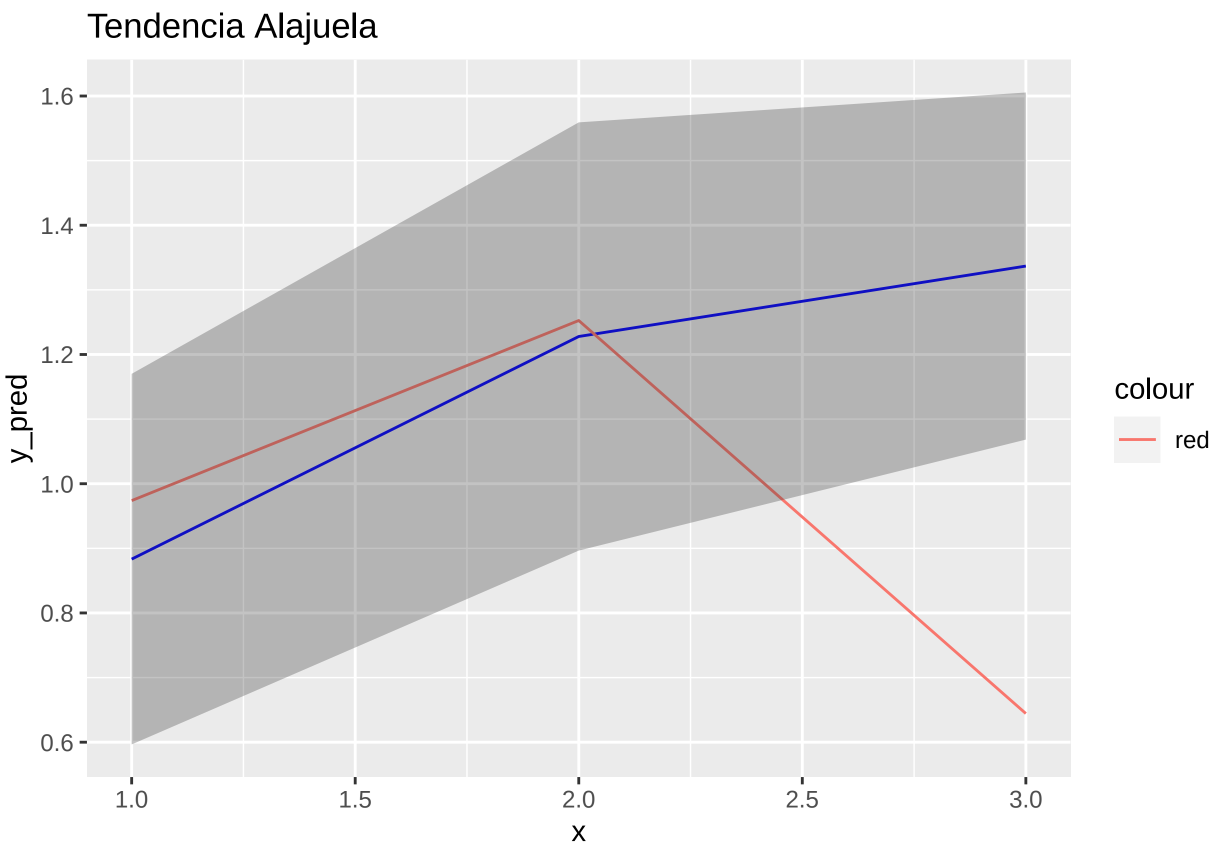
Resultados prediccion de últimos 3 meses

```
Evaluacion3
```

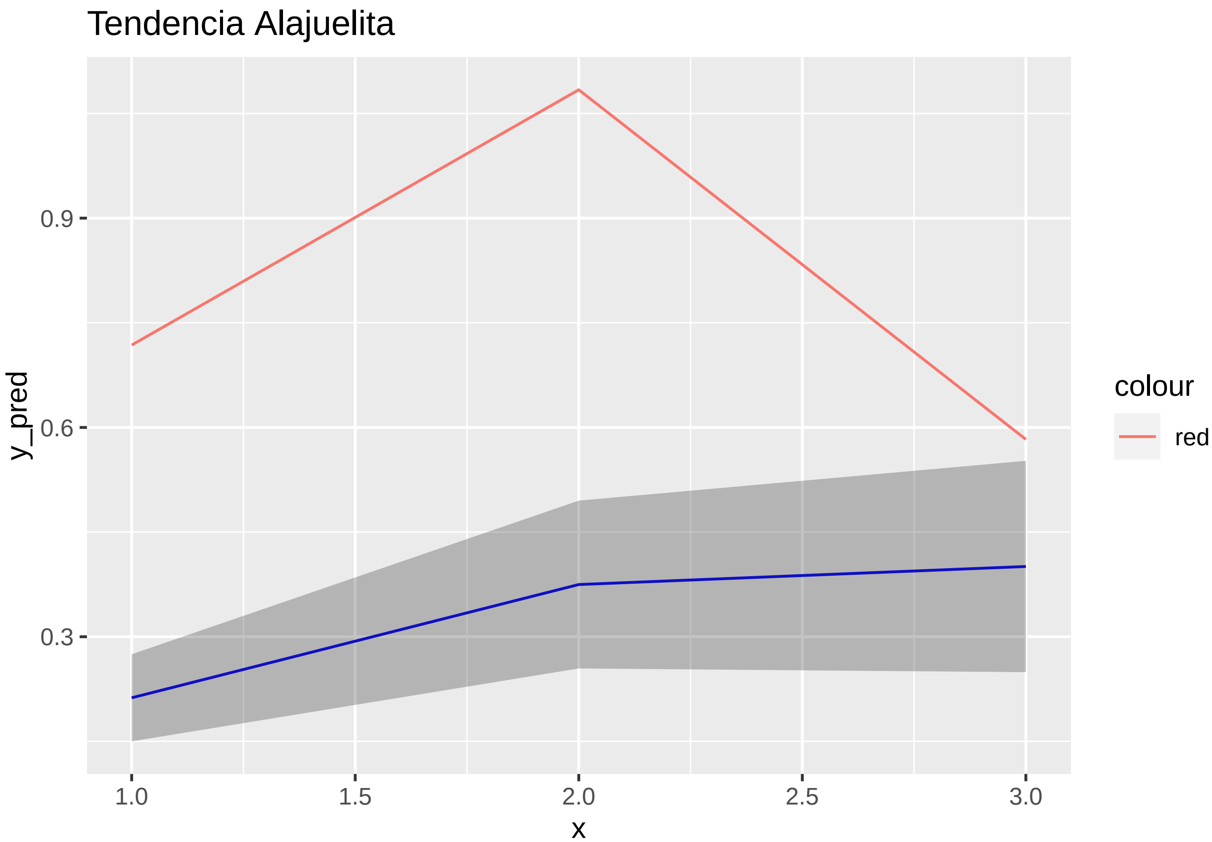
```
##              [,1]       [,2]
##  [1,]  0.17011932   6.524133
##  [2,]  0.33199908  18.106543
##  [3,]  7.22101886  31.360214
##  [4,]  3.41410027  35.612286
##  [5,]  1.20446437  12.706318
##  [6,]  0.81161020  11.955464
##  [7,]  0.01211414   2.036241
##  [8,]  2.93067384  34.283304
##  [9,] 93.41126697 605.617233
## [10,]  2.38674048  65.836939
## [11,]  1.59495615   7.006489
## [12,]  7.97355052 176.684238
## [13,] 14.17128072 241.864840
## [14,]  1.05081536   7.988876
## [15,]  6.40902581  78.147739
## [16,] 28.16225601 300.766377
## [17,] 13.81792928 262.165734
## [18,]  0.99043378   9.697624
## [19,]  1.14230916  14.373638
## [20,] 45.79385218 260.683914
## [21,]  1.84331880  31.651908
## [22,]  1.05537546  19.384706
## [23,]  2.53953563  24.922700
## [24,] 48.88919669 325.151136
## [25,]  0.03141007   8.964273
## [26,]  0.59613111  54.446502
## [27,] 40.55697792 313.853031
## [28,]  0.47517315   9.351463
## [29,]  0.30451218   4.116998
## [30,] 14.87217978  34.892968
## [31,]  1.06090303  24.729816
## [32,]         Inf        Inf
```

```
plot_list3
```

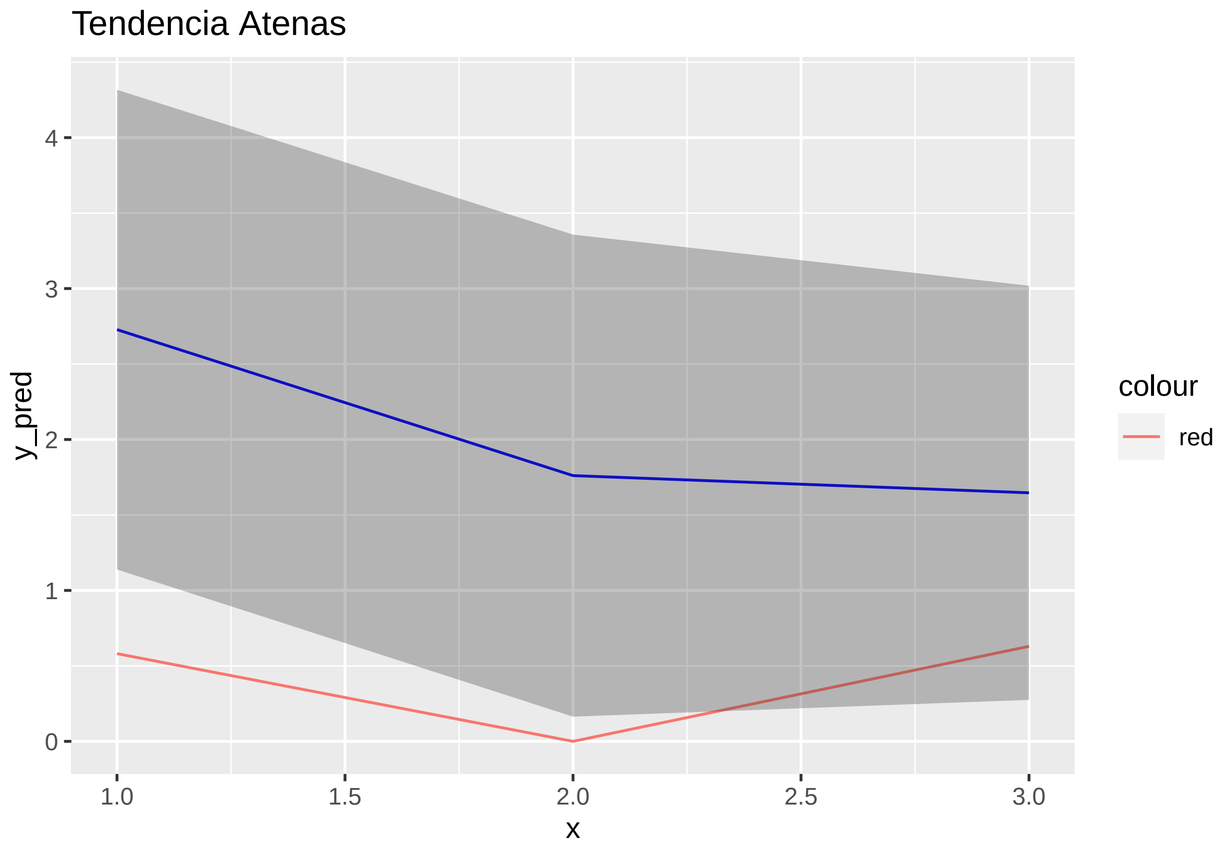
```
## [[1]]
```



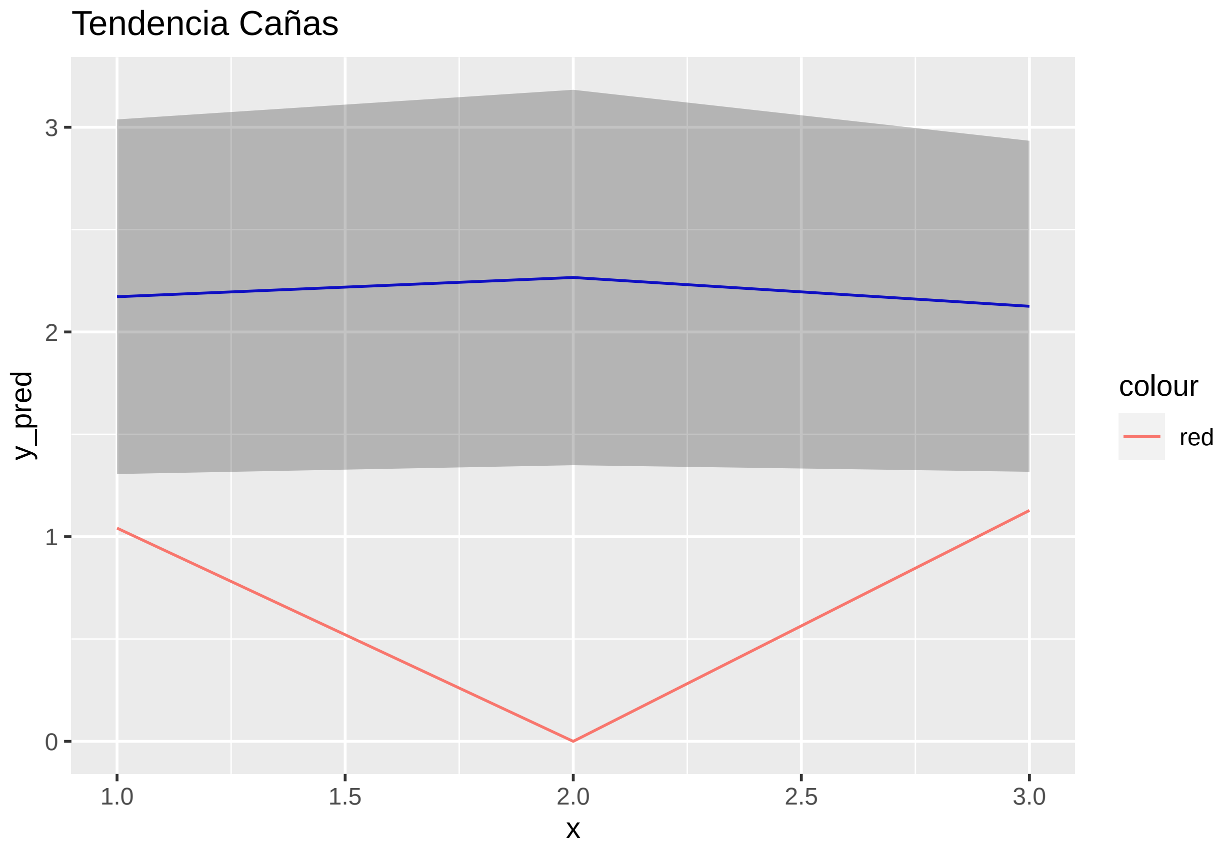

```
##
## [[2]]
```


Tendencia Alajuelita
0.9
0.6
0.3
y_pred
1.0
1.5
2.0
2.5
3.0
x
colour
red

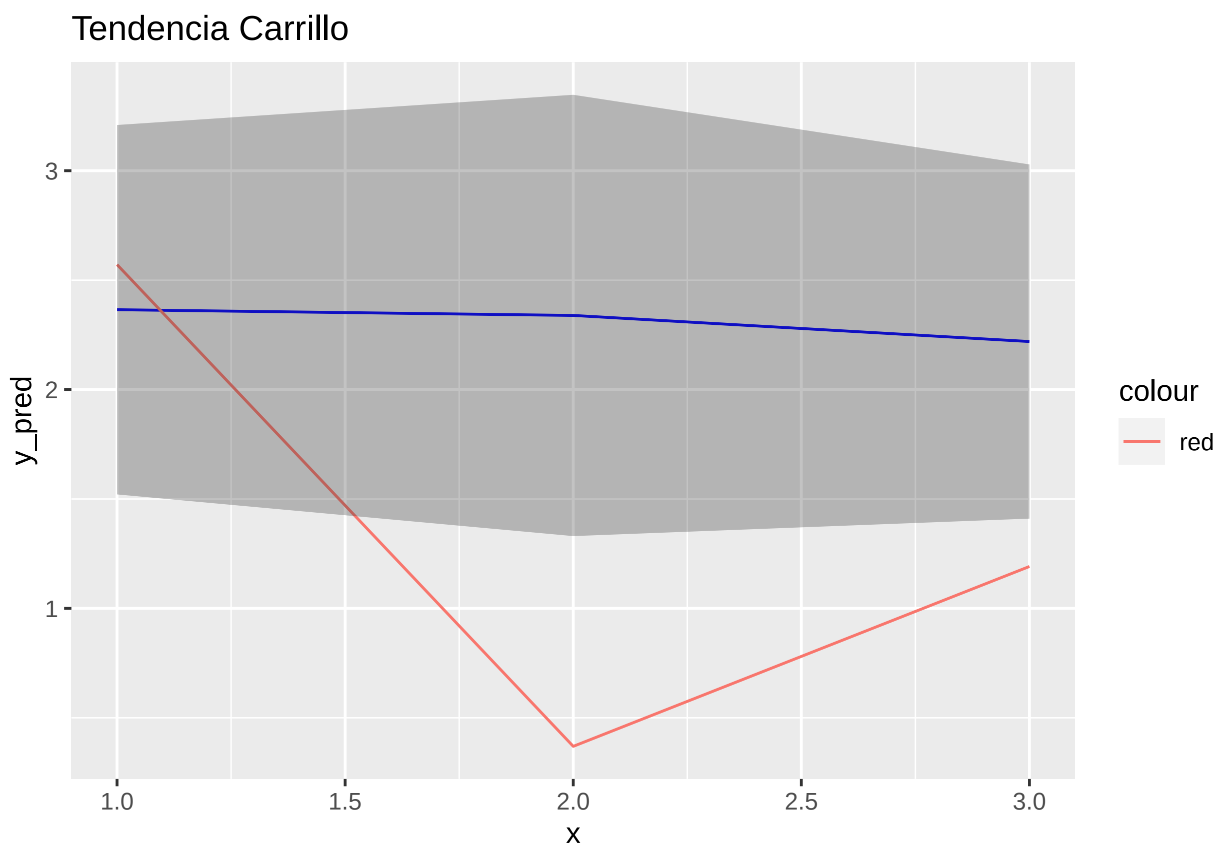
```
##
## [[3]]
```



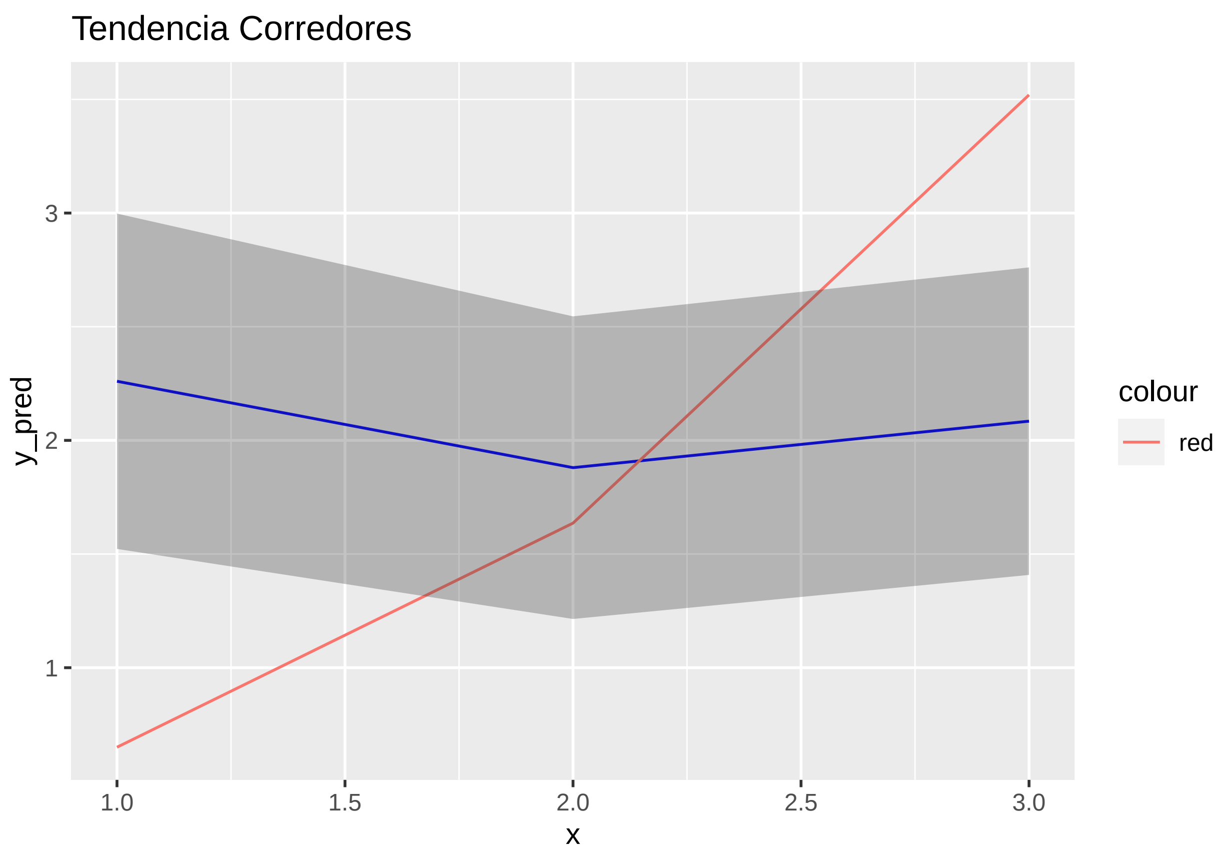

```
##
## [[4]]
```



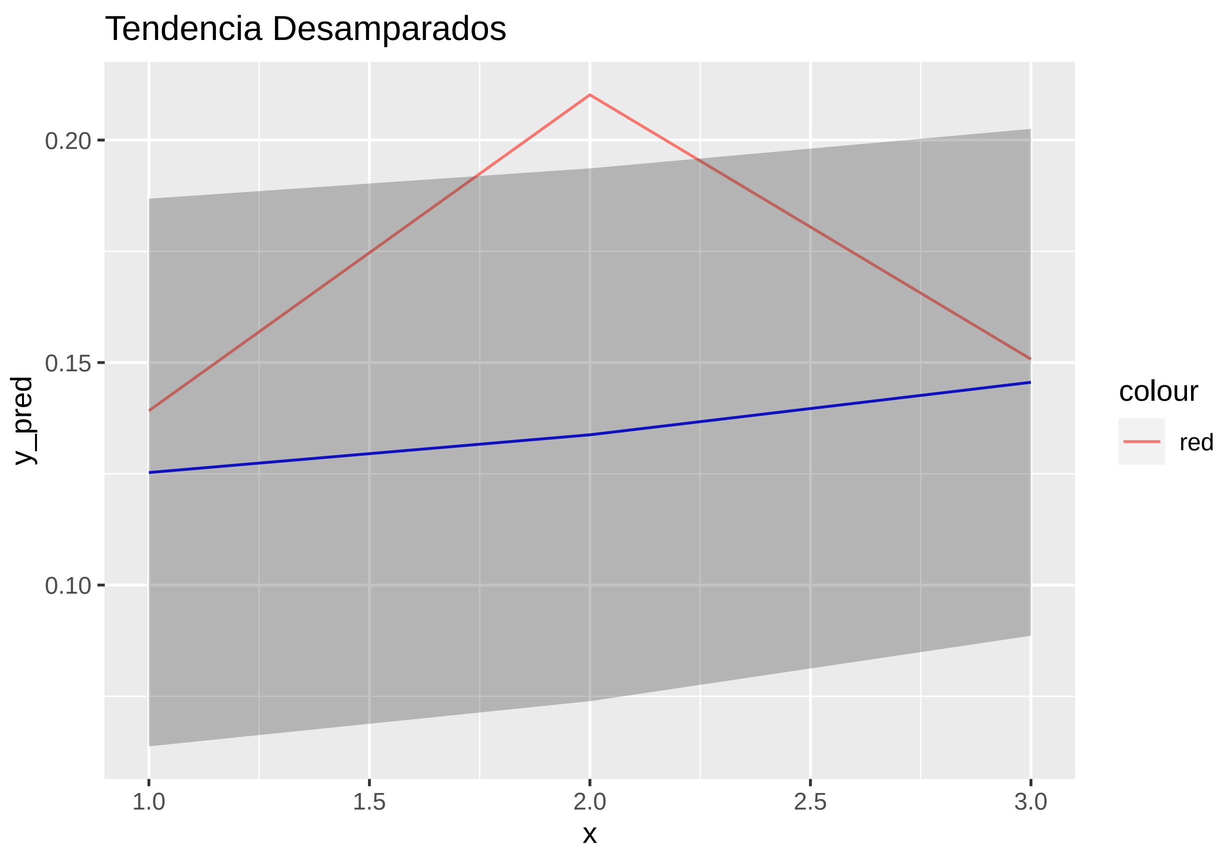

```
##
## [[5]]
```



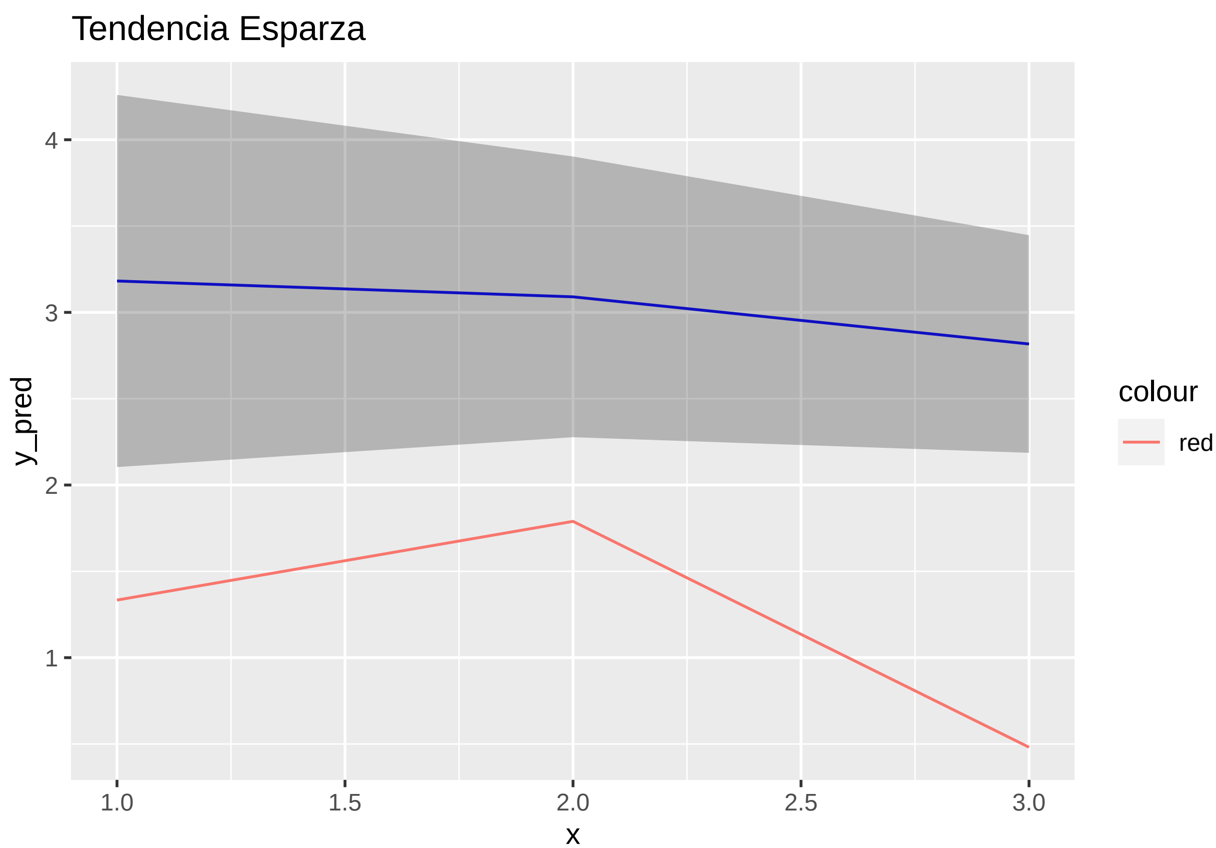

```
##
## [[6]]
```



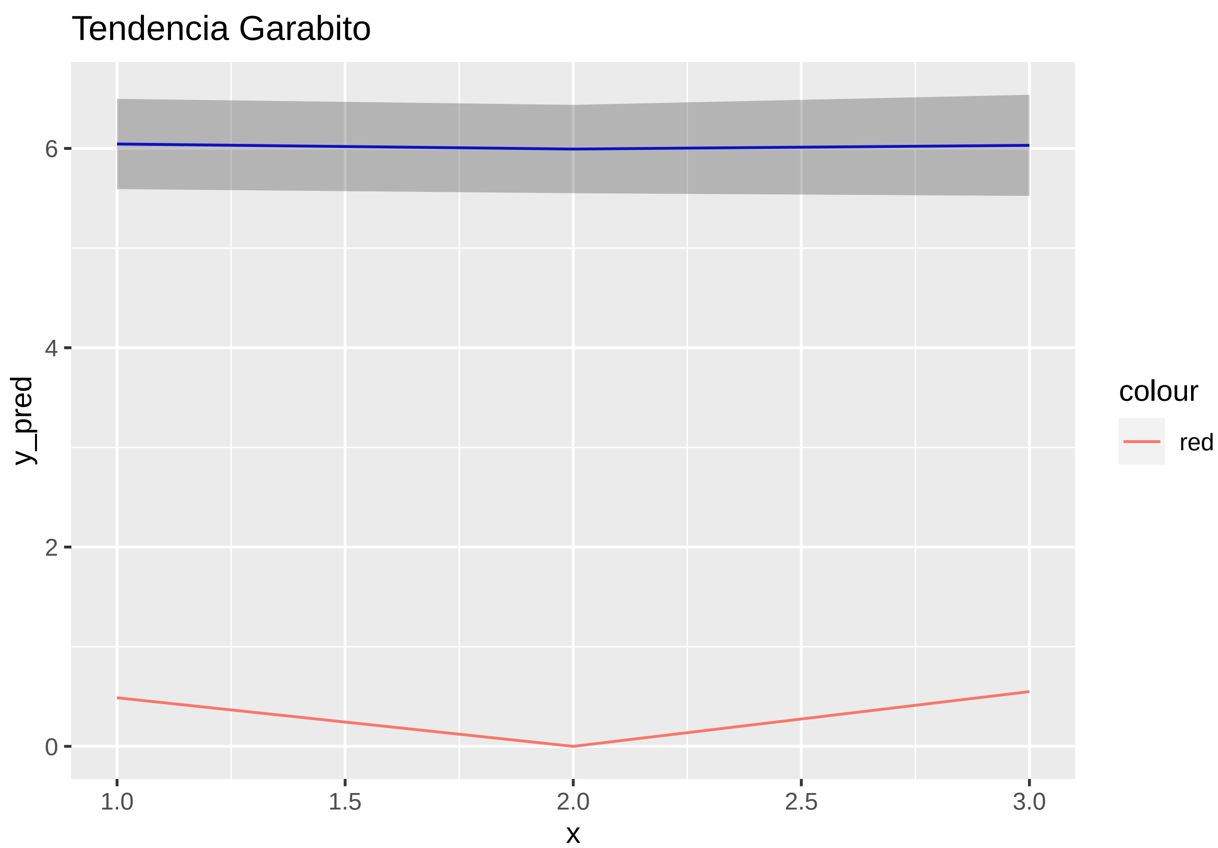

```
##
## [[7]]
```


Tendencia Desamparados
0.20
0.15
0.10
y_pred
1.0
1.5
2.0
2.5
3.0
x
colour
red

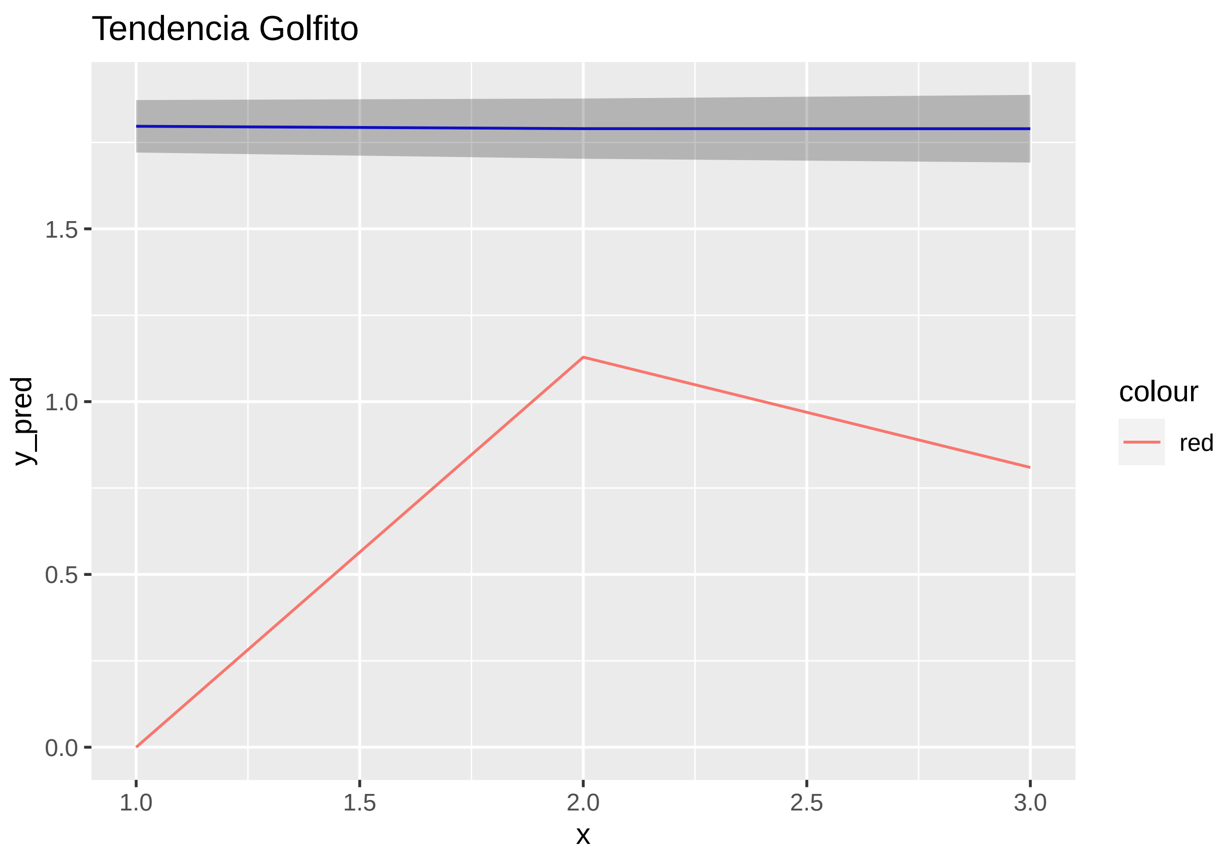
```
##
## [[8]]
```



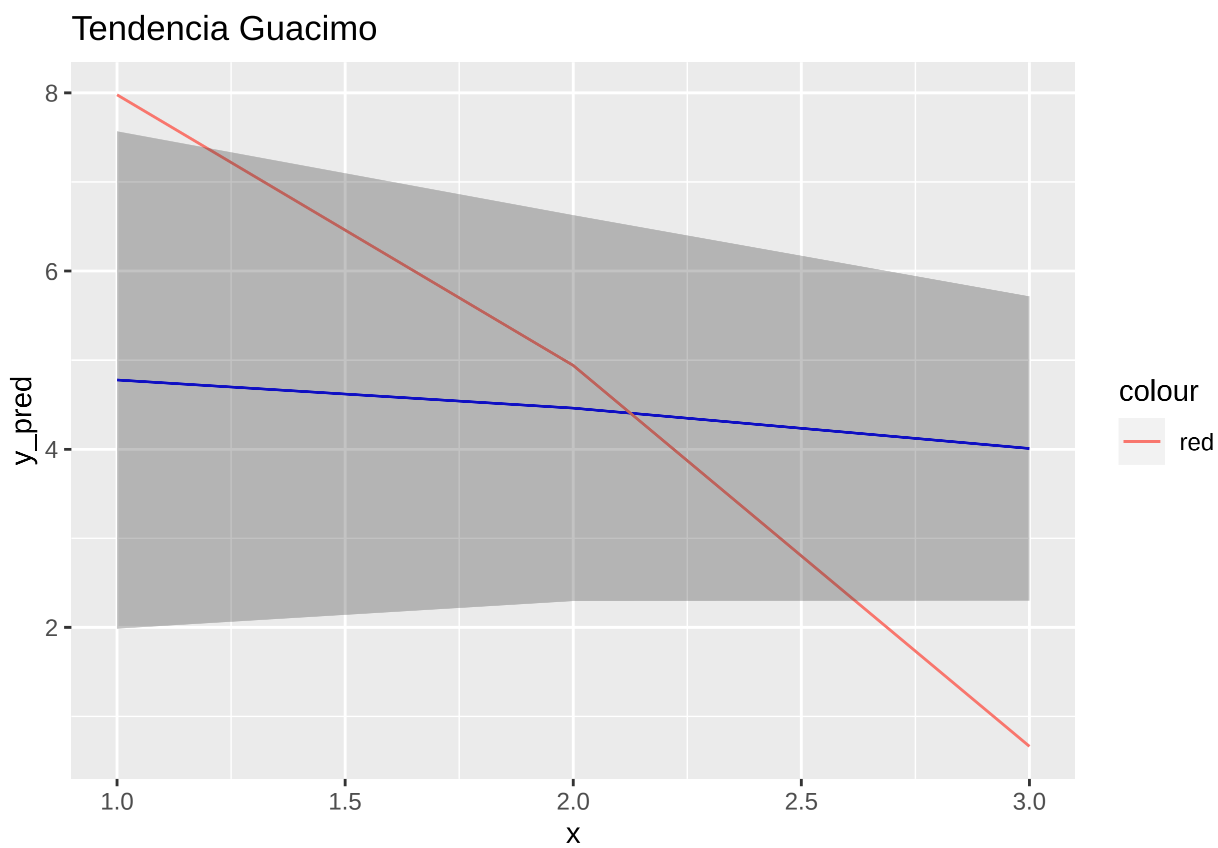

```
##
## [[9]]
```



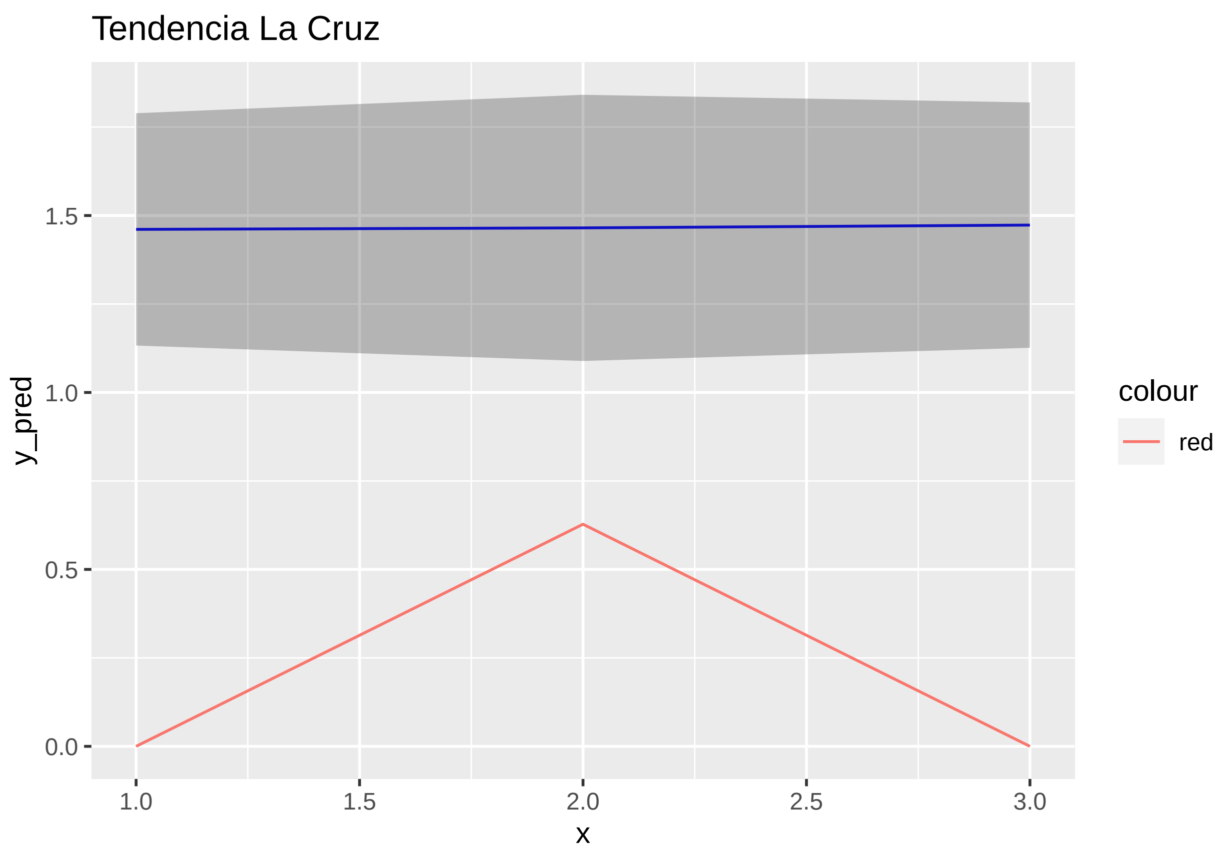

```
##
## [[10]]
```



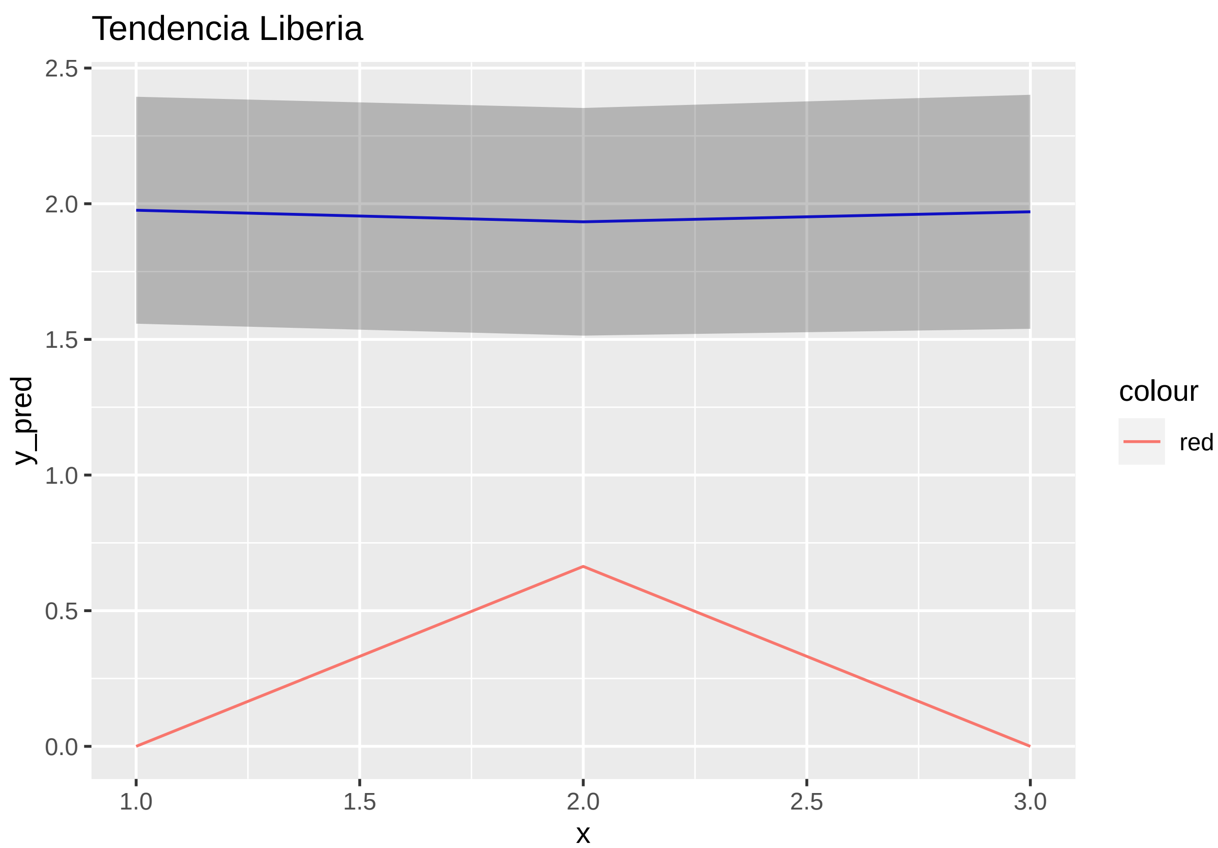

```
##
## [[11]]
```



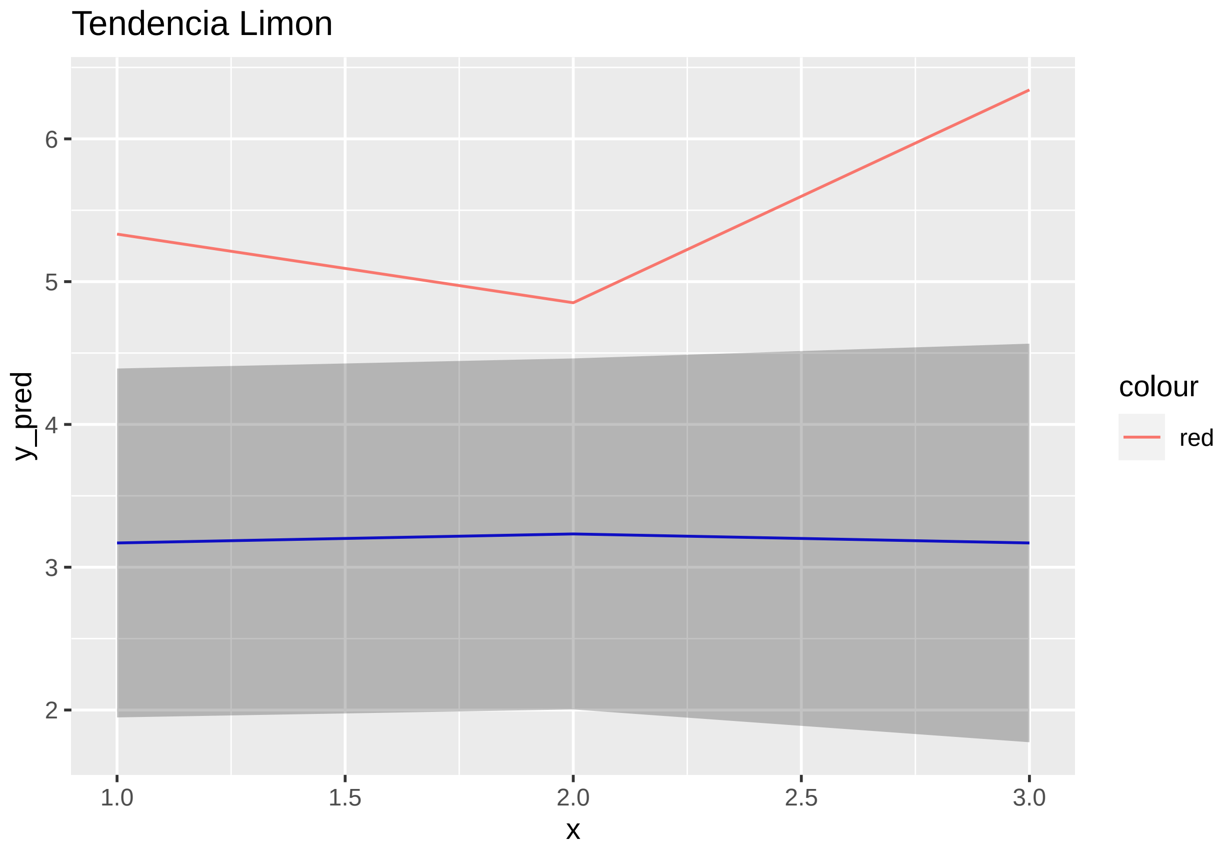

```
##
## [[12]]
```



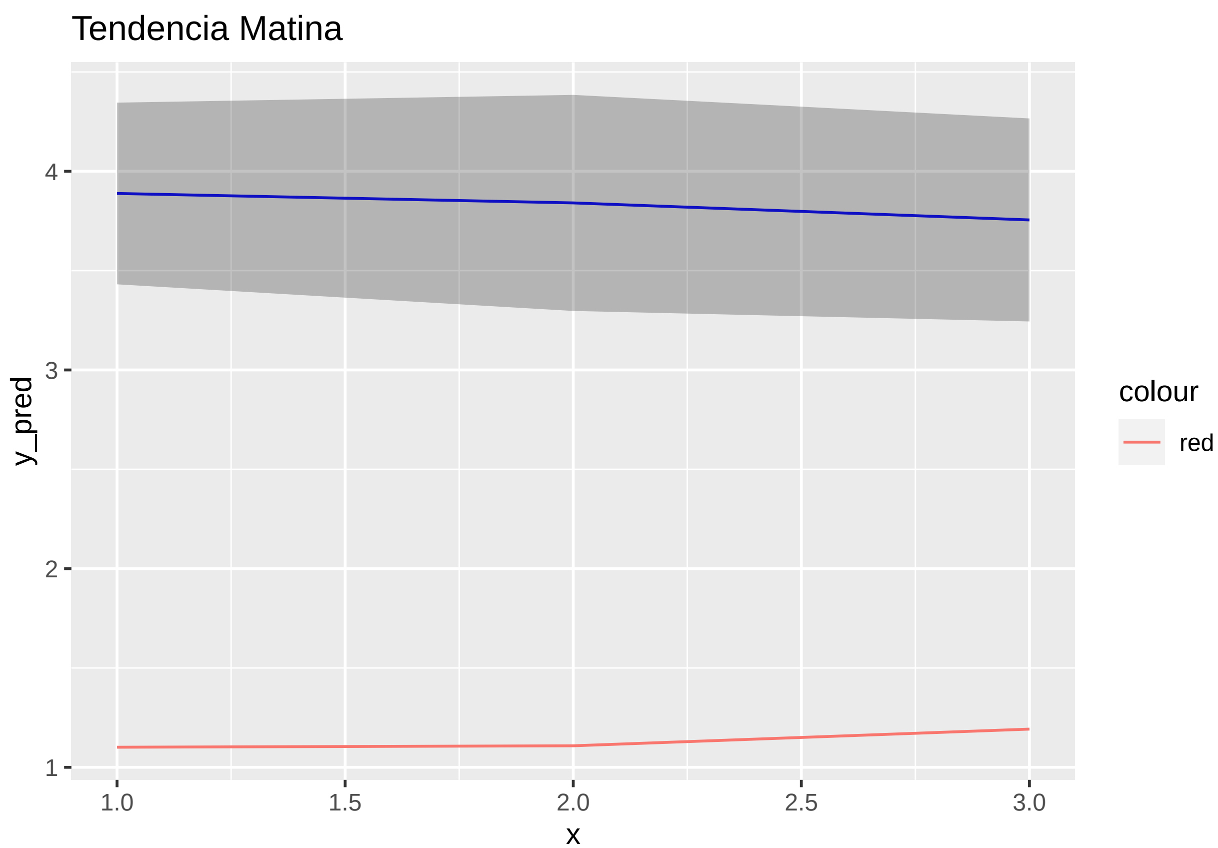

```
##
## [[13]]
```


Tendencia Liberia
y_pred
x
colour
red
0.0
0.5
1.0
1.5
2.0
2.5
1.0
1.5
2.0
2.5
3.0

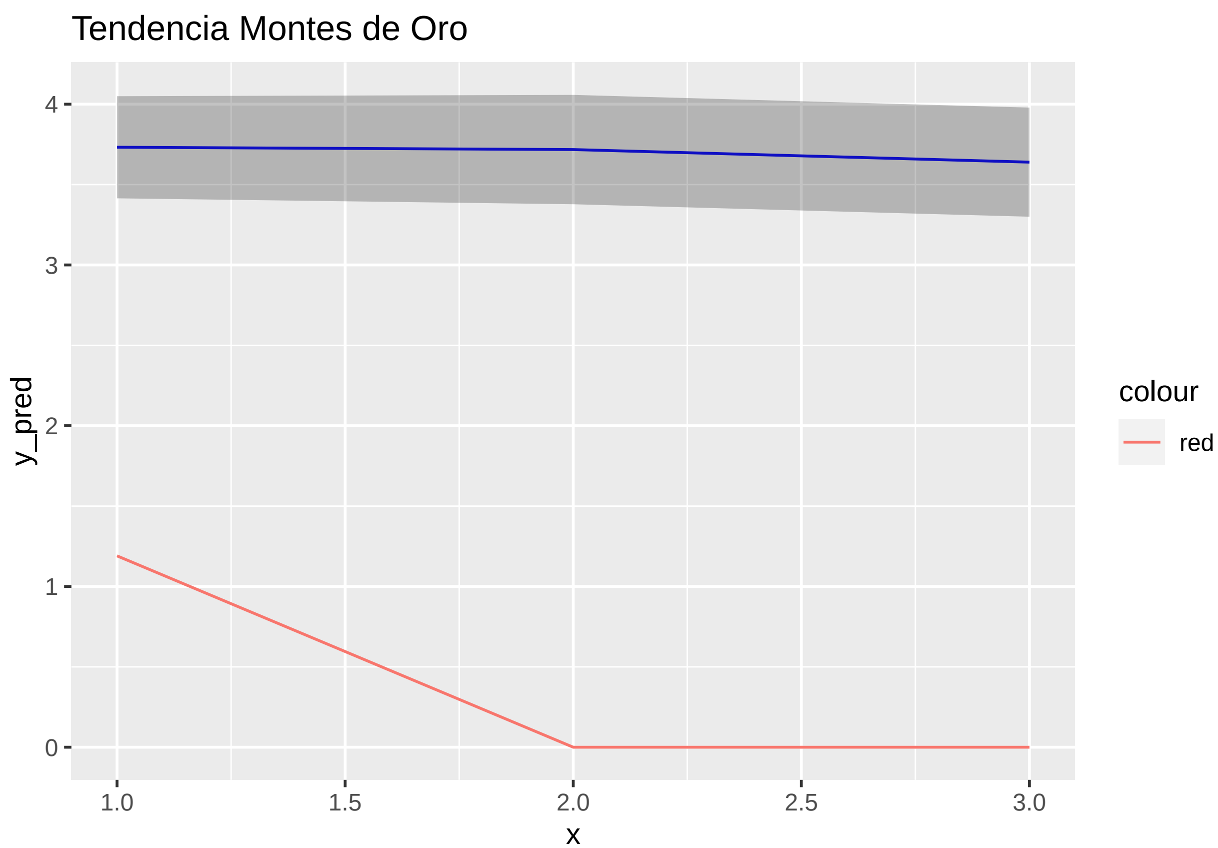
```
##
## [[14]]
```



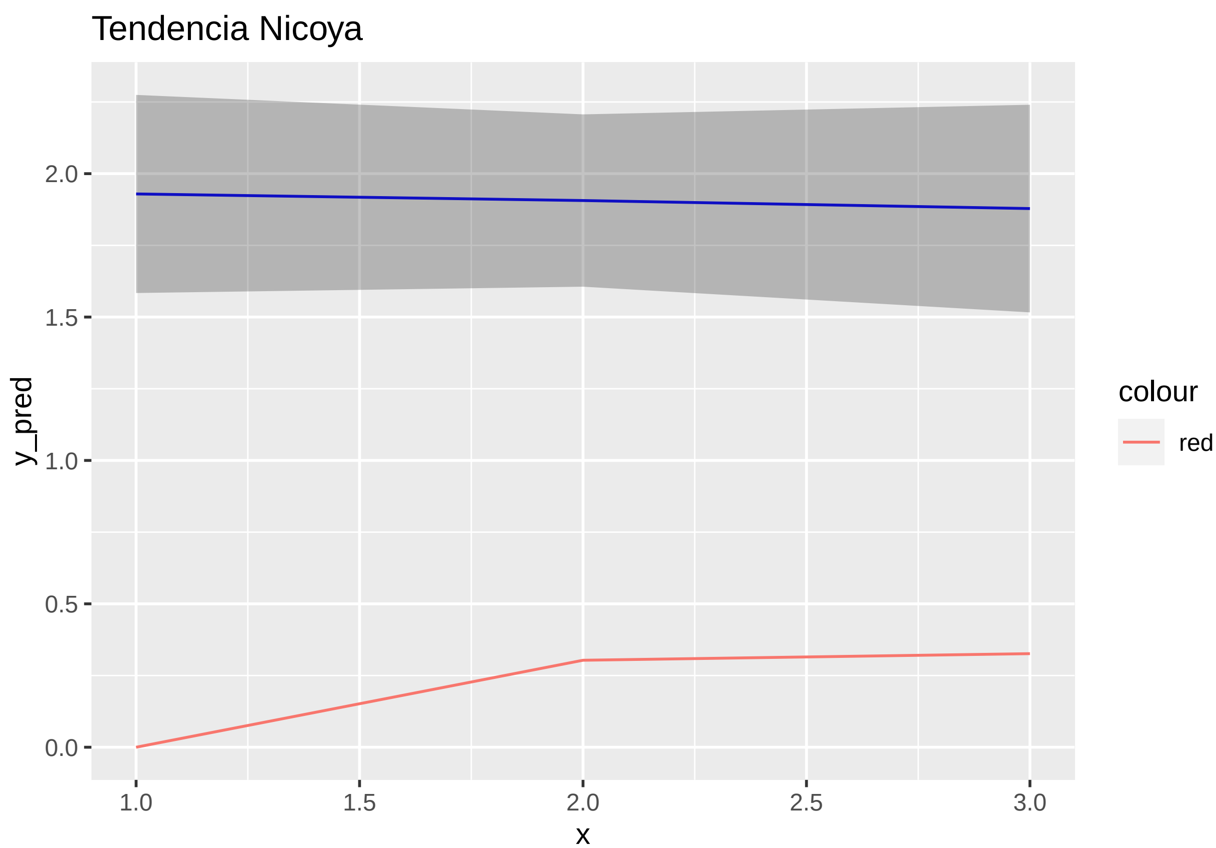

```
##
## [[15]]
```



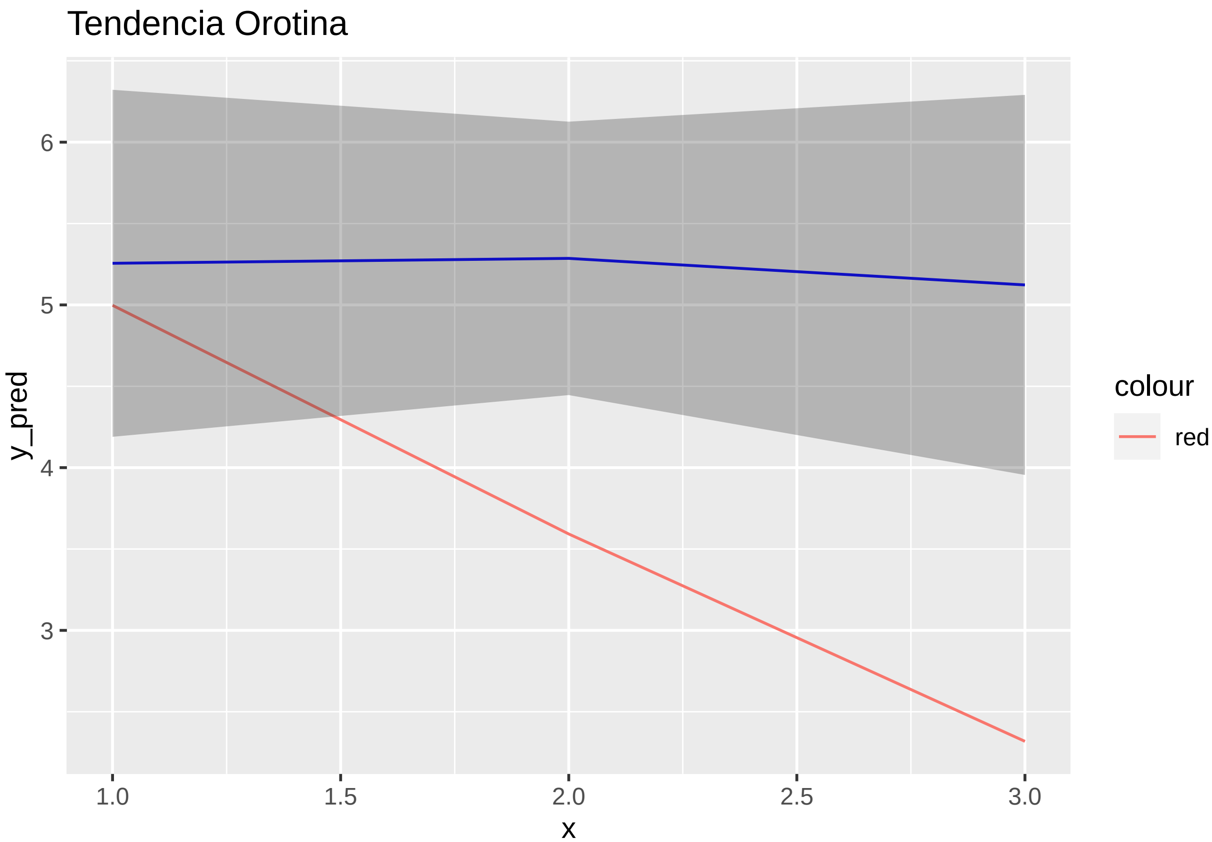

```
##
## [[16]]
```


Tendencia Montes de Oro
y_pred
x
colour
red
0
1
2
3
4
1.0
1.5
2.0
2.5
3.0

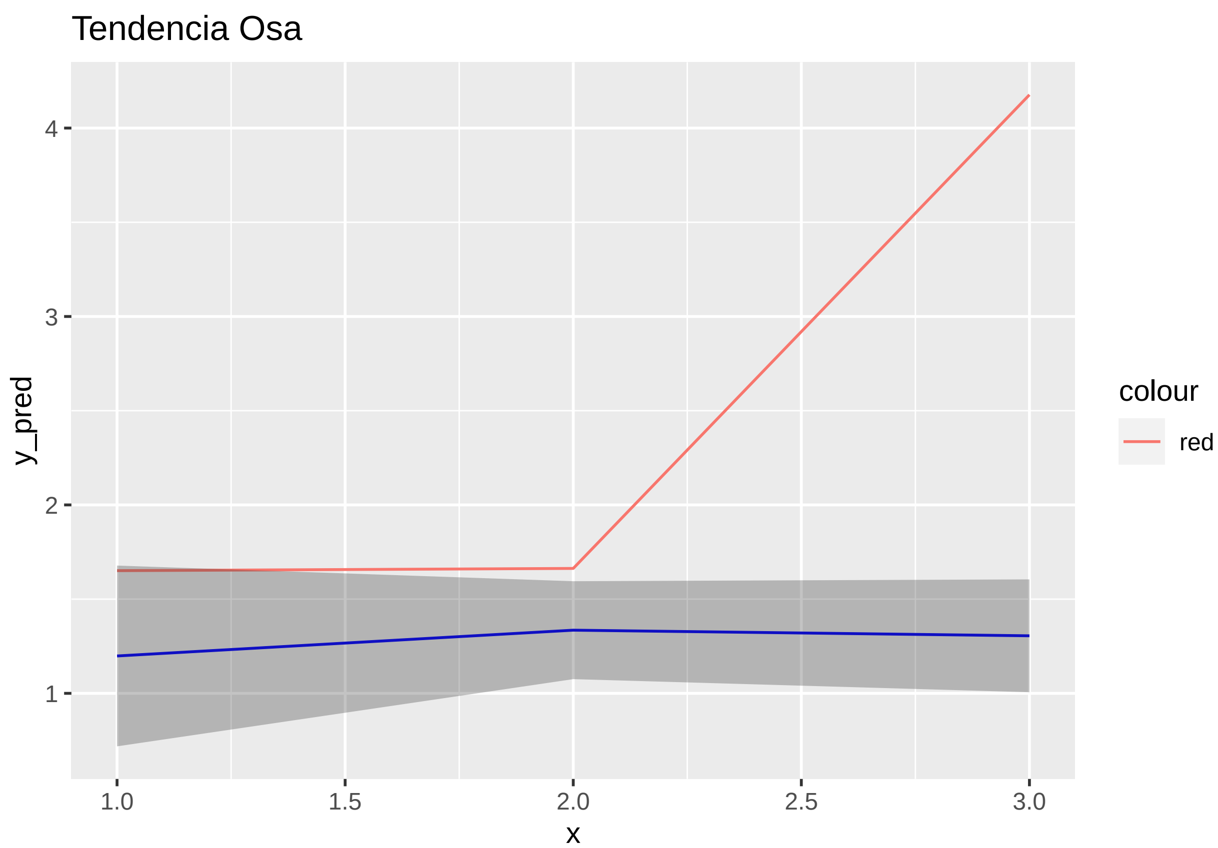
```
##
## [[17]]
```


Tendencia Nicoya
y_pred
2.0
1.5
1.0
0.5
0.0
1.0
1.5
2.0
2.5
3.0
x
colour
red

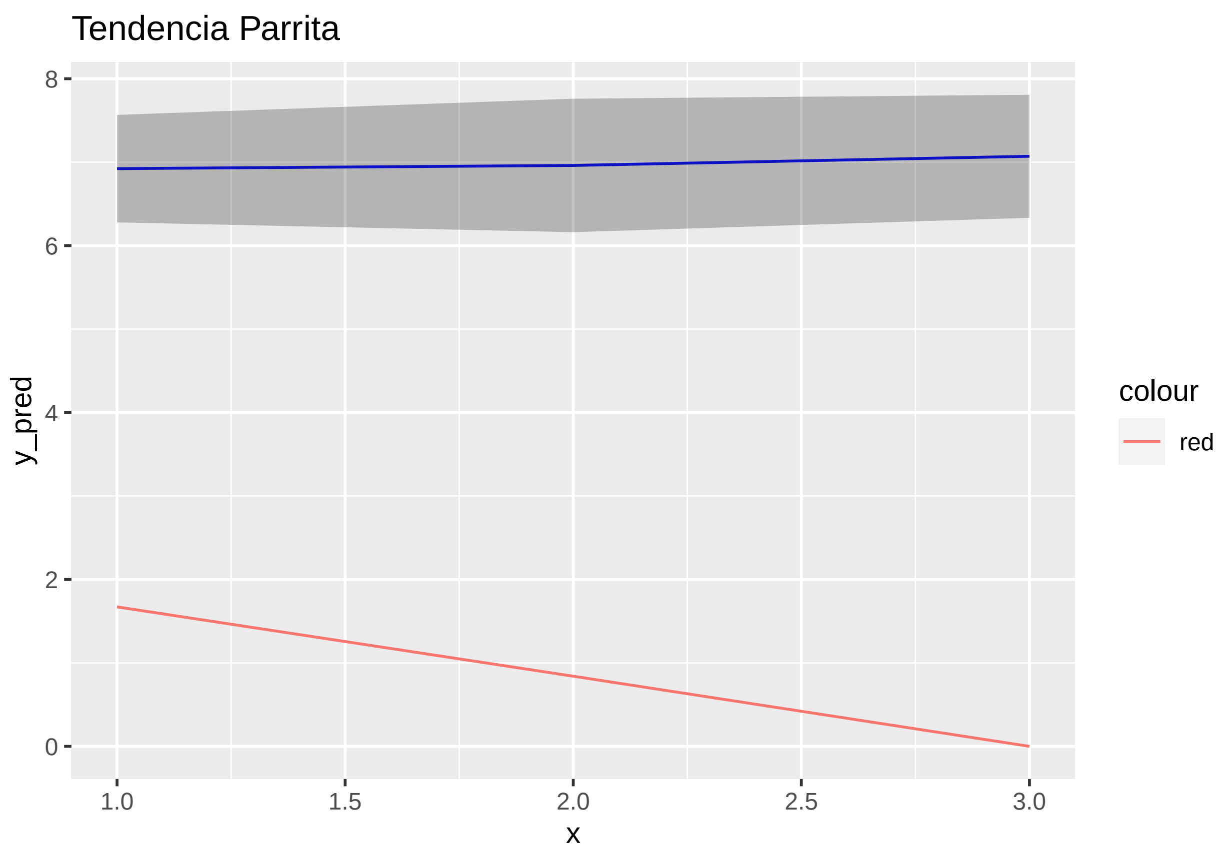
```
##
## [[18]]
```



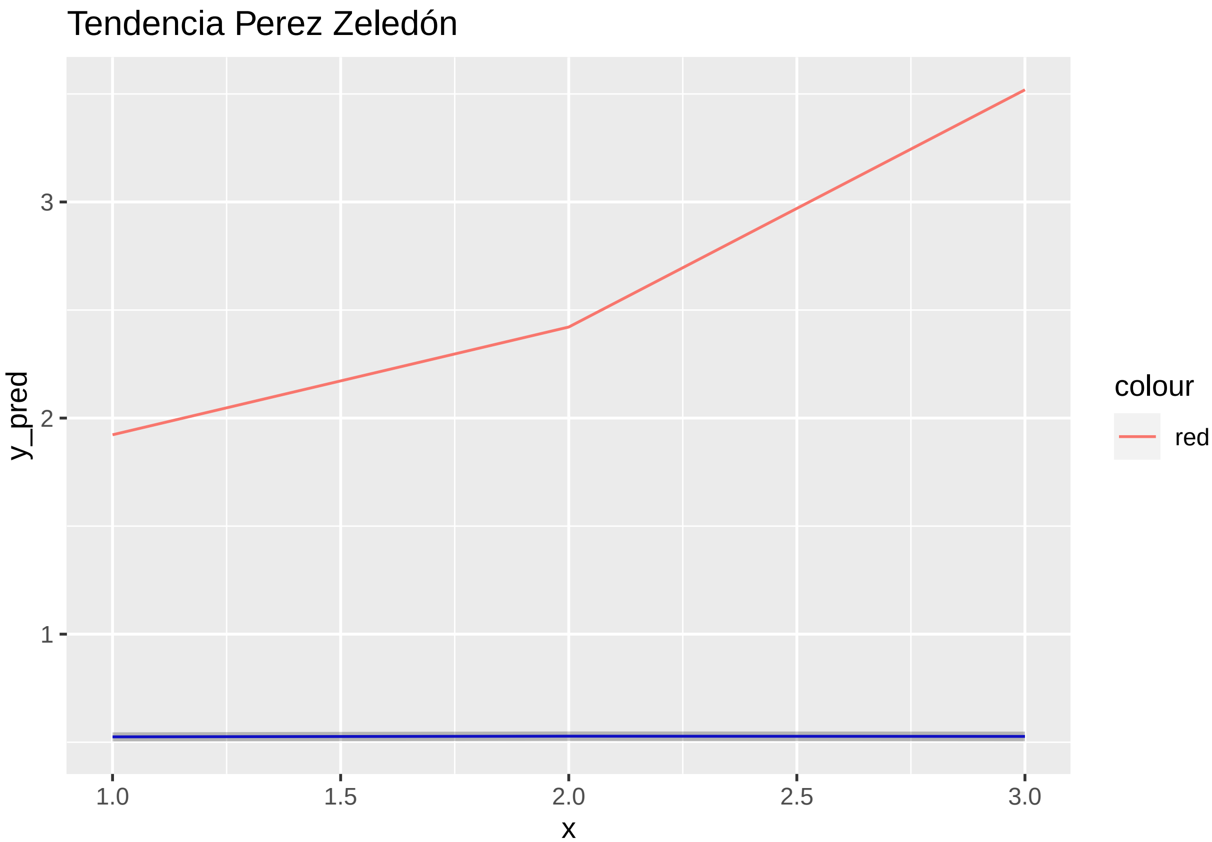

```
##
## [[19]]
```



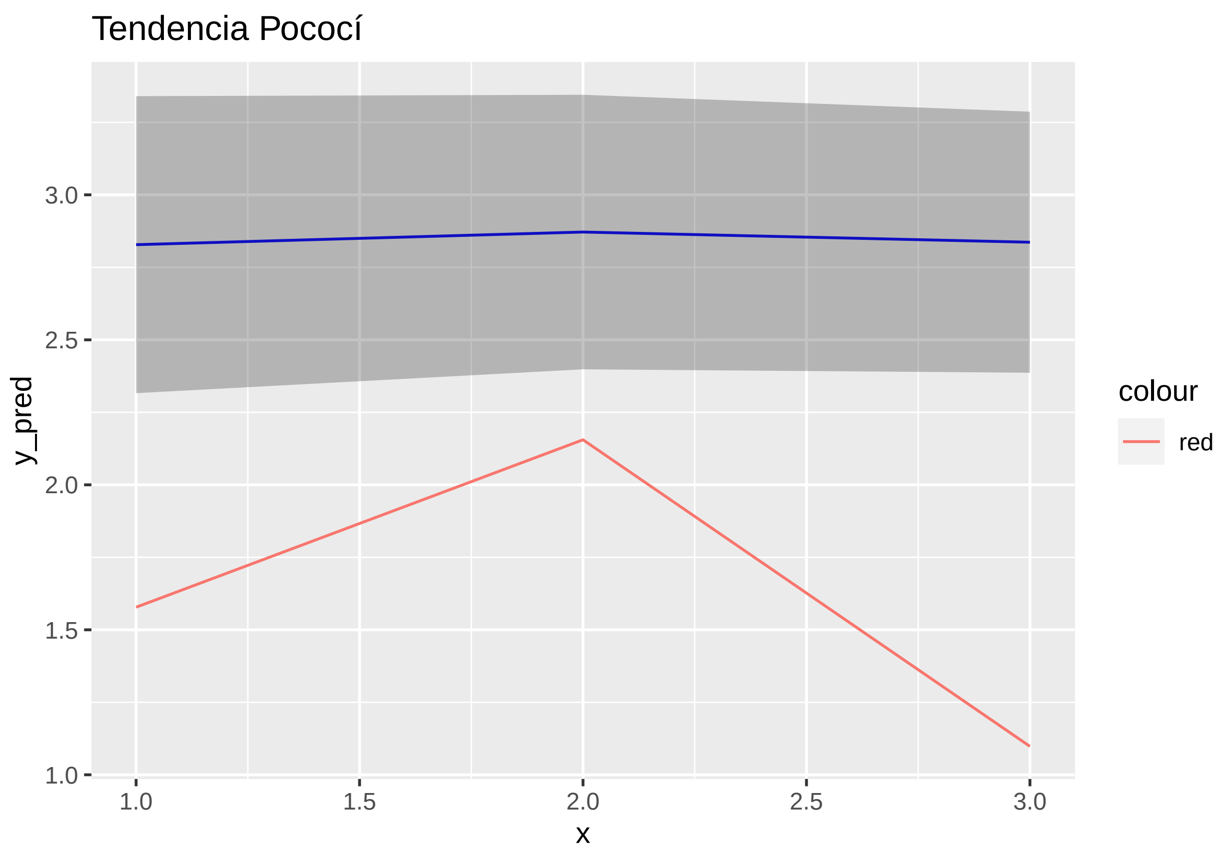

```
##
## [[20]]
```




```
##
## [[21]]
```

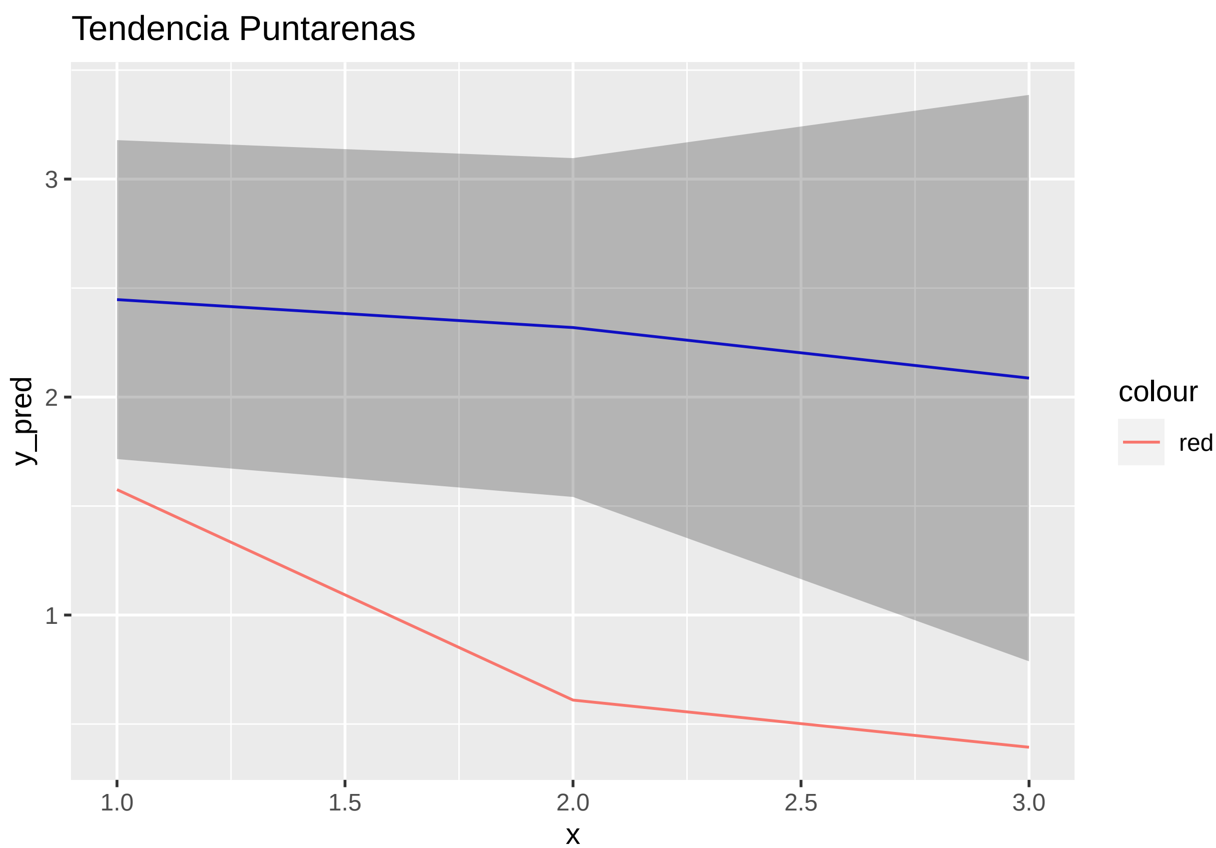



```
##
## [[22]]
```

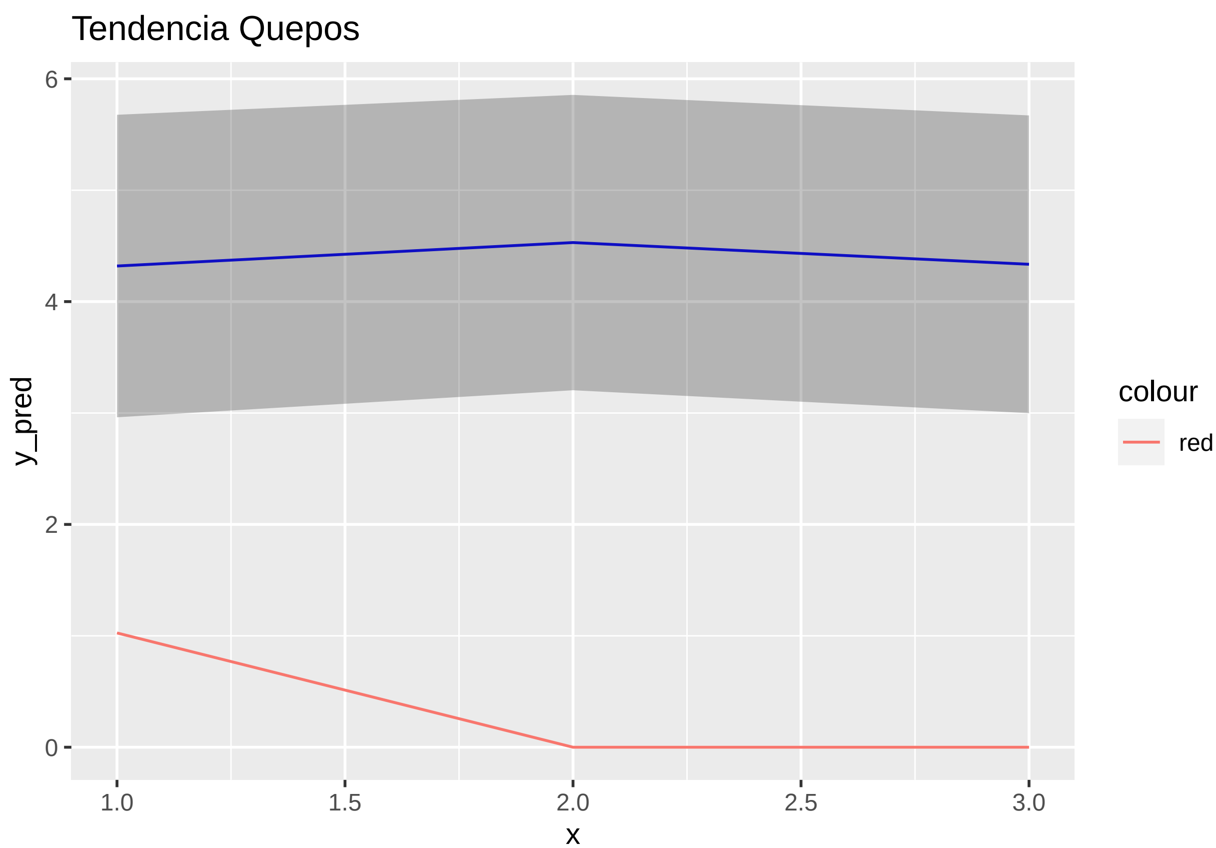


Tendencia Pococí
y_pred
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
x
colour
red

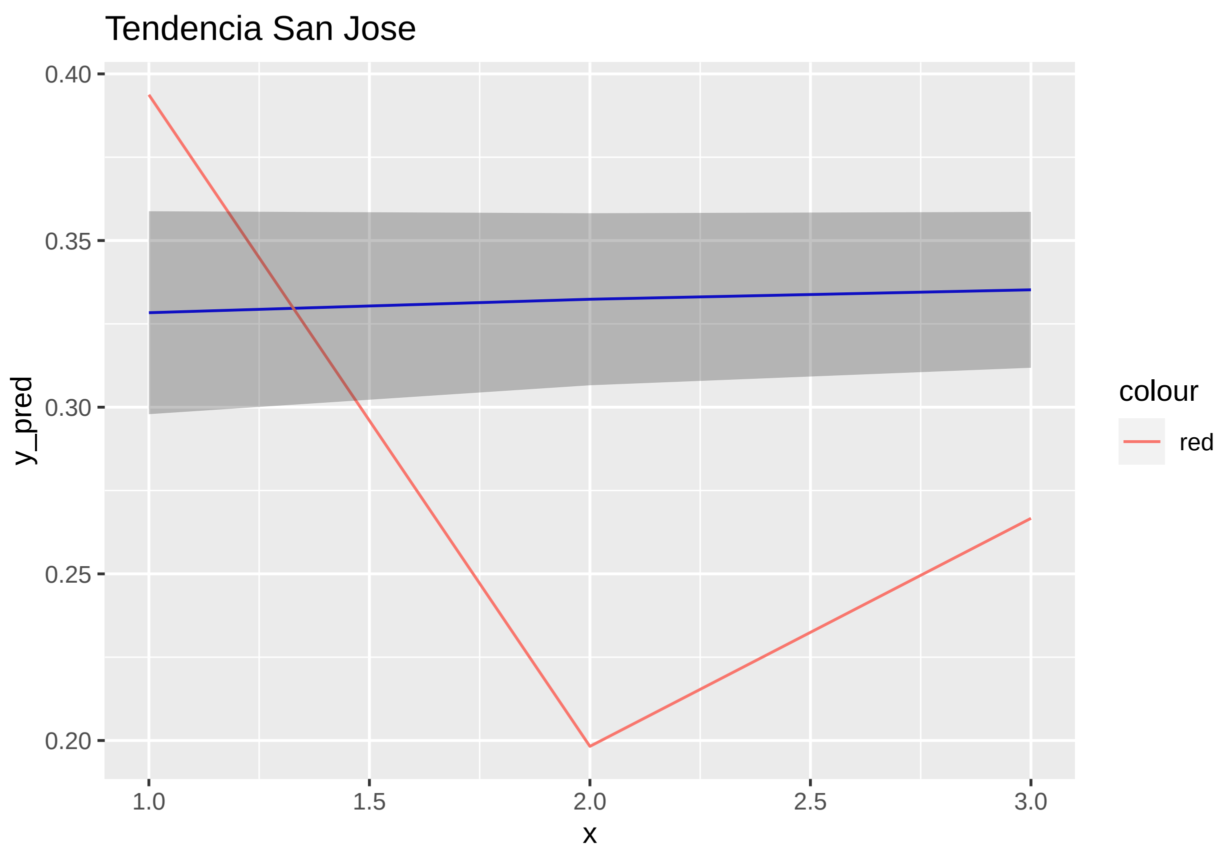

```
##
## [[23]]
```




```
##
## [[24]]
```

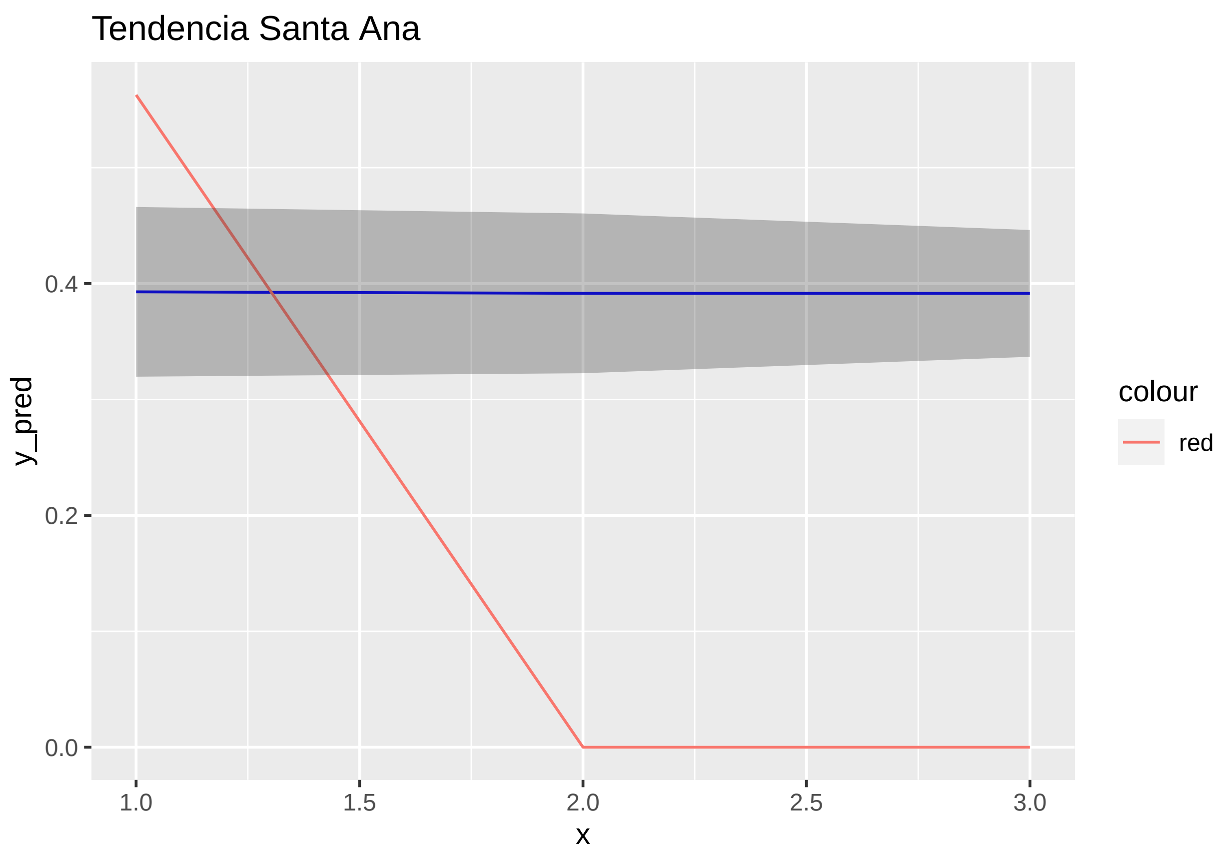



```
##
## [[25]]
```

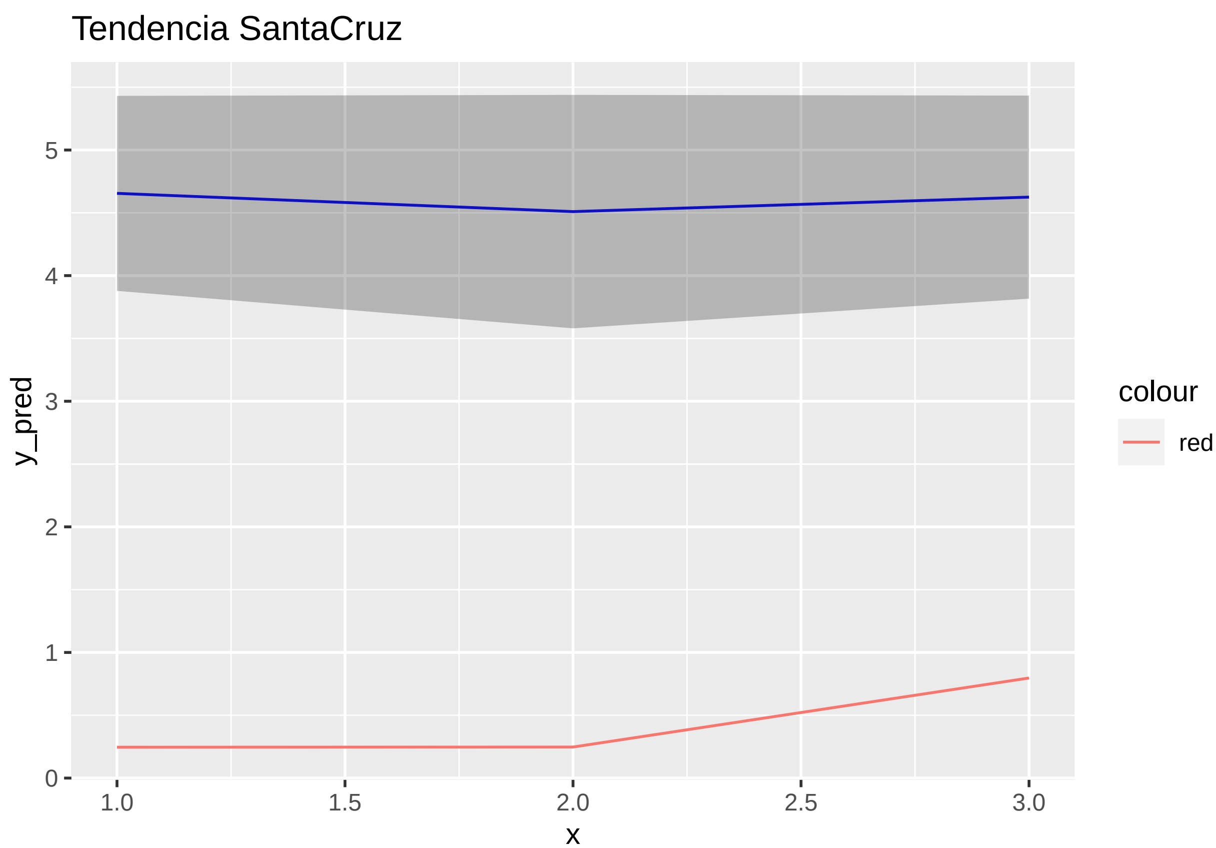


Tendencia San Jose
y_pred
0.40
0.35
0.30
0.25
0.20
1.0
1.5
2.0
2.5
3.0
x
colour
red

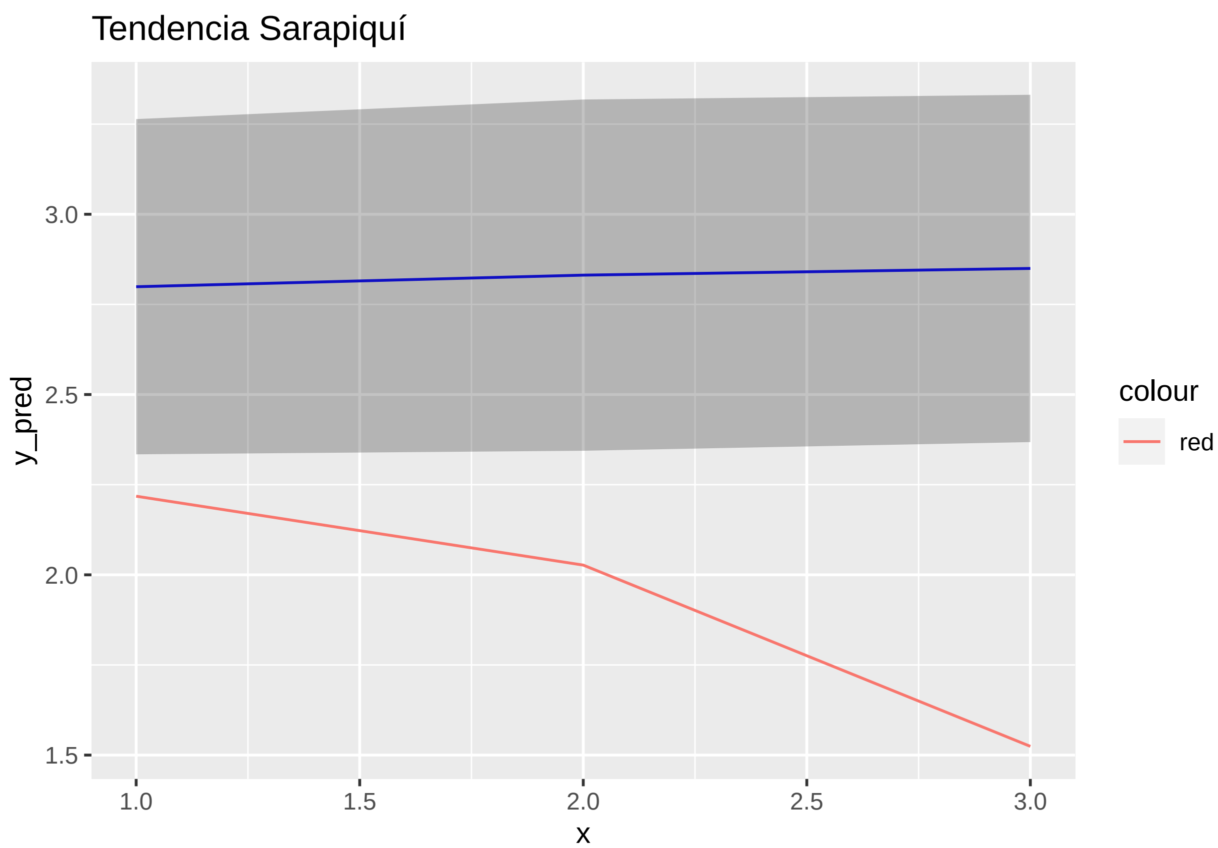

```
##
## [[26]]
```



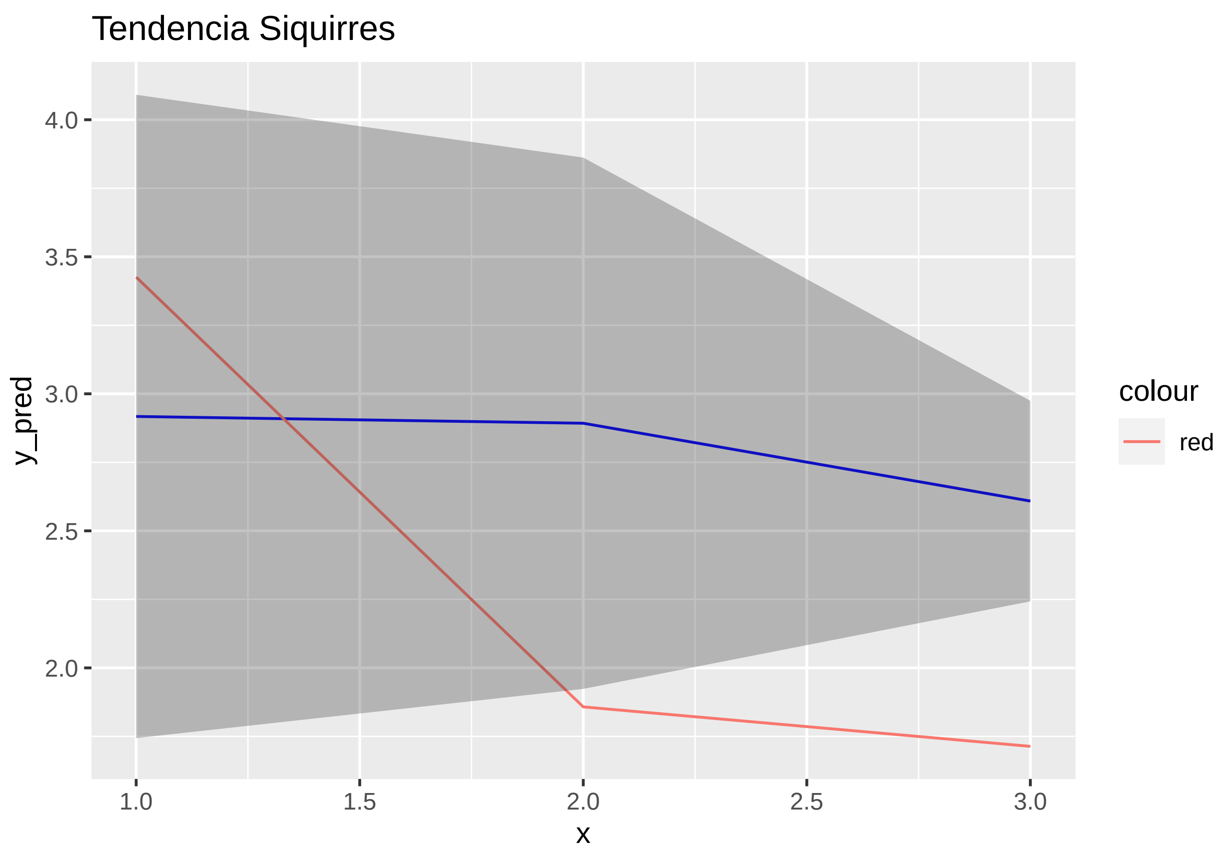

```
##
## [[27]]
```



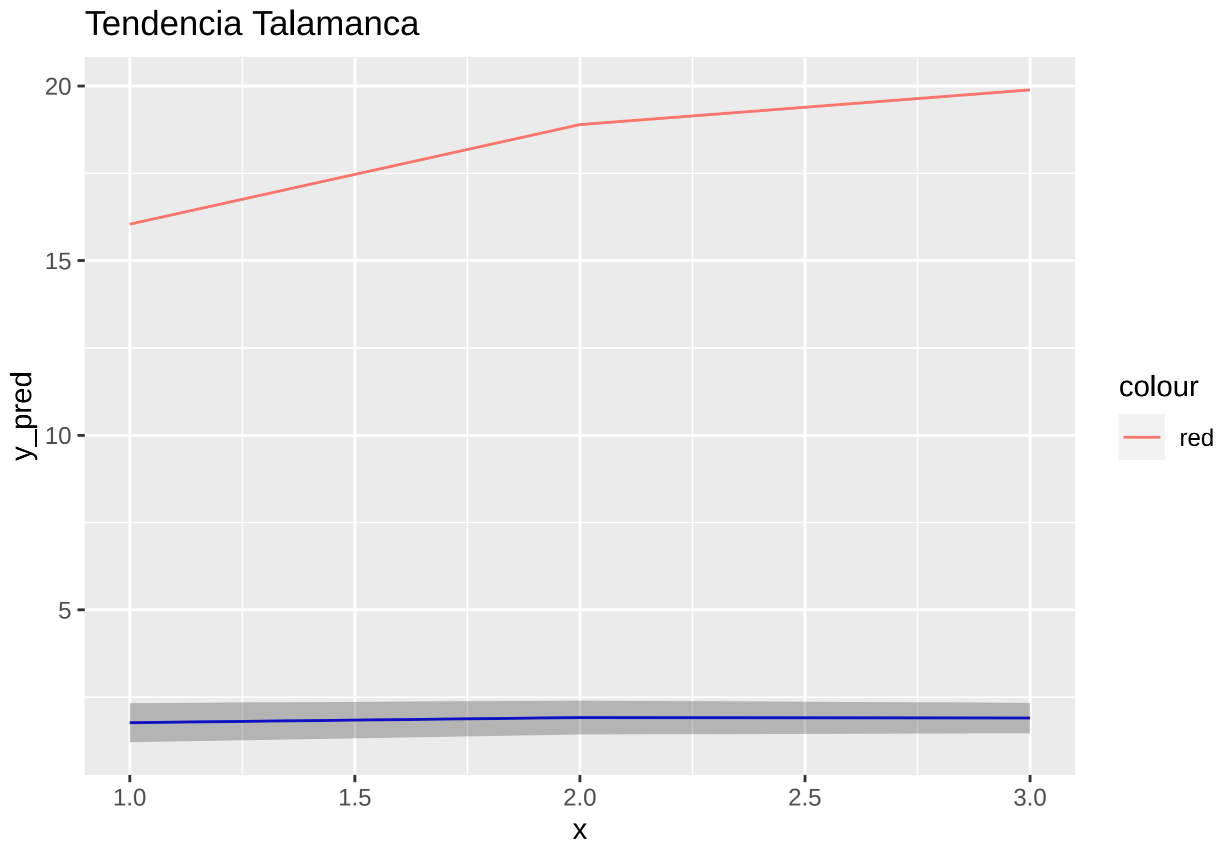

```
##
## [[28]]
```


Tendencia Sarapiquí
3.0
2.5
2.0
1.5
y_pred
1.0
1.5
2.0
2.5
3.0
x
colour
red

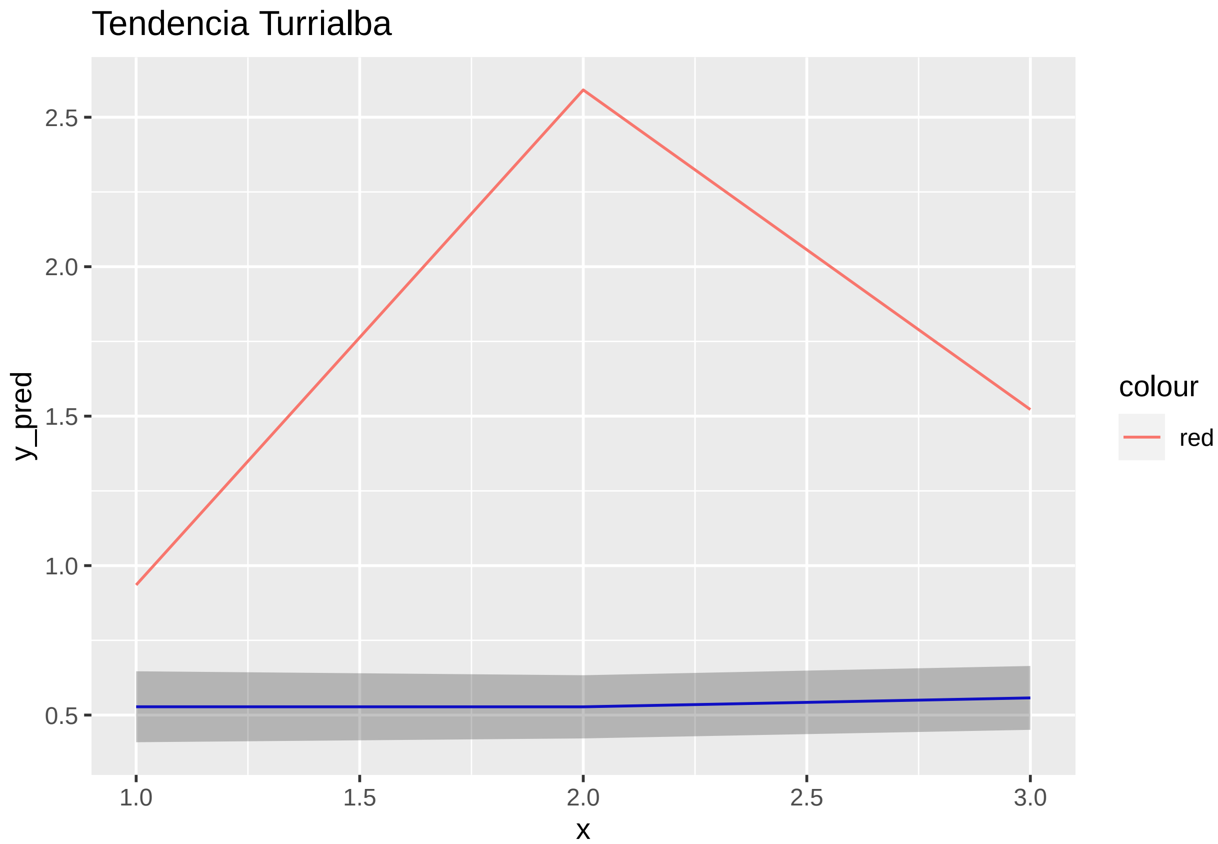
```
##
## [[29]]
```


Tendencia Siquirres
4.0
3.5
3.0
2.5
2.0
y_pred
1.0
1.5
2.0
2.5
3.0
x
colour
red

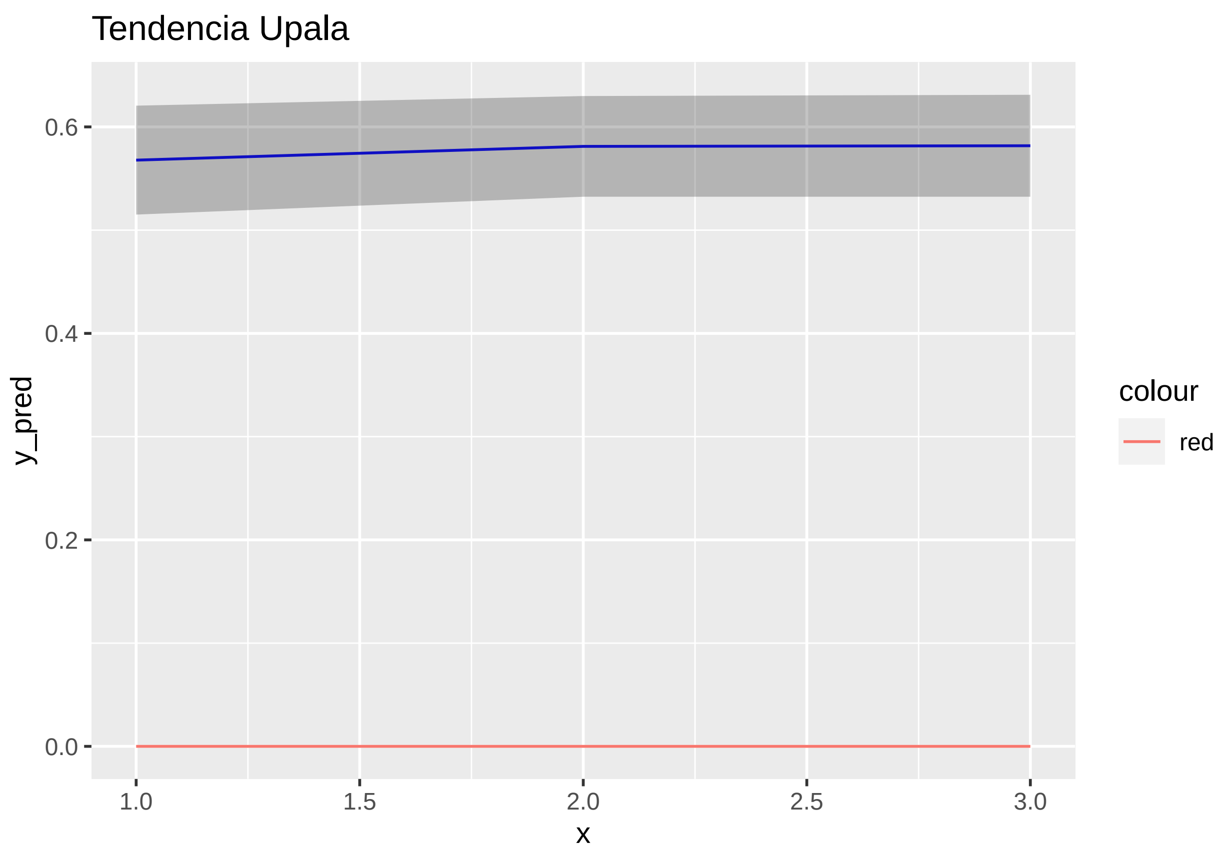
```
##
## [[30]]
```


Tendencia Talamanca
20
15
10
5
y_pred
1.0
1.5
2.0
2.5
3.0
x
colour
red

```
##
## [[31]]
```


Tendencia Turrialba
y_pred
2.5
2.0
1.5
1.0
0.5
1.0
1.5
2.0
2.5
3.0
x
colour
red

```
##
## [[32]]
```



Valores aproximados dentro del modelo

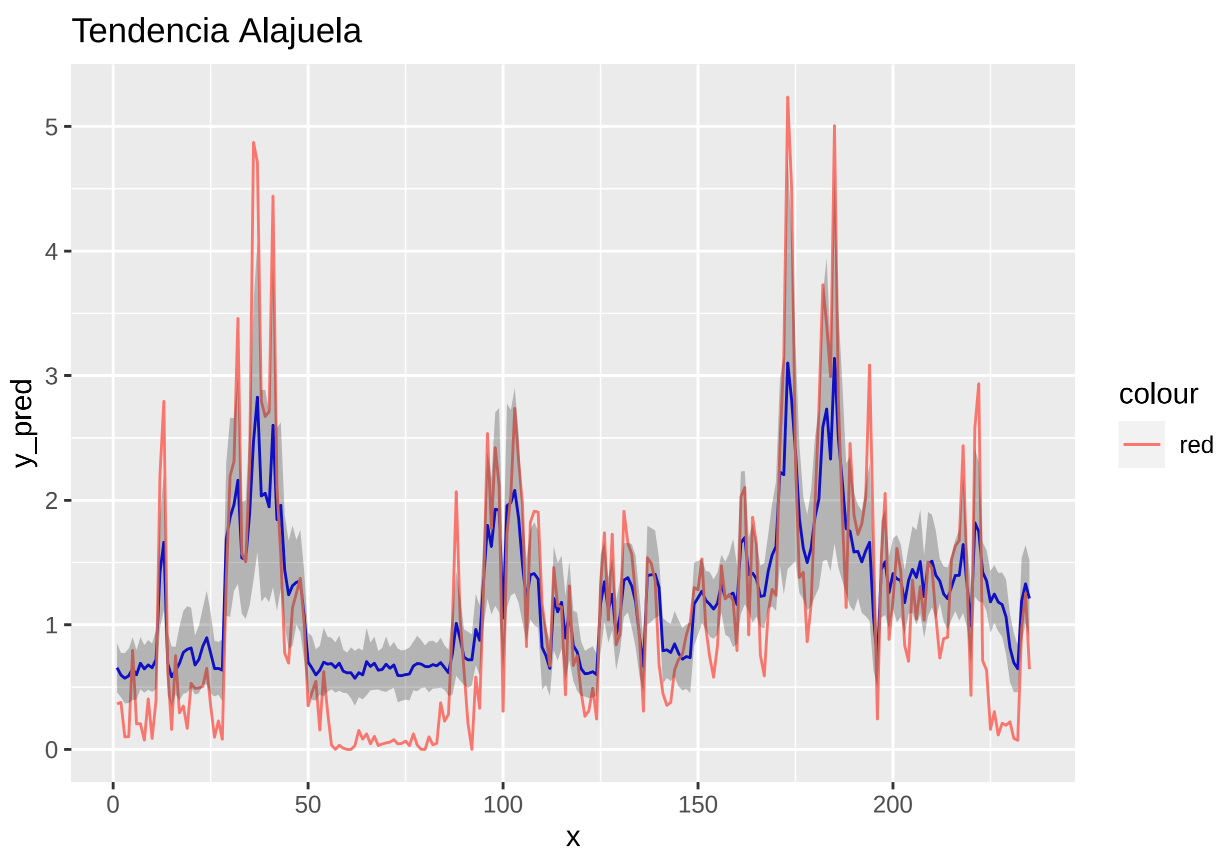
```
Evaluacion_c
```

```
##             [,1]      [,2]
##  [1,]  0.2899243  6.336410
##  [2,]  0.1813619 19.050655
##  [3,]  1.7255596  6.675219
##  [4,]  1.8958163  7.522802
##  [5,]  2.6655264 11.338539
##  [6,]  2.0497242  9.302687
##  [7,]  0.1076688  7.695768
##  [8,]  2.6457528 13.860206
##  [9,] 10.8421340 34.949097
## [10,]  4.3718985 38.283779
## [11,]  1.1330518  5.475617
## [12,]  6.5355781 31.057102
## [13,]  3.9604497 32.579141
## [14,]  2.8038554 19.099305
## [15,]  4.0965862 23.858428
## [16,]  5.6003805 31.238513
## [17,]  2.7808634 25.681097
## [18,]  7.1941154 17.314984
## [19,]  4.1884385 25.271802
```

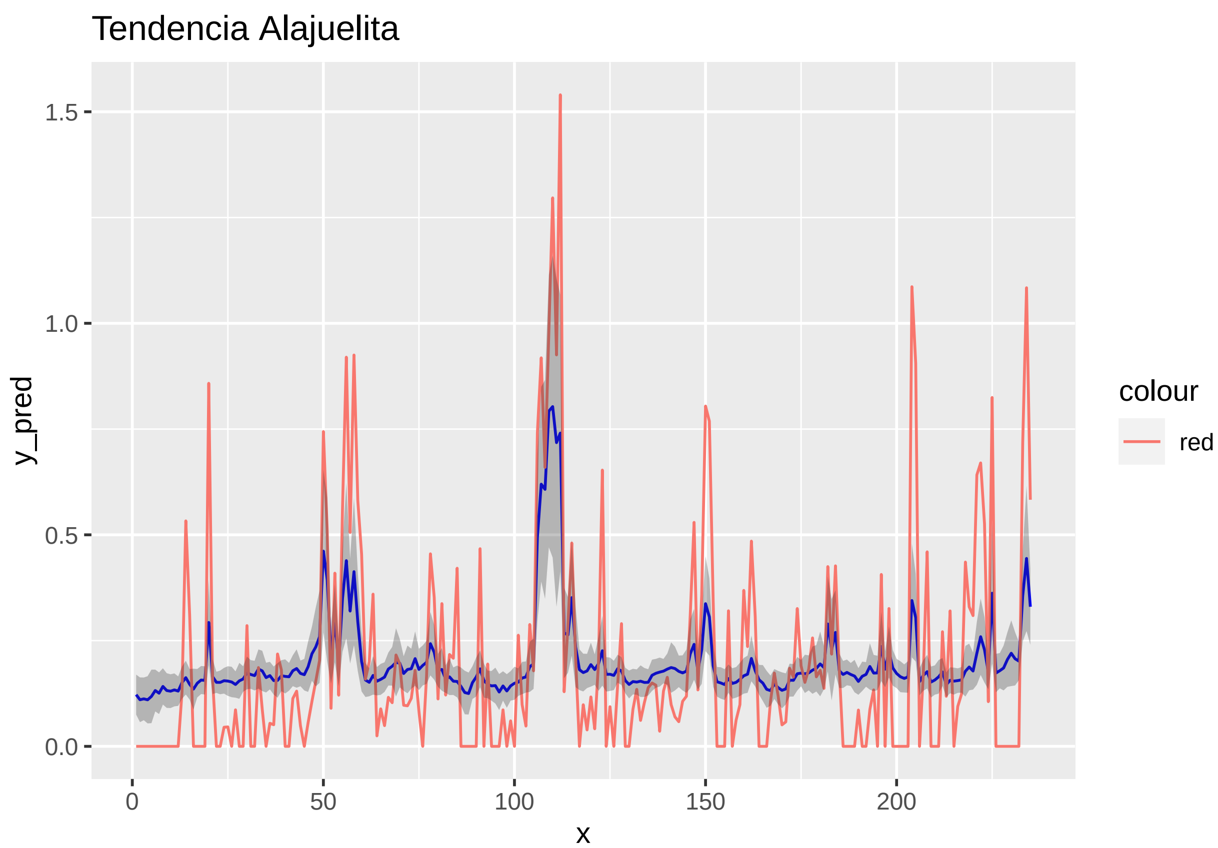
```
## [20,] 52.9163807 47.646329
## [21,]  2.2312385 38.103876
## [22,]  1.7547487 11.751129
## [23,]  1.4211853 11.730373
## [24,] 12.7761314 21.242980
## [25,]  0.2431911 23.236959
## [26,]  0.8458298 28.842565
## [27,] 13.0292622 37.035652
## [28,]  9.9773632 34.243826
## [29,]  2.2199829 16.523835
## [30,]  7.1376112 33.175337
## [31,]  2.6884849 30.789798
## [32,]  1.4040543 33.155706
```

```
plot_list_c
```

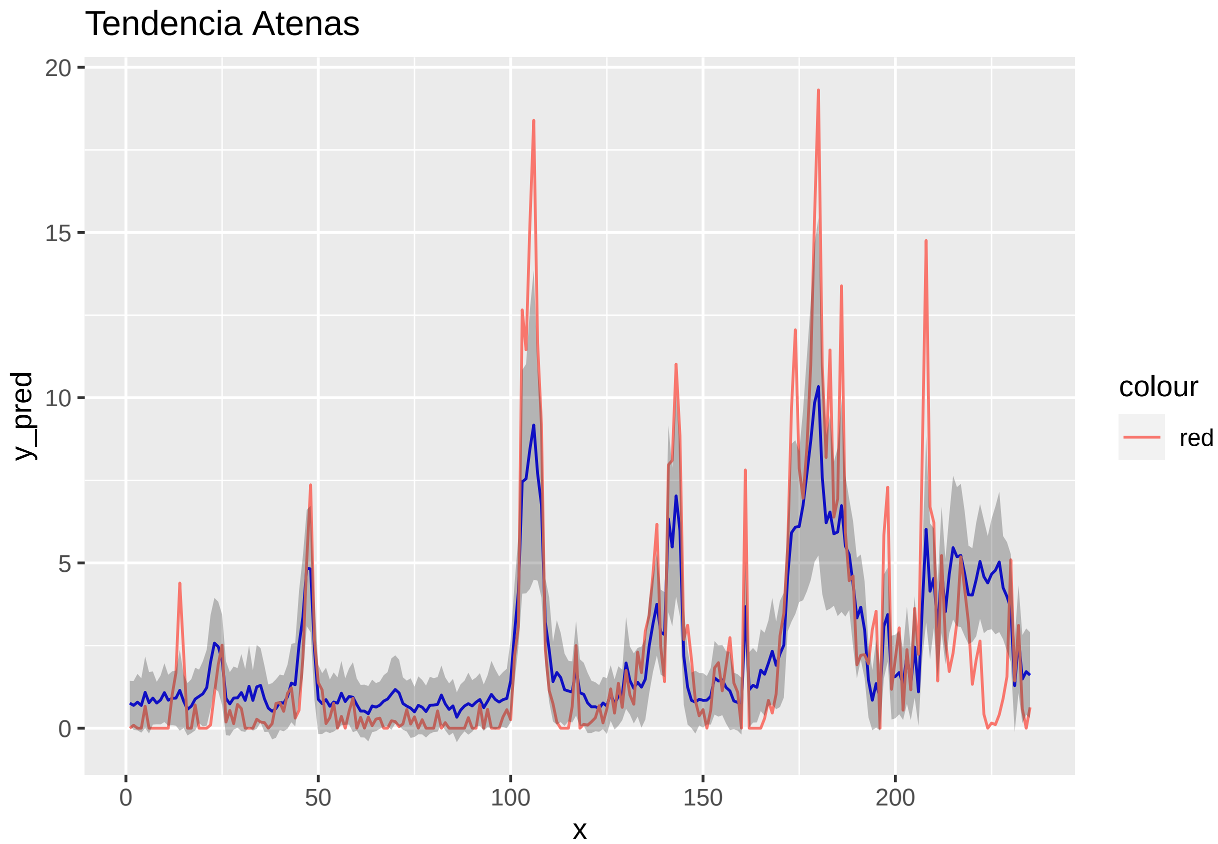
```
## [[1]]
```



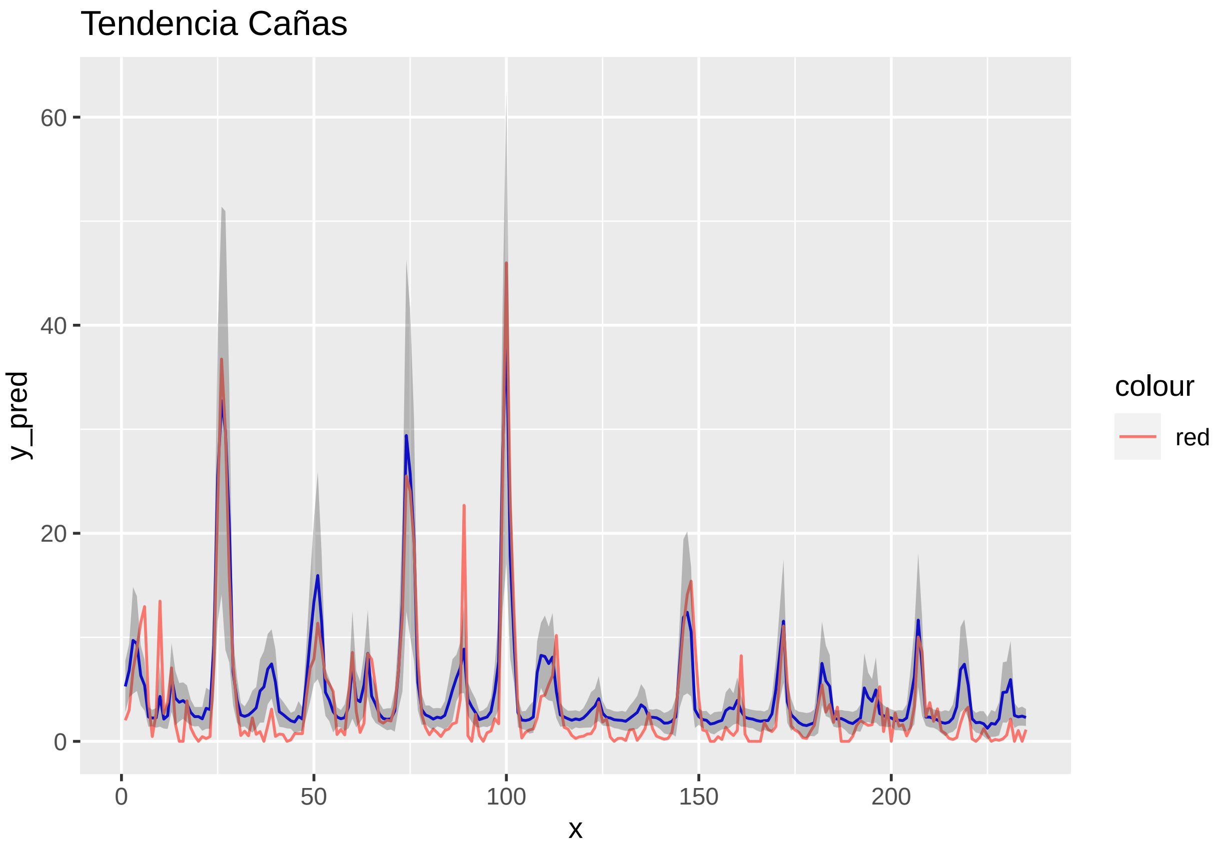

```
##
## [[2]]
```


Tendencia Alajuelita
y_pred
1.5
1.0
0.5
0.0
0
50
100
150
200
x
colour
red

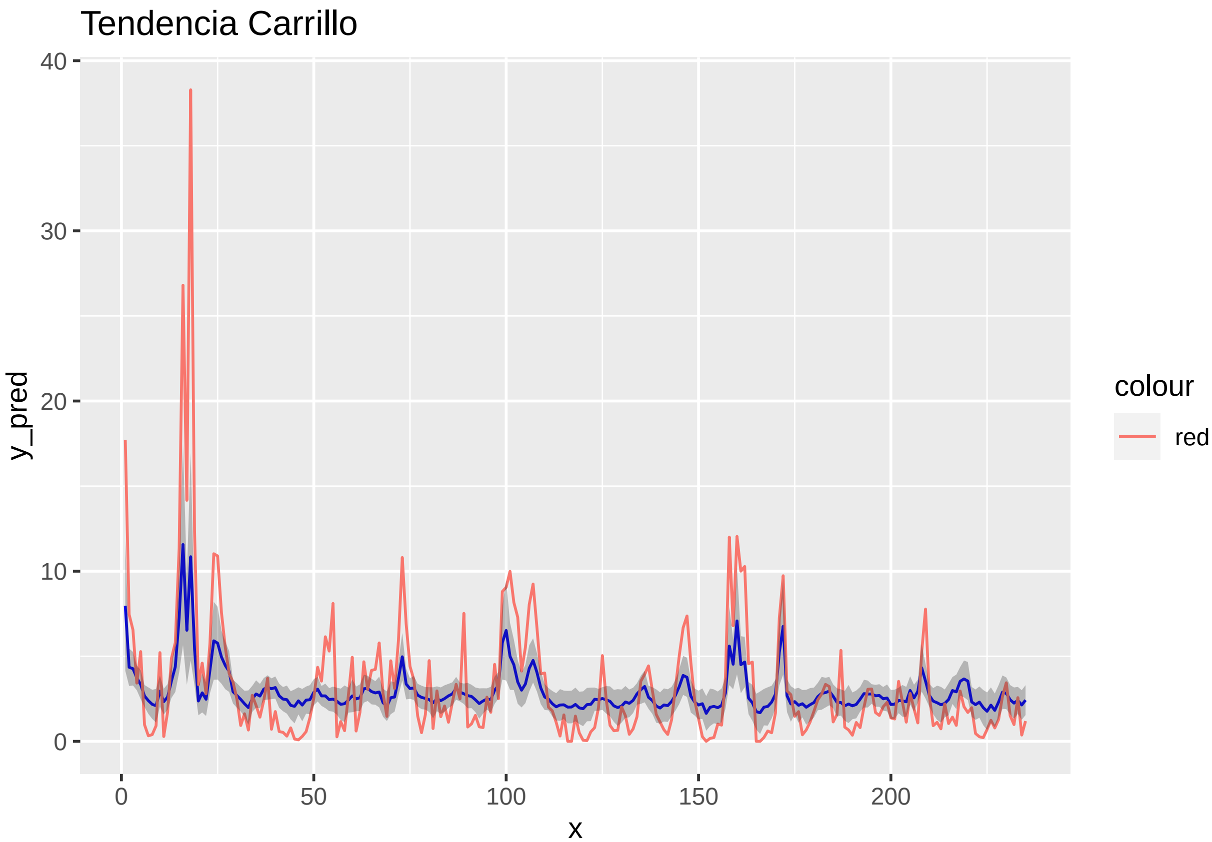
```
##
## [[3]]
```


Tendencia Atenas
20
15
10
5
0
y_pred
0
50
100
150
200
x
colour
red

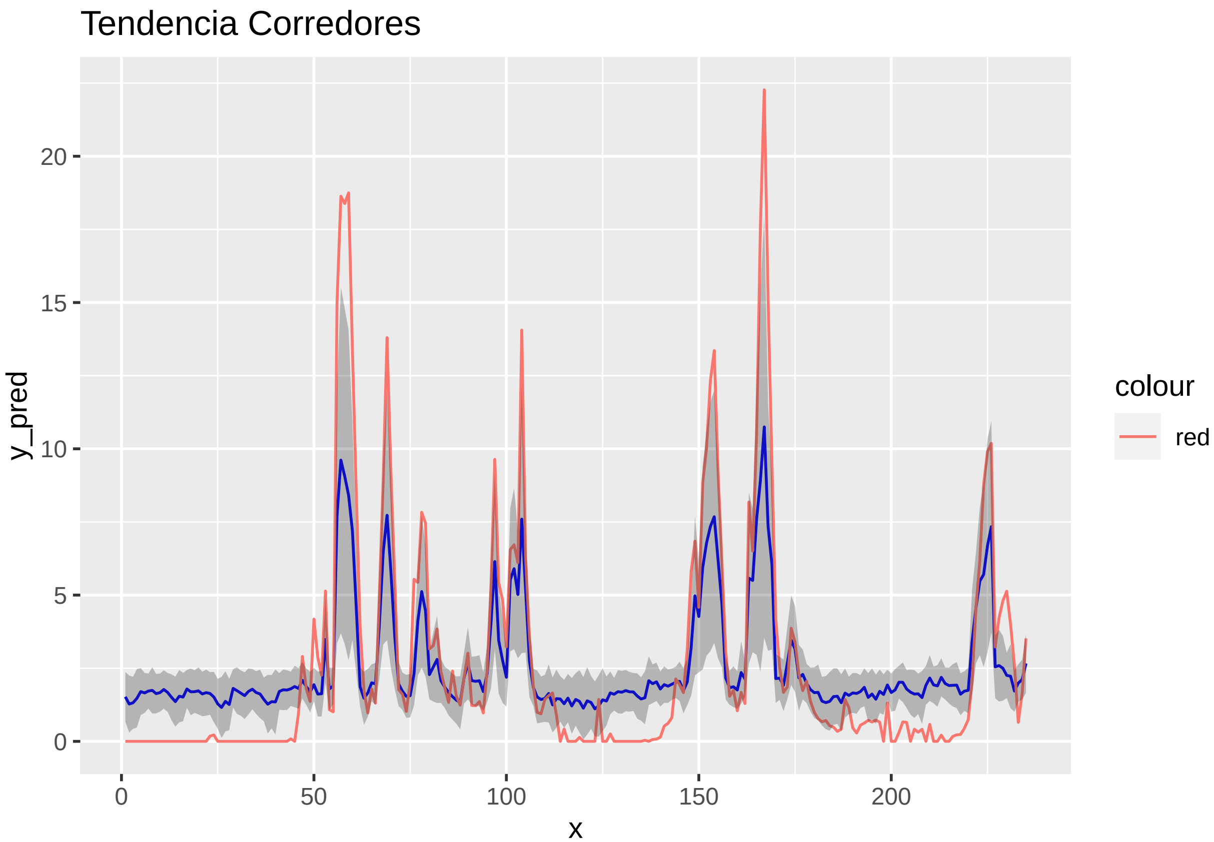
```
##
## [[4]]
```



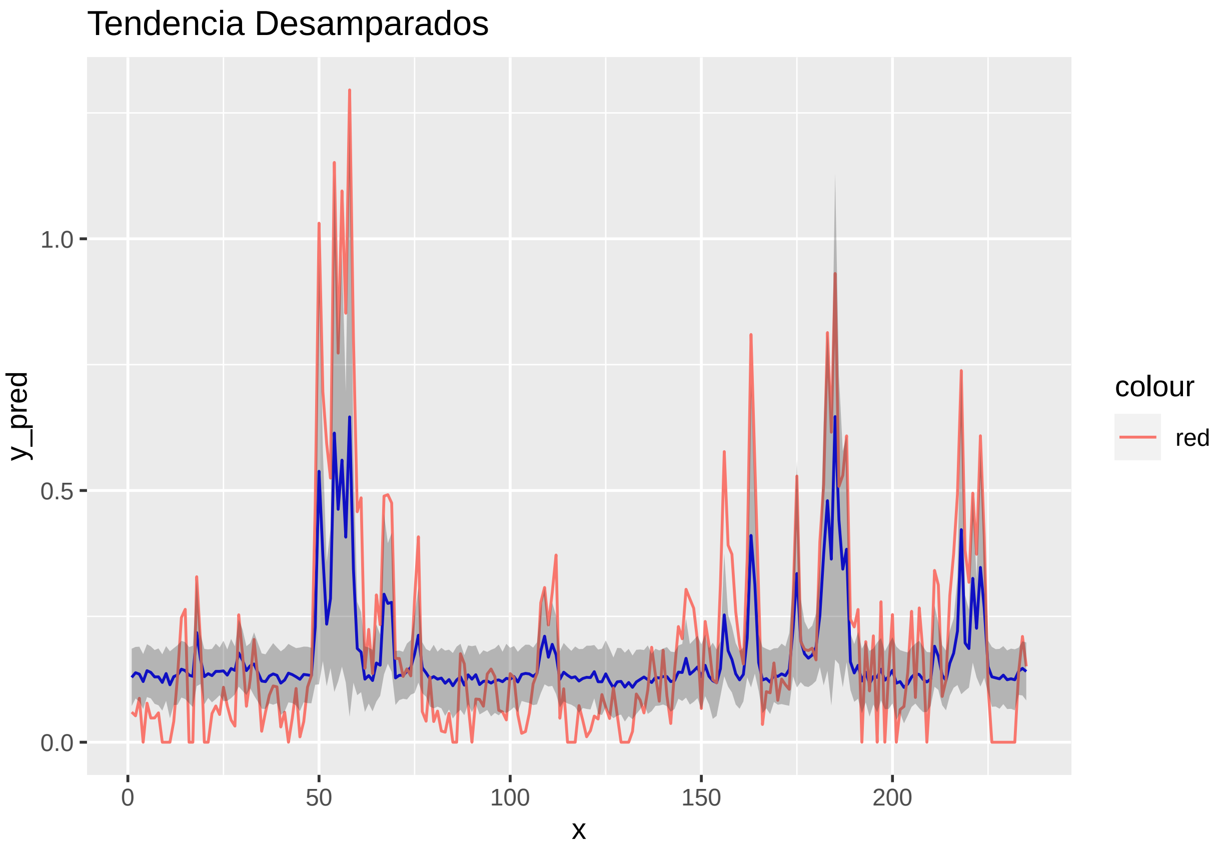

```
##
## [[5]]
```


Tendencia Carrillo
40
30
20
10
0
y_pred
0
50
100
150
200
x
colour
red

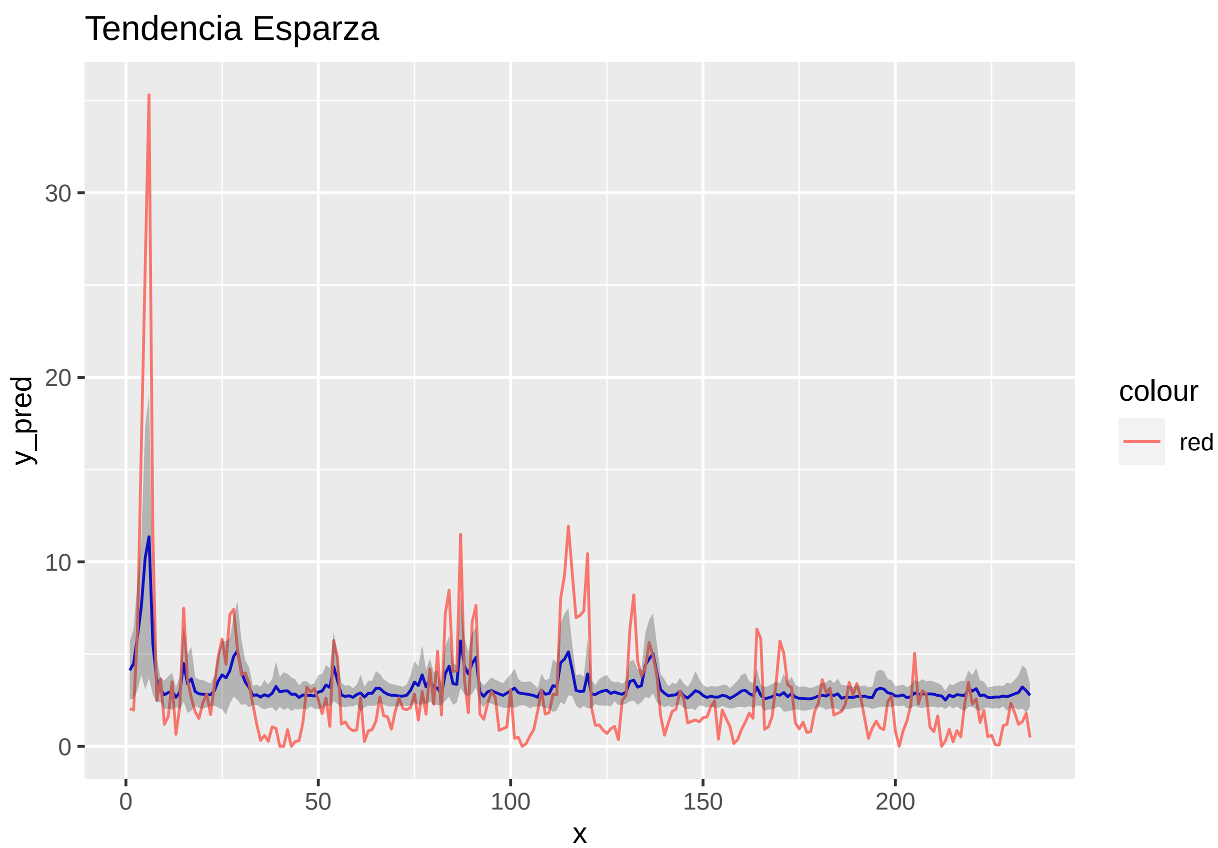
```
##
## [[6]]
```


Tendencia Corredores
y_pred
x
colour
red
0
5
10
15
20
0
50
100
150
200

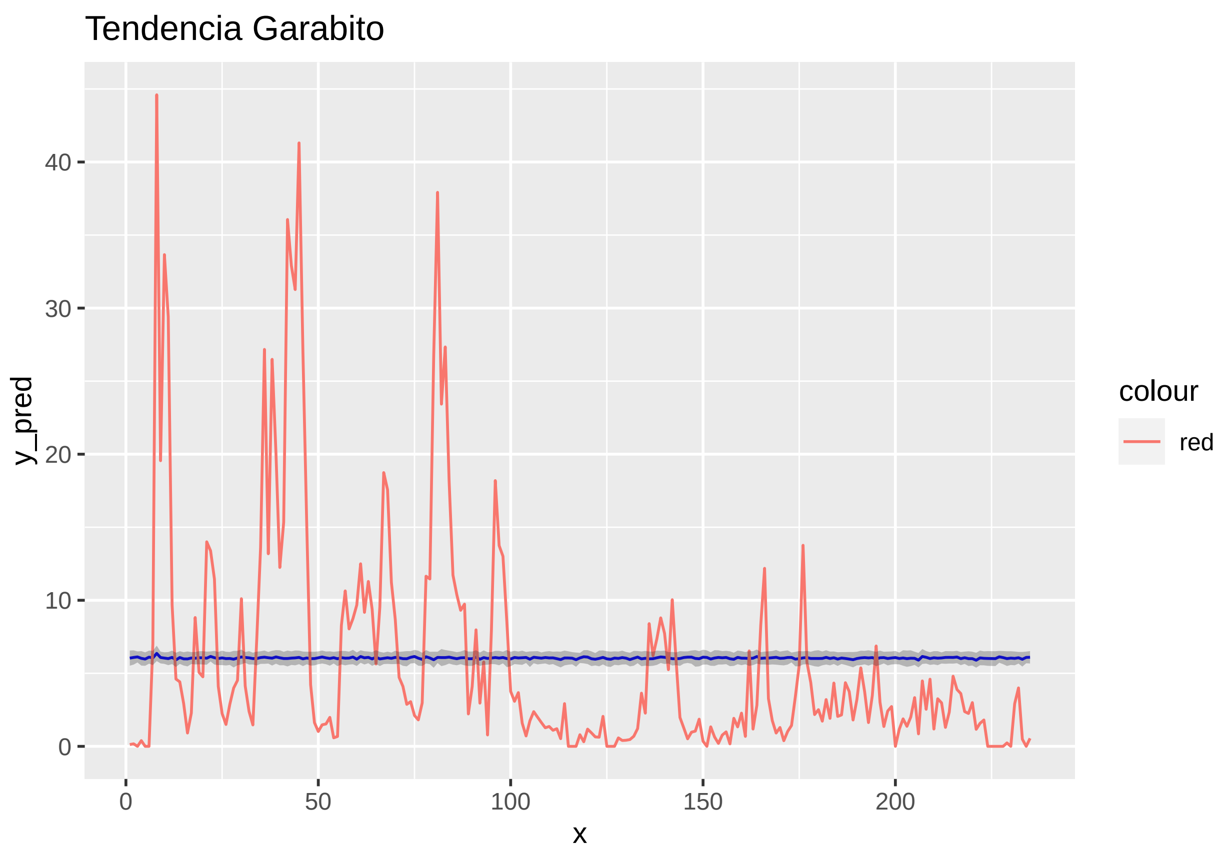
```
##
## [[7]]
```


Tendencia Desamparados
y_pred
1.0
0.5
0.0
0
50
100
150
200
x
colour
red

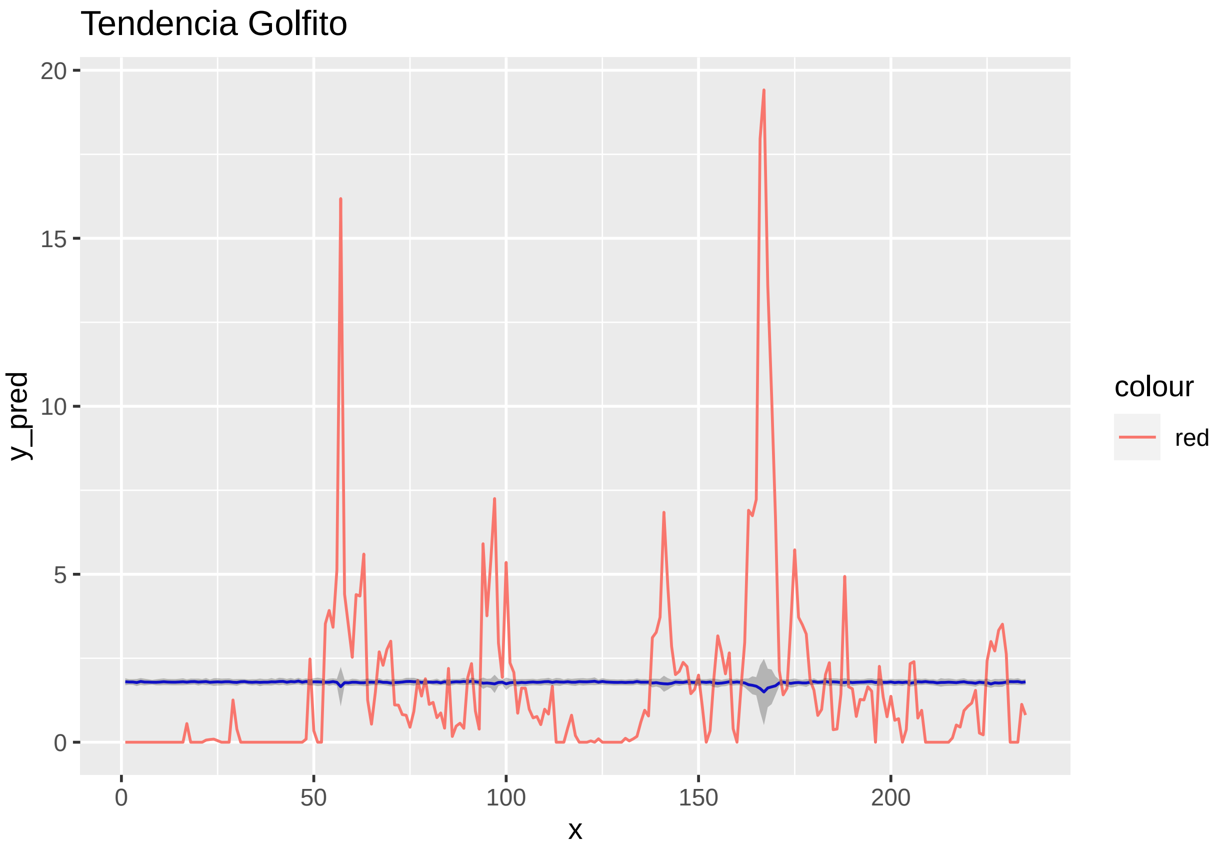
```
##
## [[8]]
```


Tendencia Esparza
y_pred
30
20
10
0
0
50
100
150
200
x
colour
red

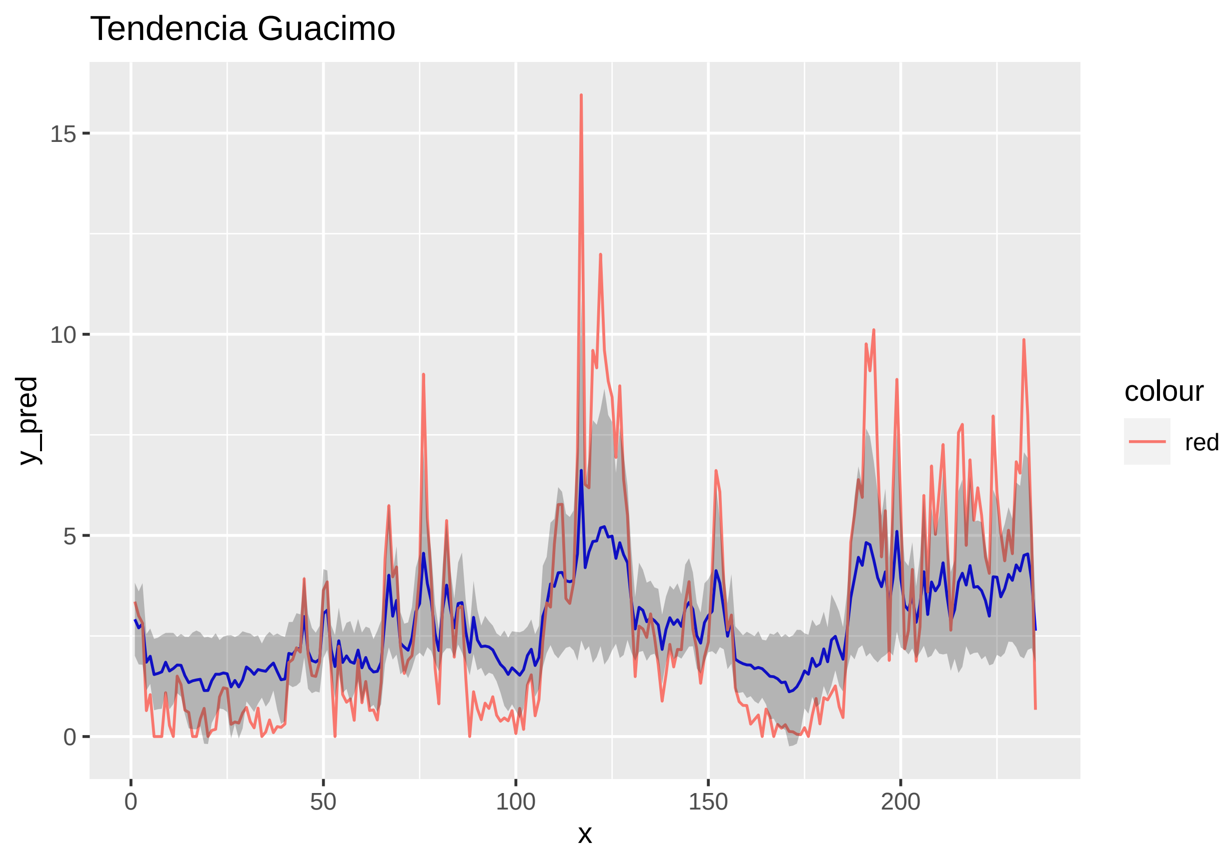
```
##
## [[9]]
```


Tendencia Garabito
y_pred
x
colour
red
0
10
20
30
40
0
50
100
150
200

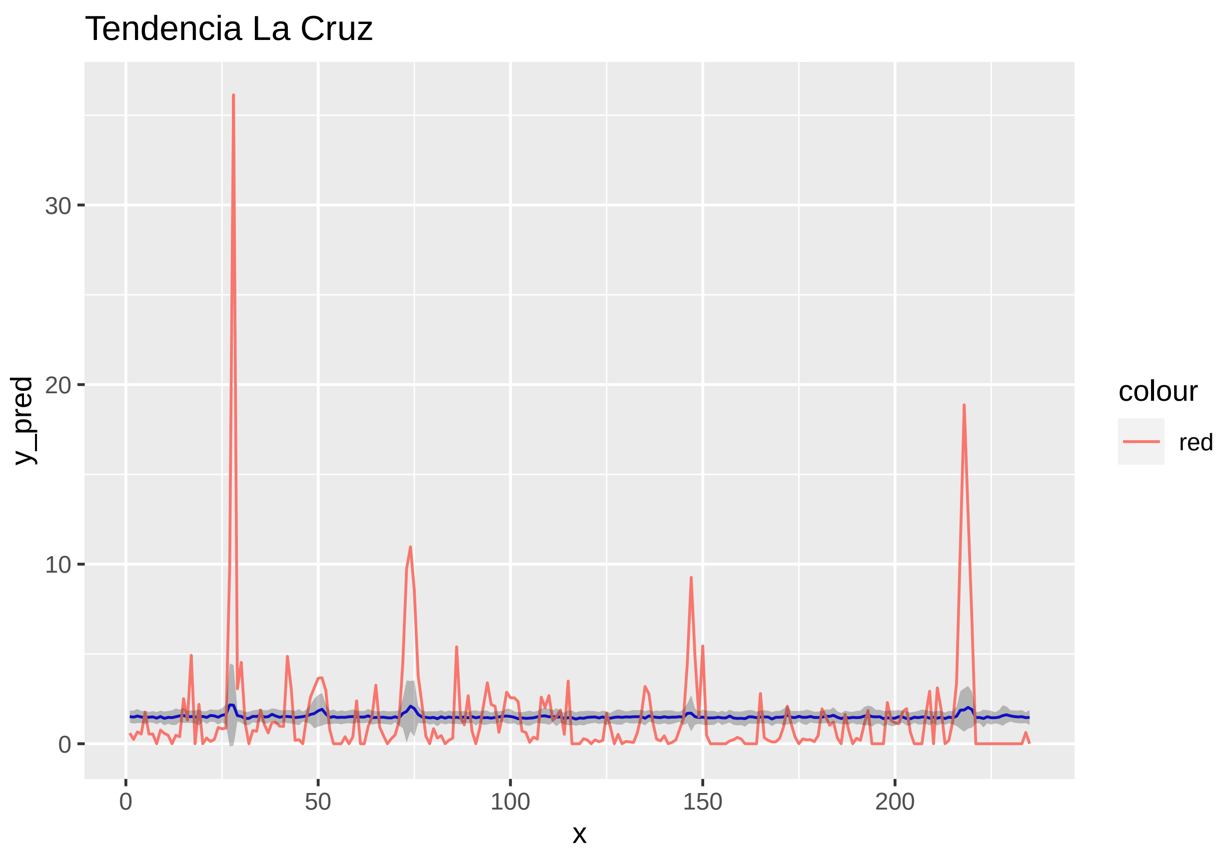
```
##
## [[10]]
```


Tendencia Golfito
y_pred
x
colour
red
0
5
10
15
20
0
50
100
150
200

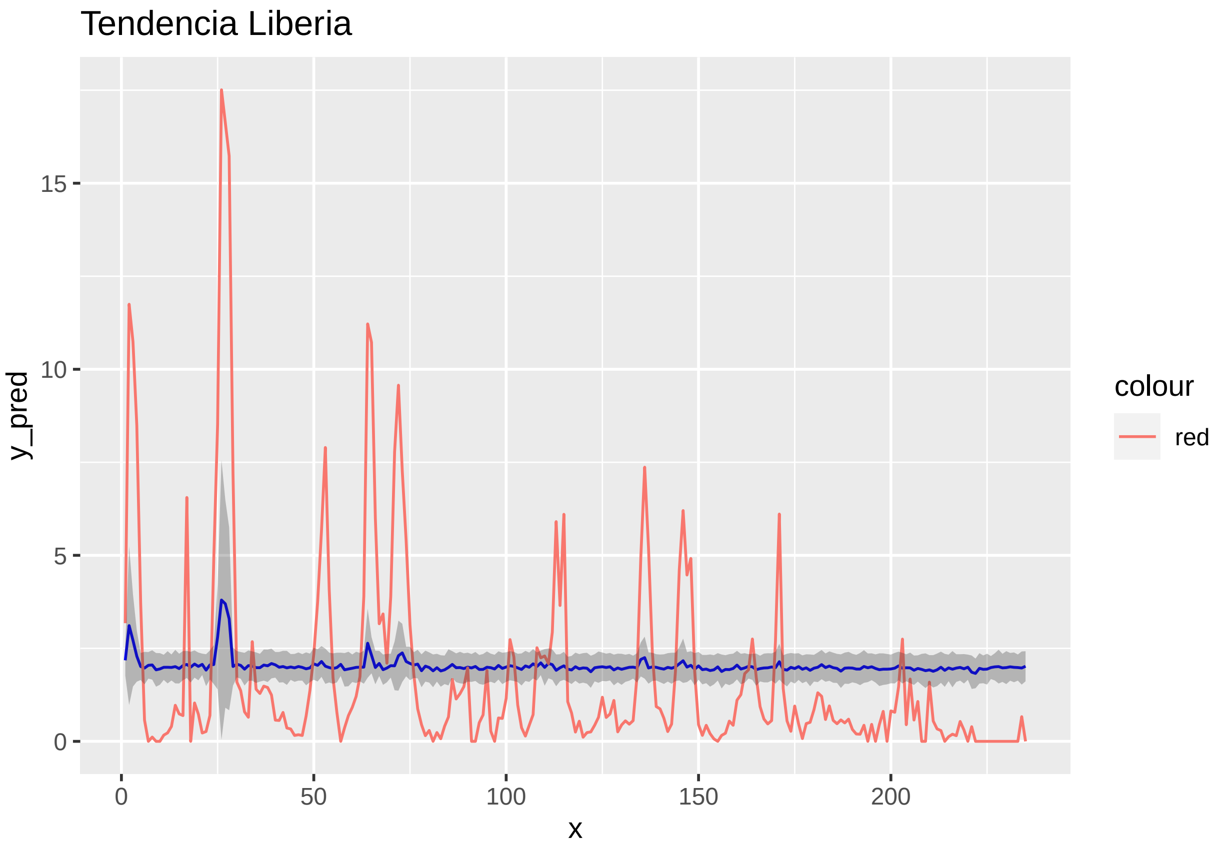
```
##
## [[11]]
```



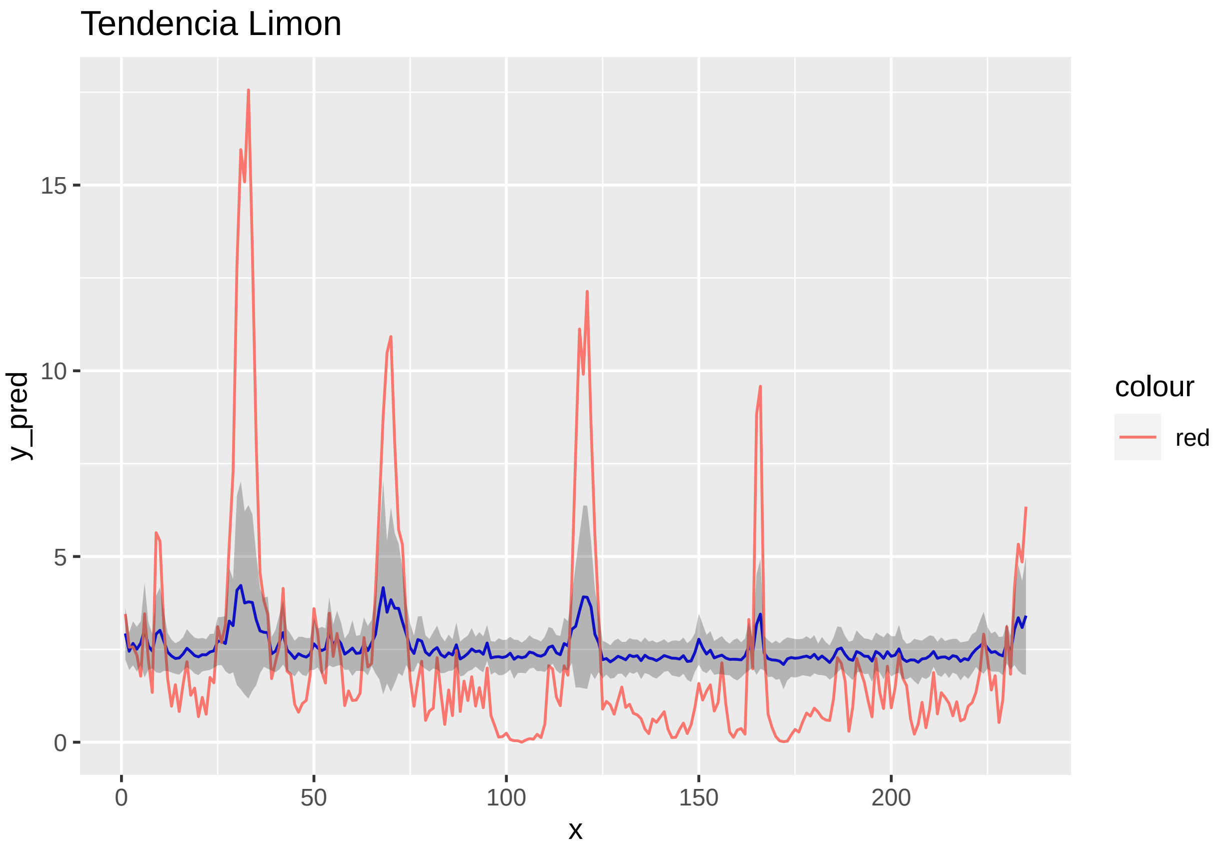

```
##
## [[12]]
```



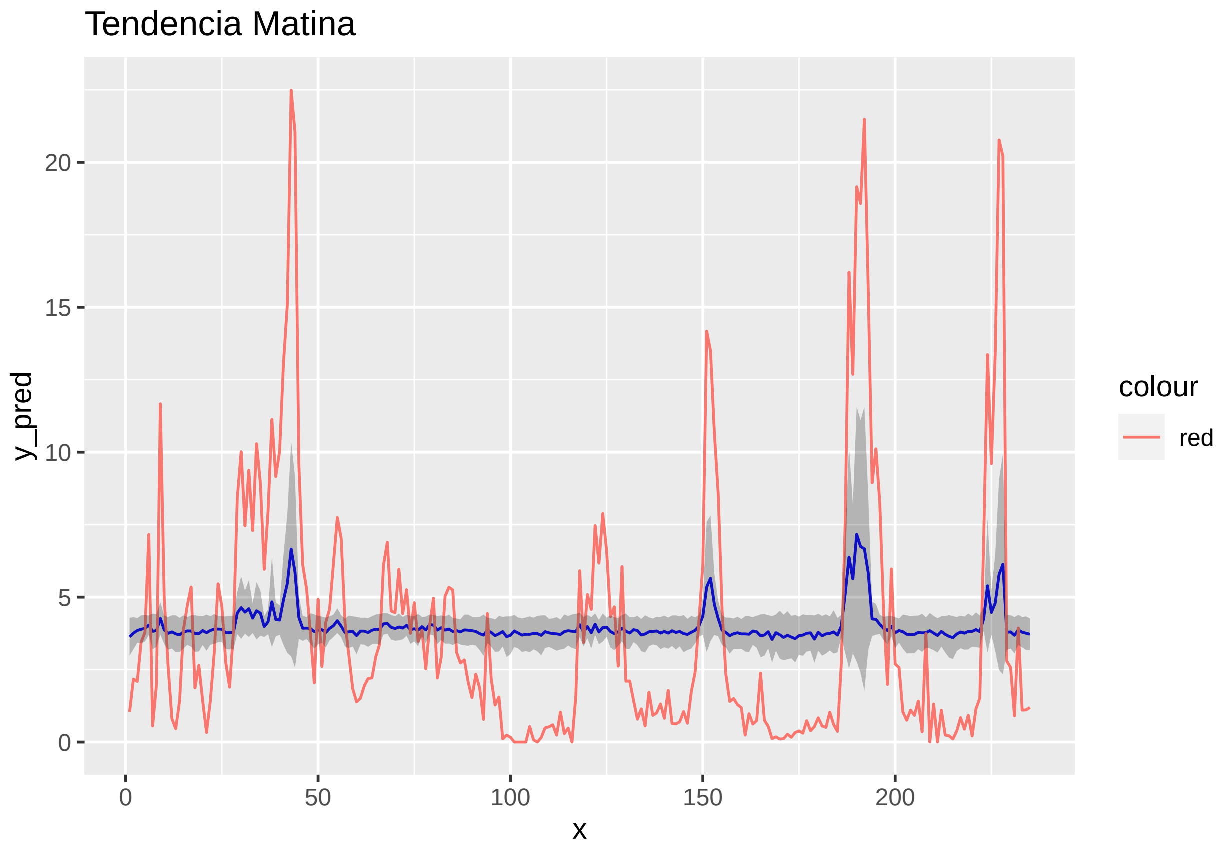

```
##
## [[13]]
```



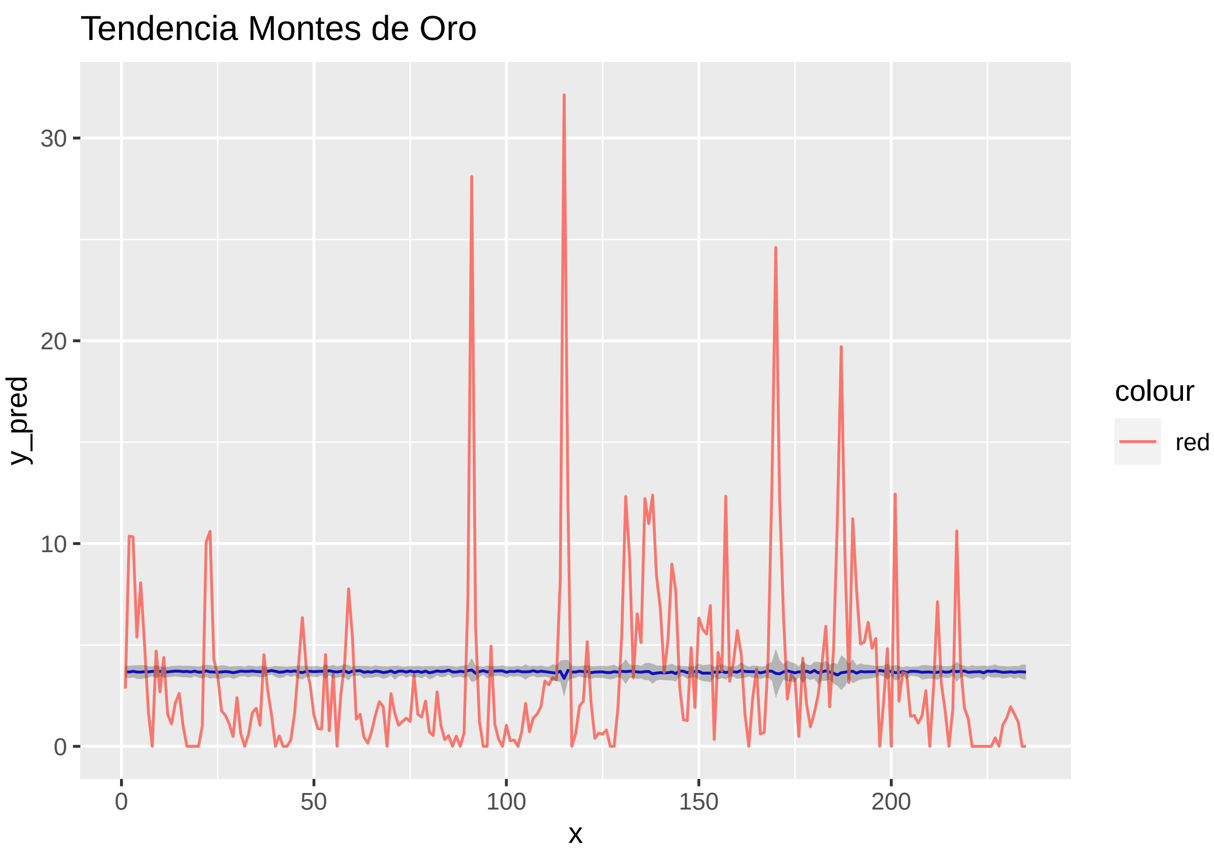

```
##
## [[14]]
```



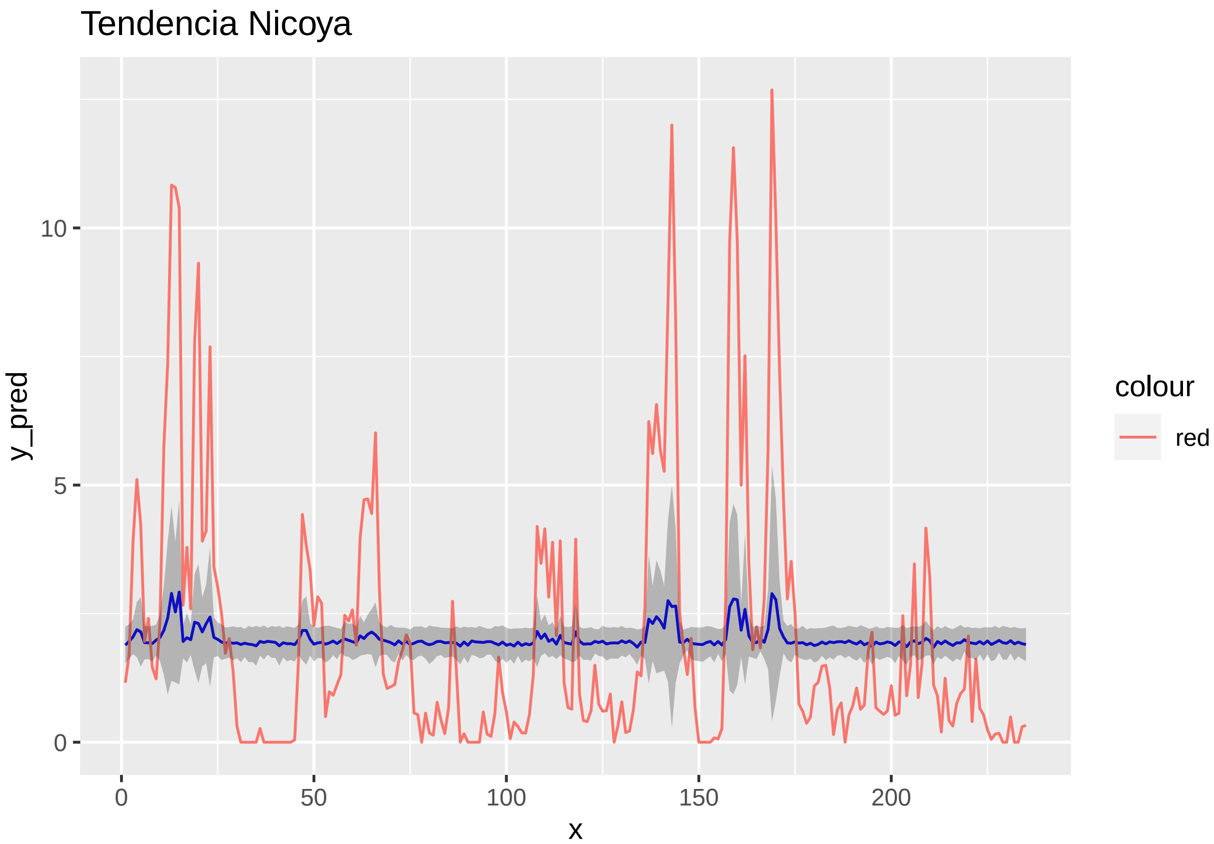

```
##
## [[15]]
```


Tendencia Matina
y_pred
x
colour
red
0
5
10
15
20
0
50
100
150
200

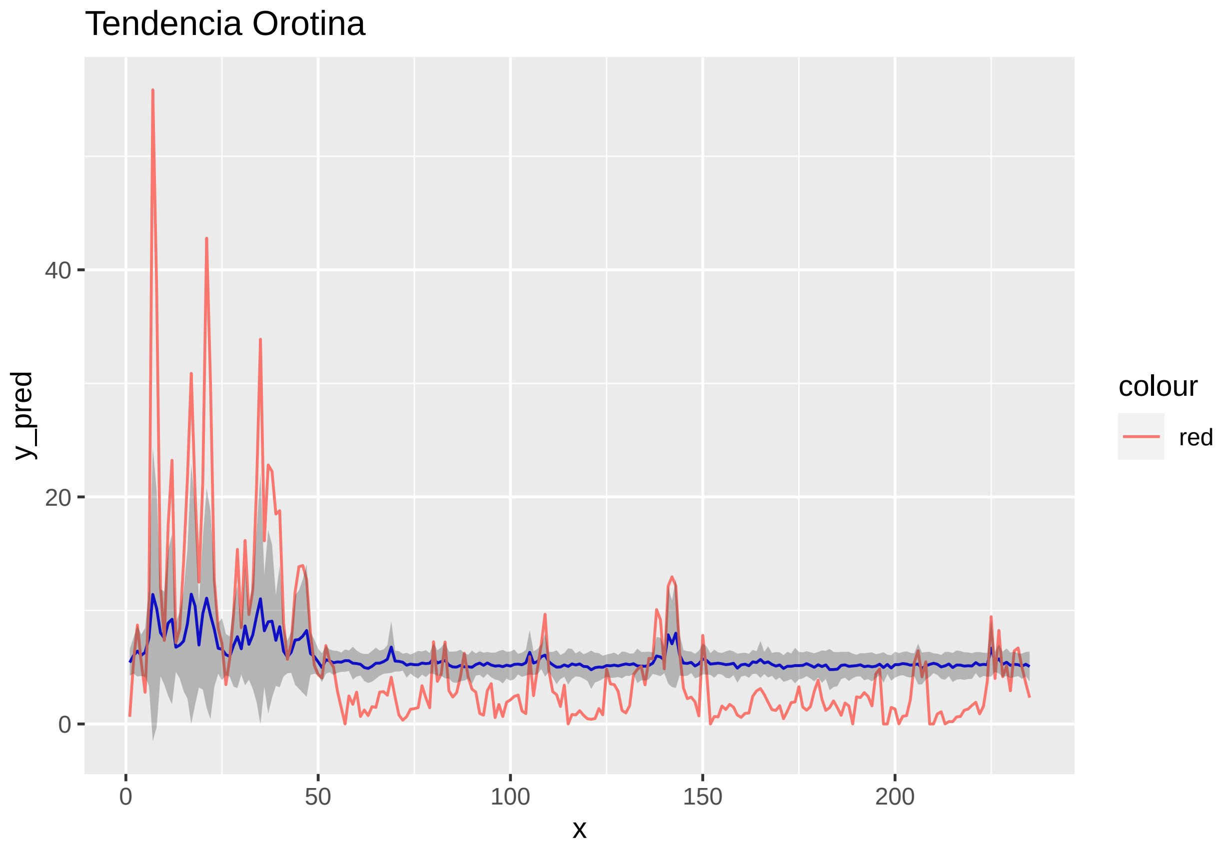
```
##
## [[16]]
```


Tendencia Montes de Oro
y_pred
x
colour
red
0
10
20
30
0
50
100
150
200

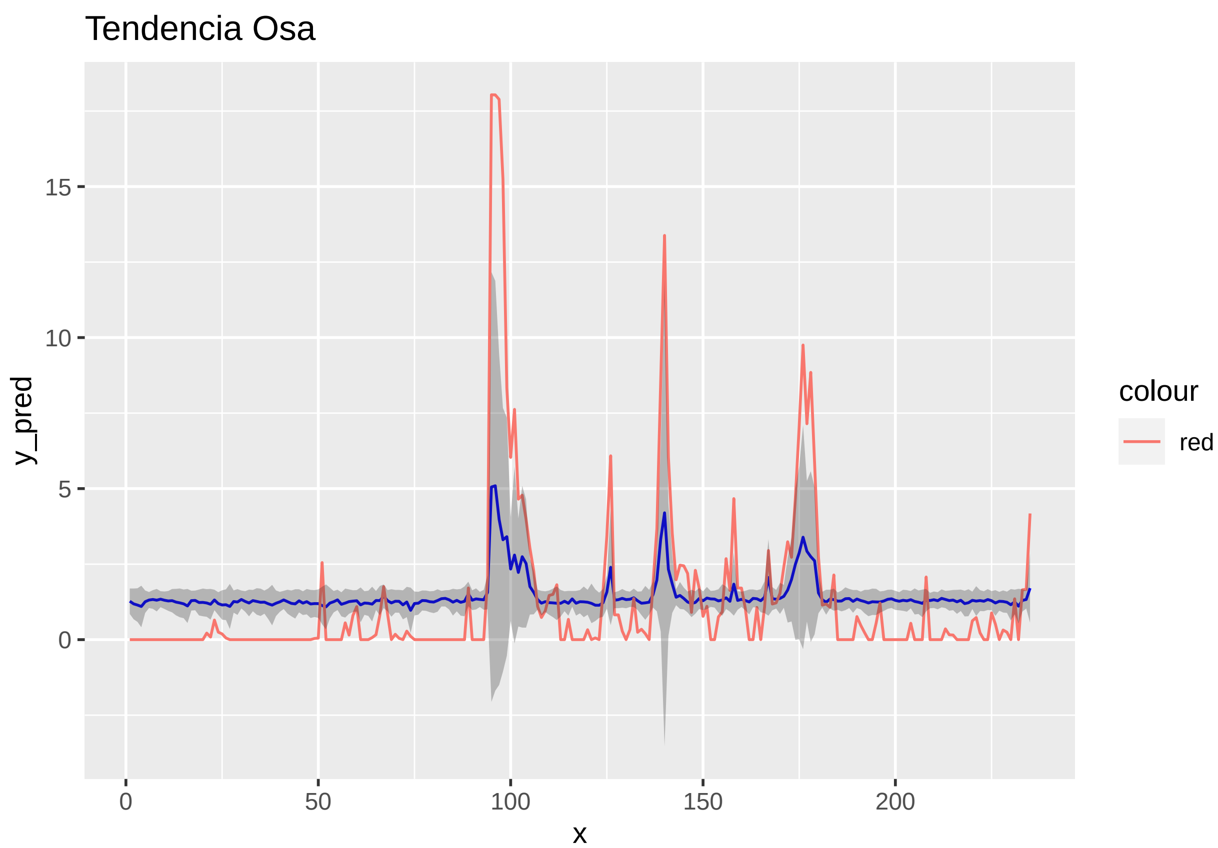
```
##
## [[17]]
```



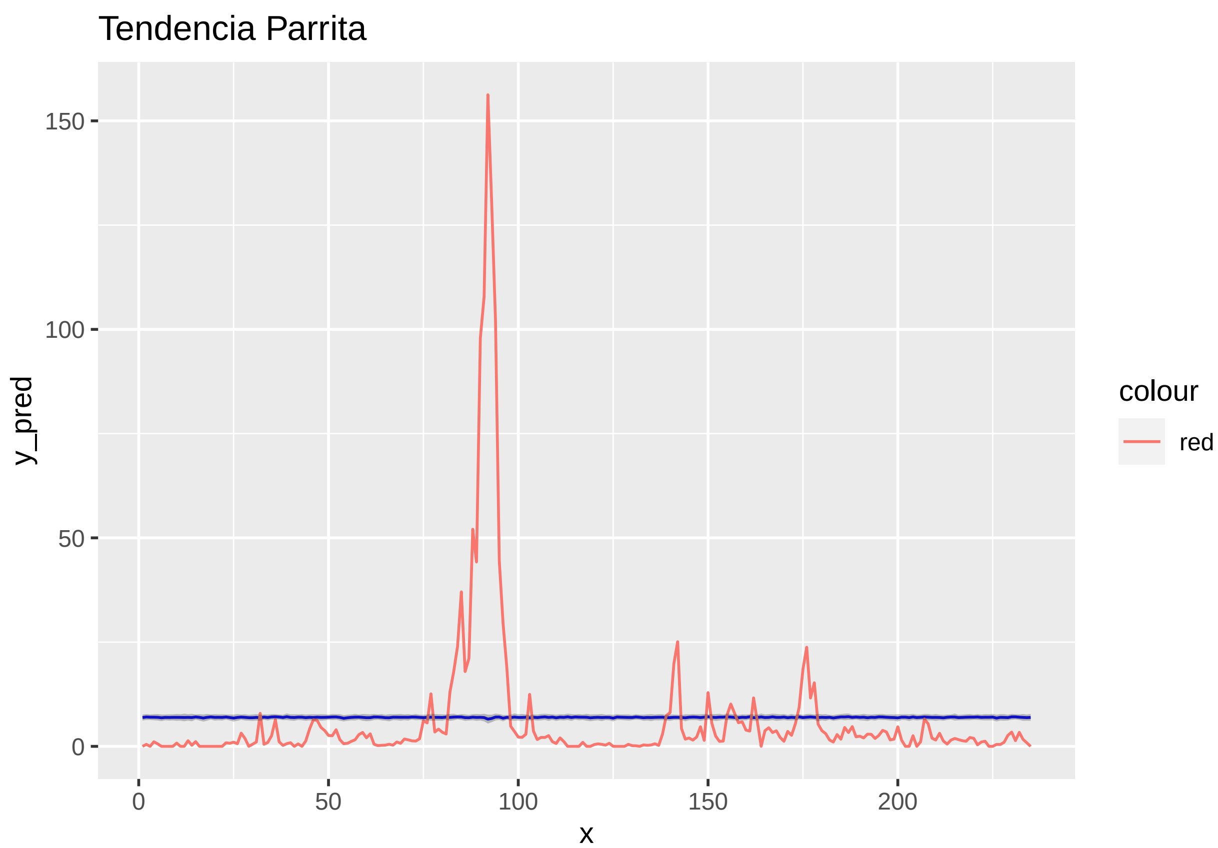

```
##
## [[18]]
```



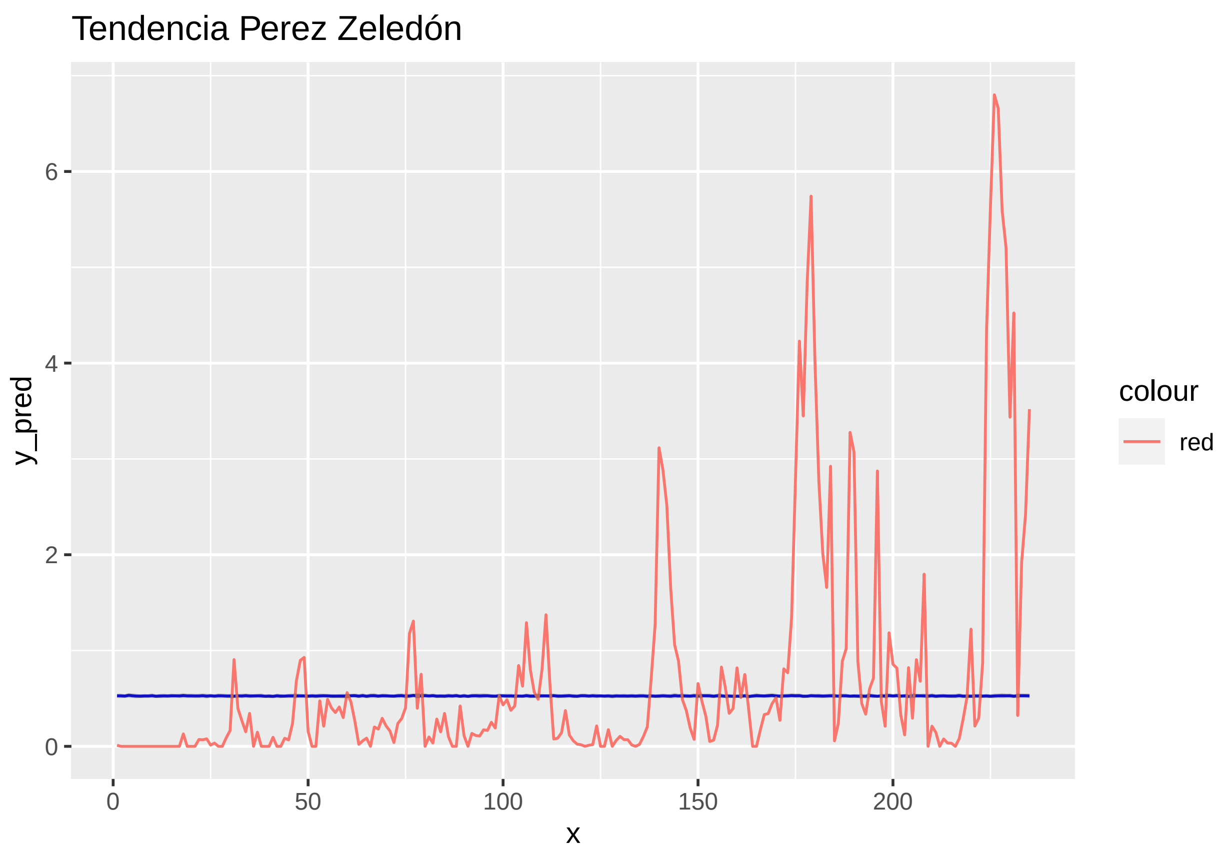

```
##
## [[19]]
```



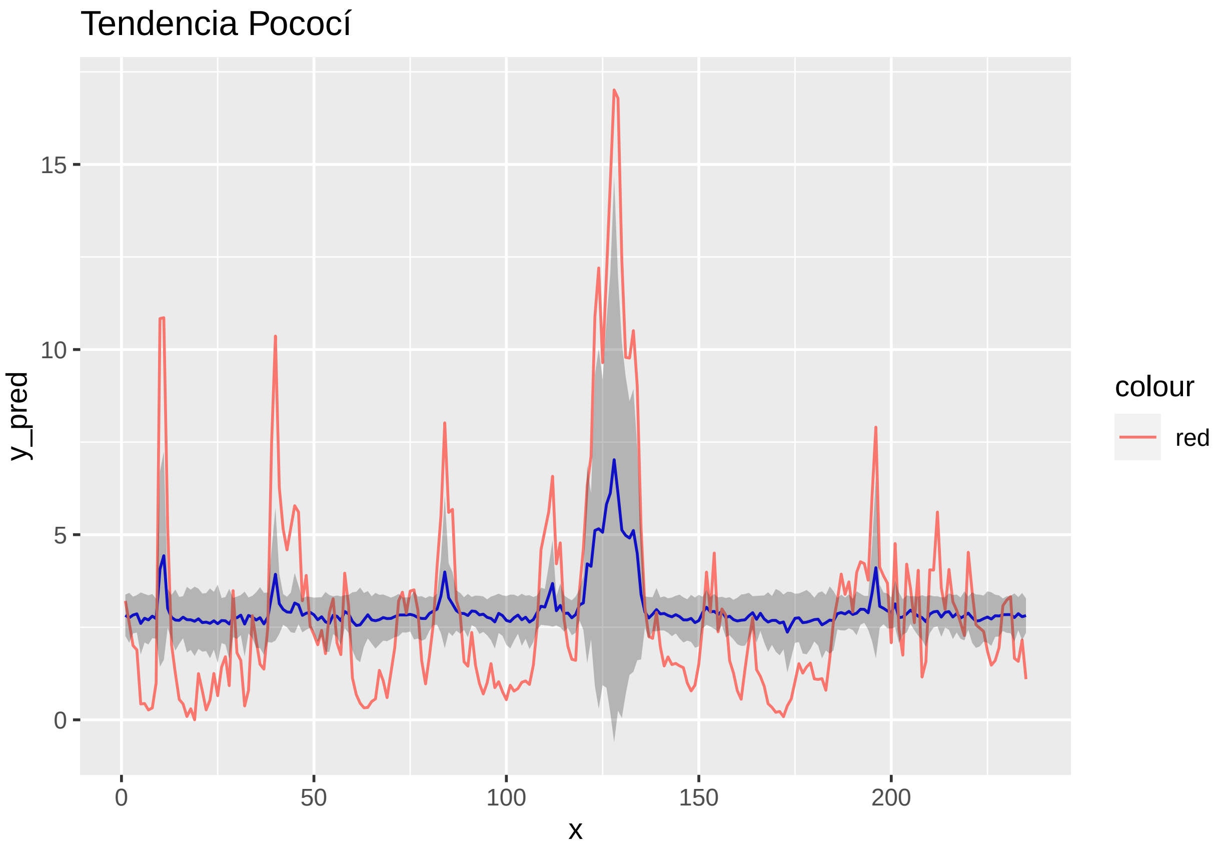

```
##
## [[20]]
```


Tendencia Parrita
y_pred
150
100
50
0
0
50
100
150
200
x
colour
red

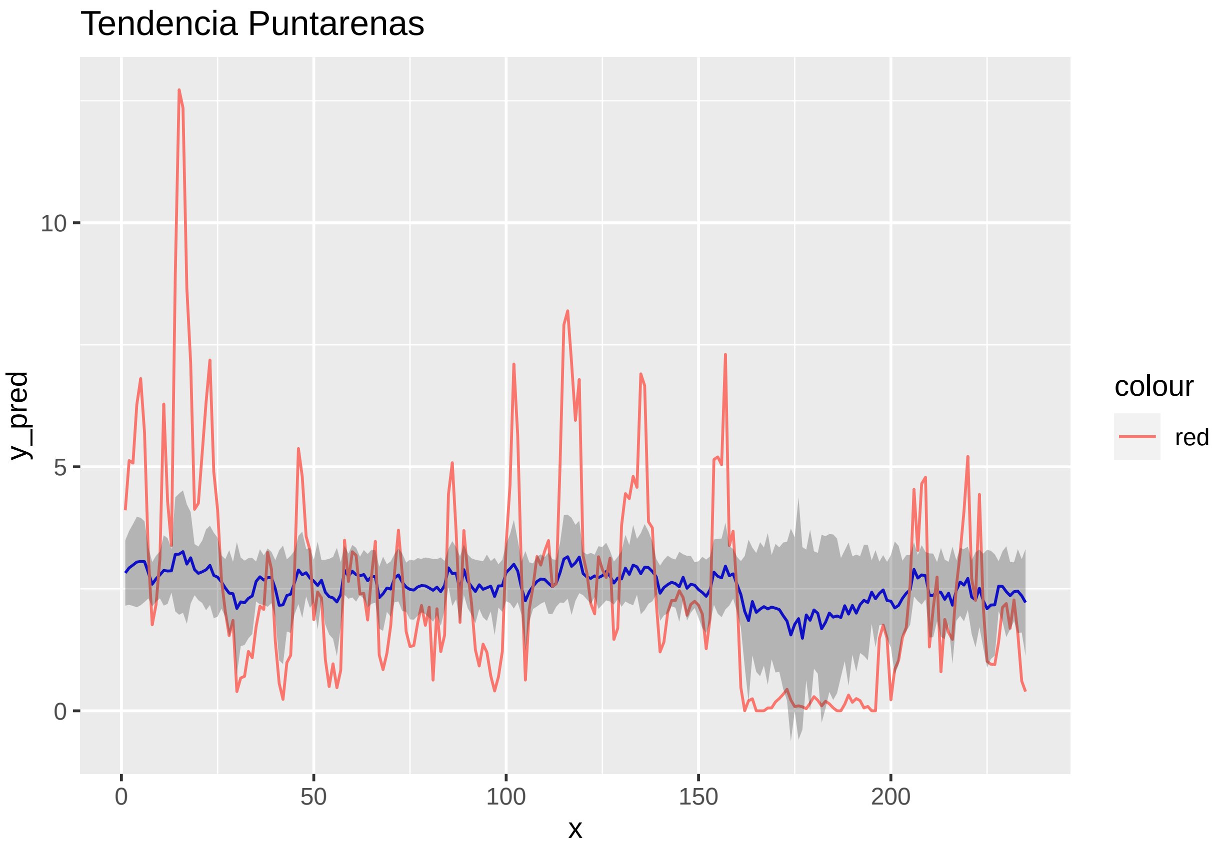
```
##
## [[21]]
```


Tendencia Perez Zeledón
y_pred
x
colour
red
0
2
4
6
0
50
100
150
200

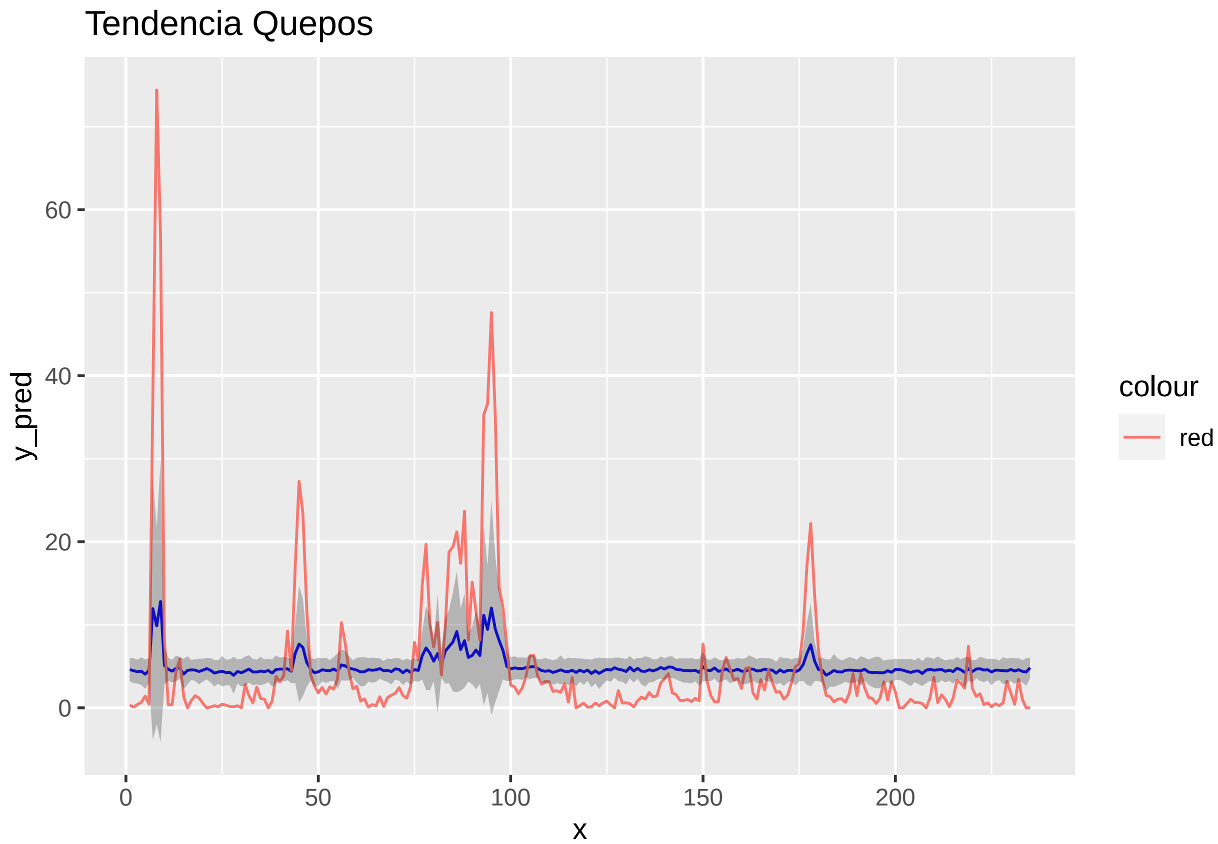
```
##
## [[22]]
```



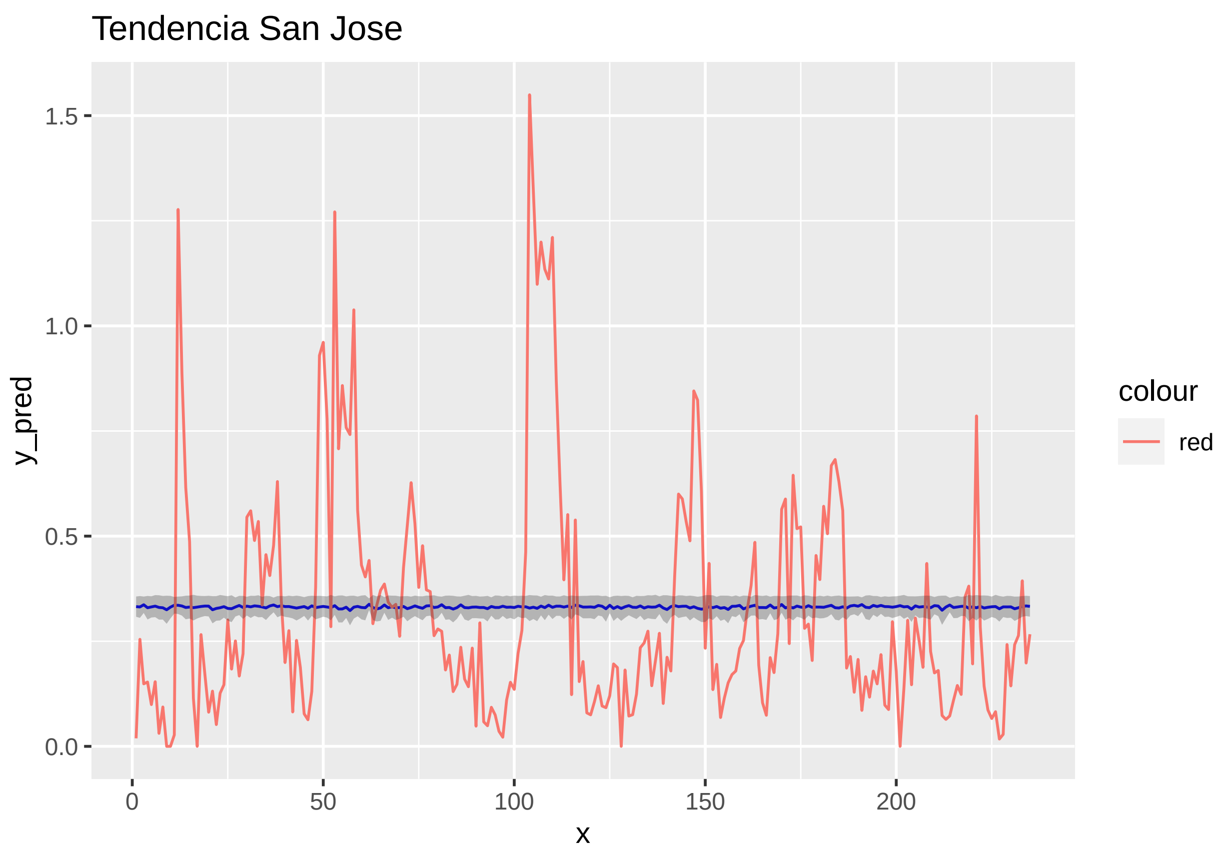

```
##
## [[23]]
```


Tendencia Puntarenas
y_pred
10
5
0
0
50
100
150
200
x
colour
red

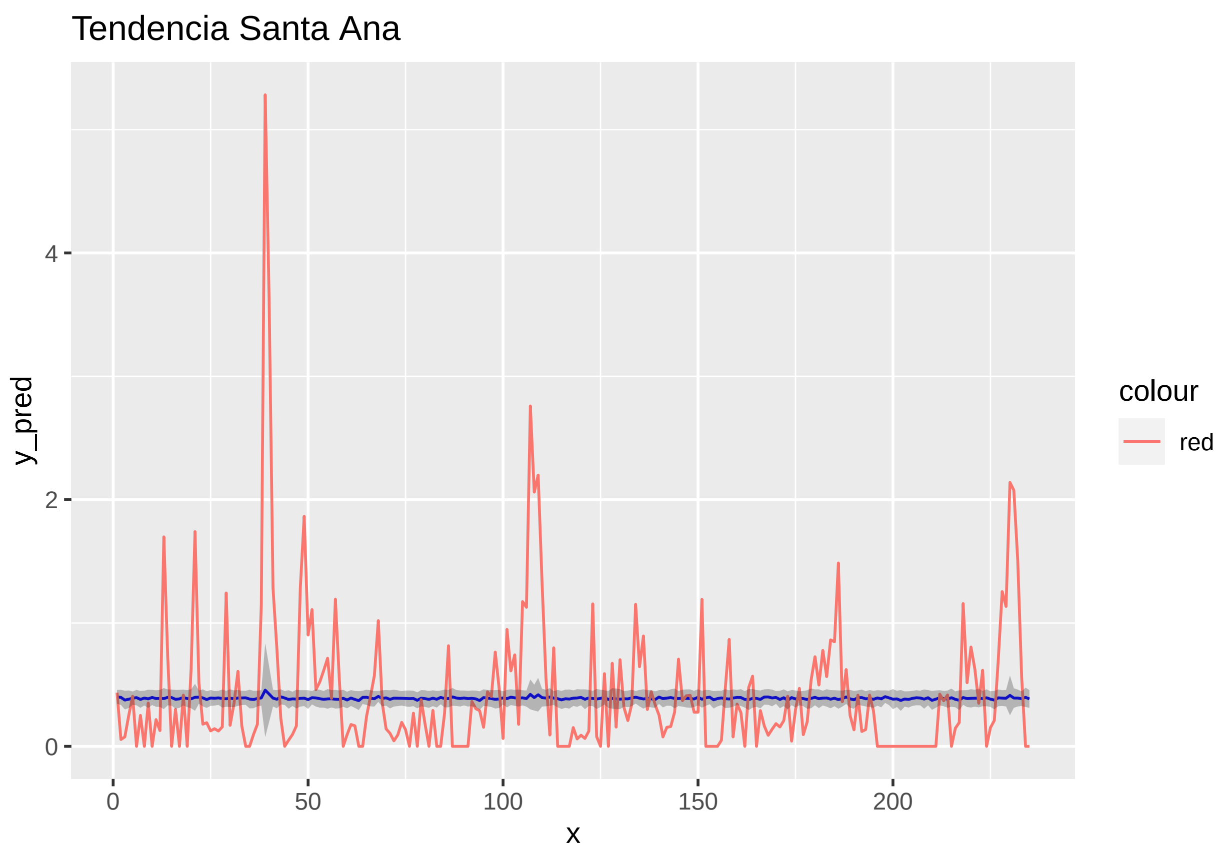
```
##
## [[24]]
```



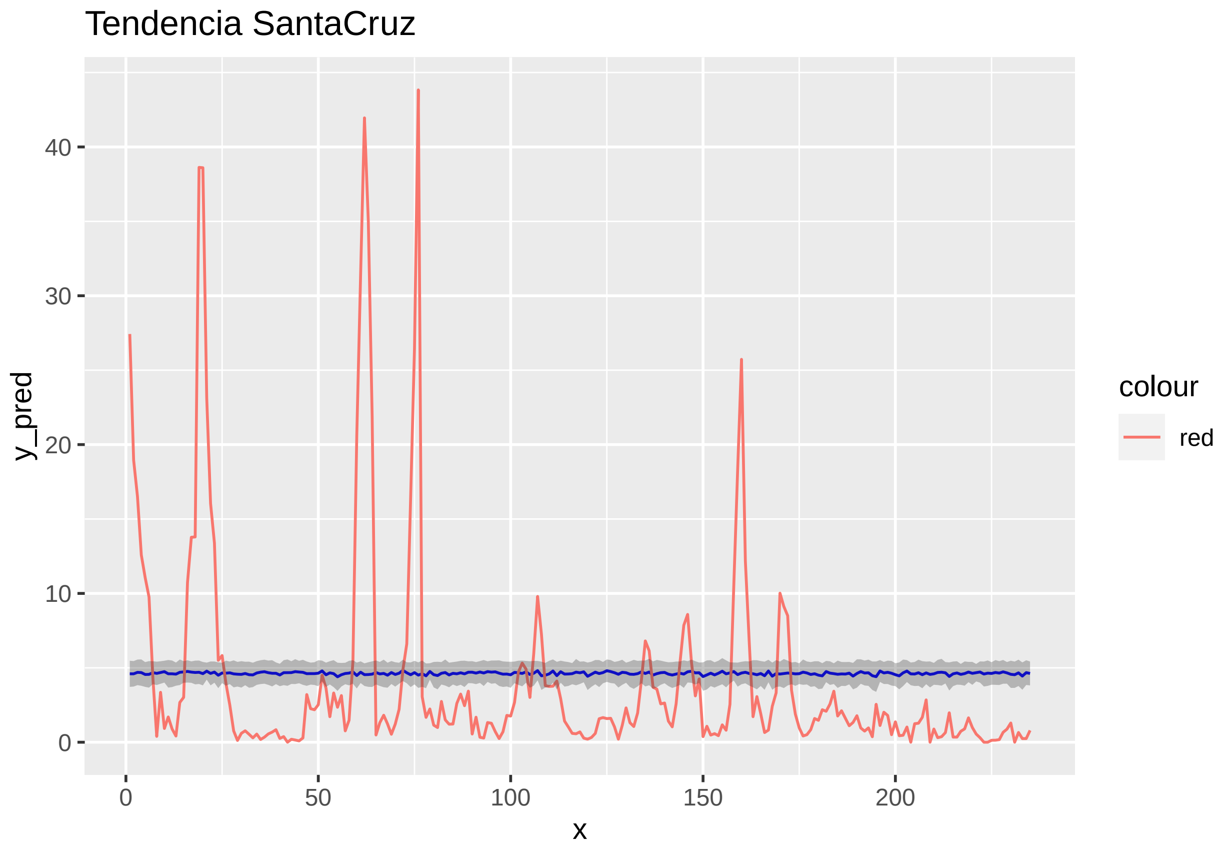

```
##
## [[25]]
```


Tendencia San Jose
y_pred
1.5
1.0
0.5
0.0
0
50
100
150
200
x
colour
red

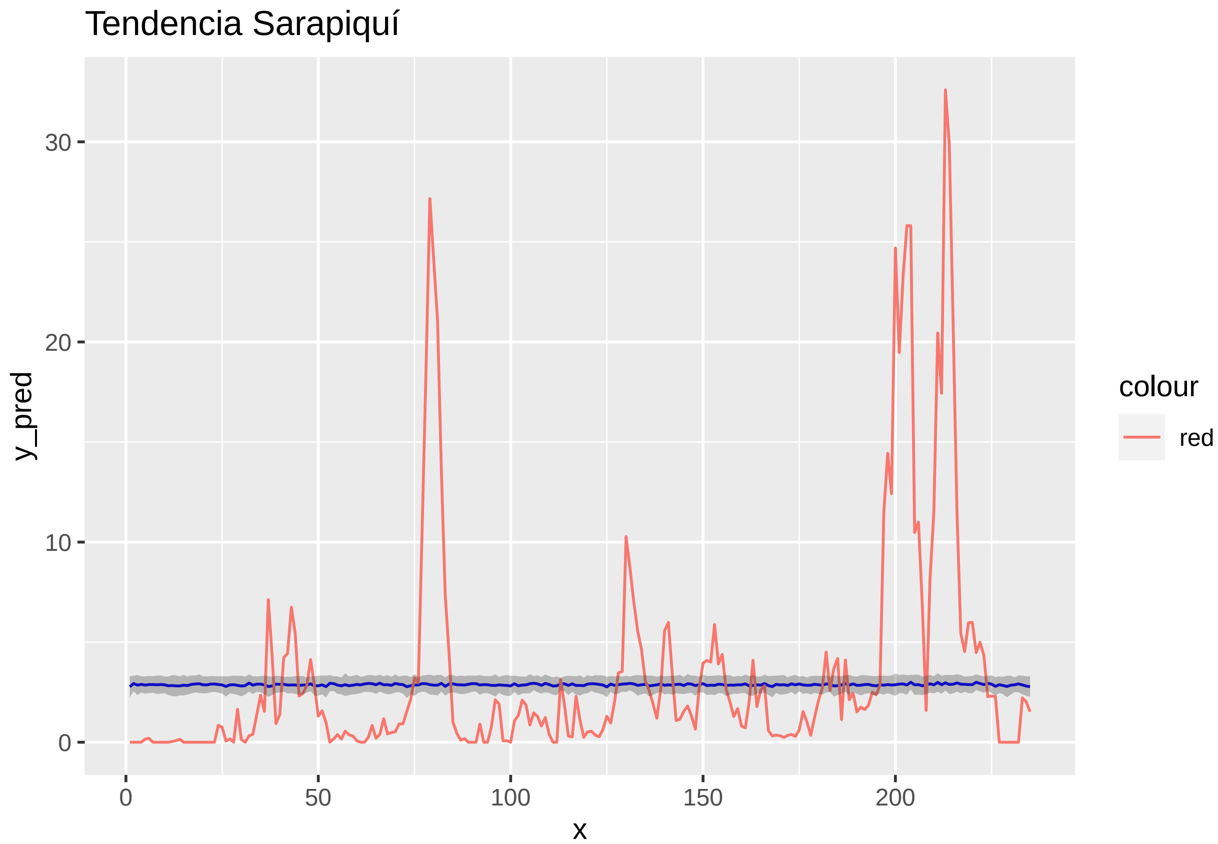
```
##
## [[26]]
```



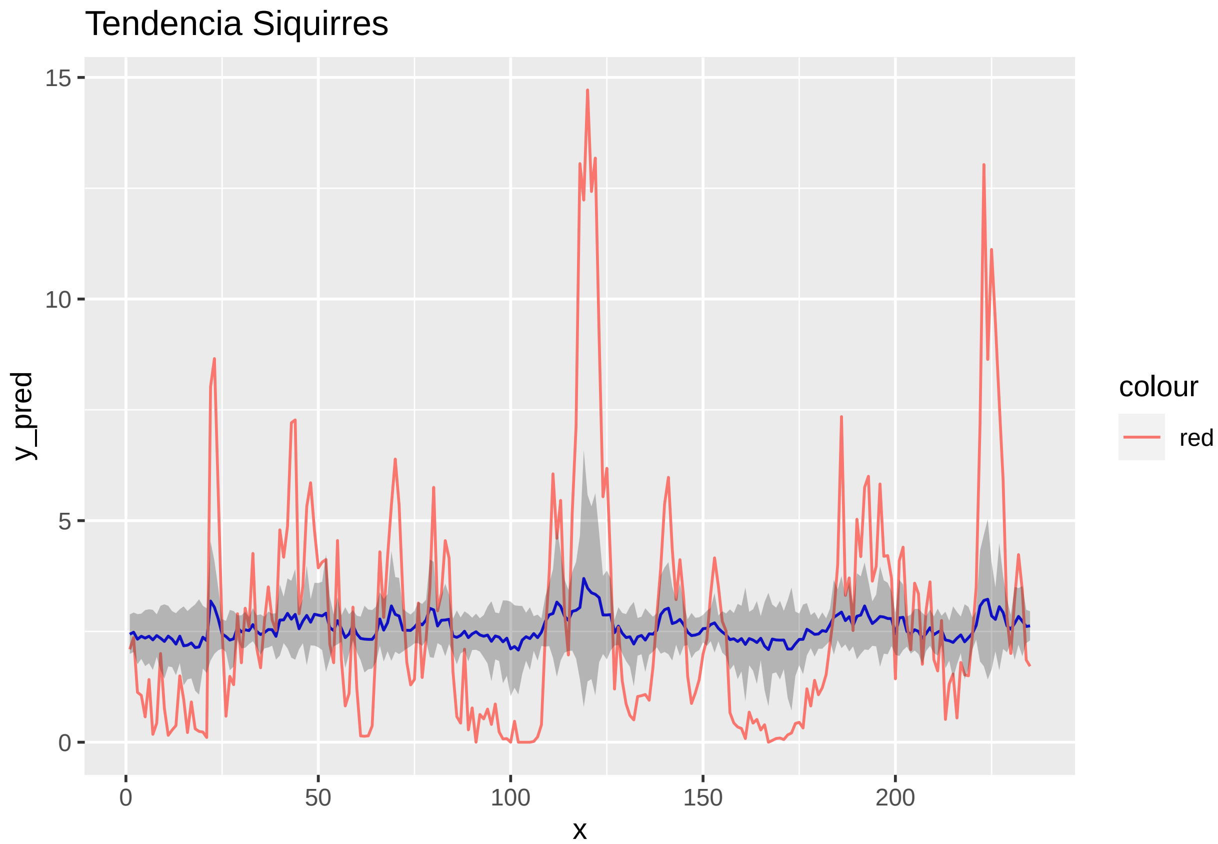

```
##
## [[27]]
```


Tendencia SantaCruz
y_pred
40
30
20
10
0
0
50
100
150
200
x
colour
red

```
##
## [[28]]
```

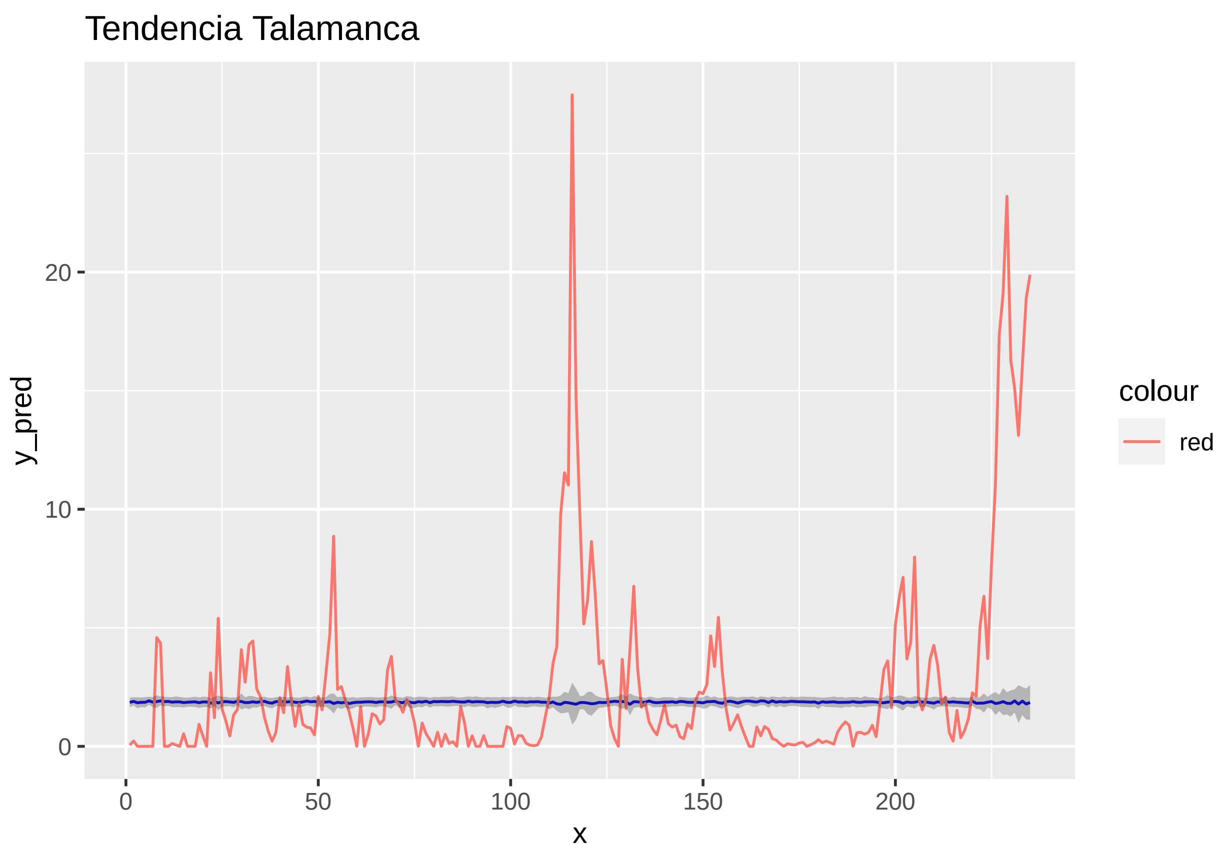

Tendencia Sarapiquí
y_pred
x
colour
red
0
10
20
30
0
50
100
150
200

```
##
## [[29]]
```

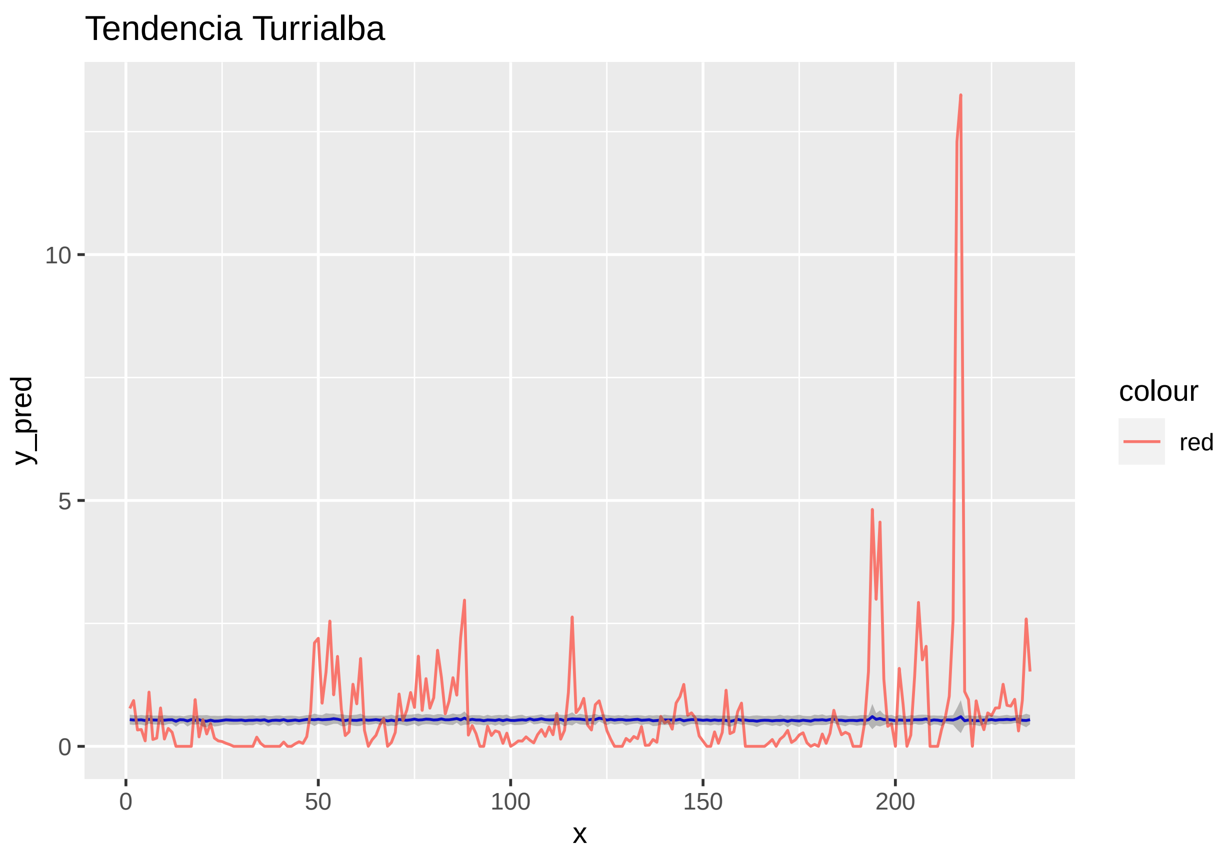


Tendencia Siquirres
y_pred
x
colour
red
15
10
5
0
0
50
100
150
200

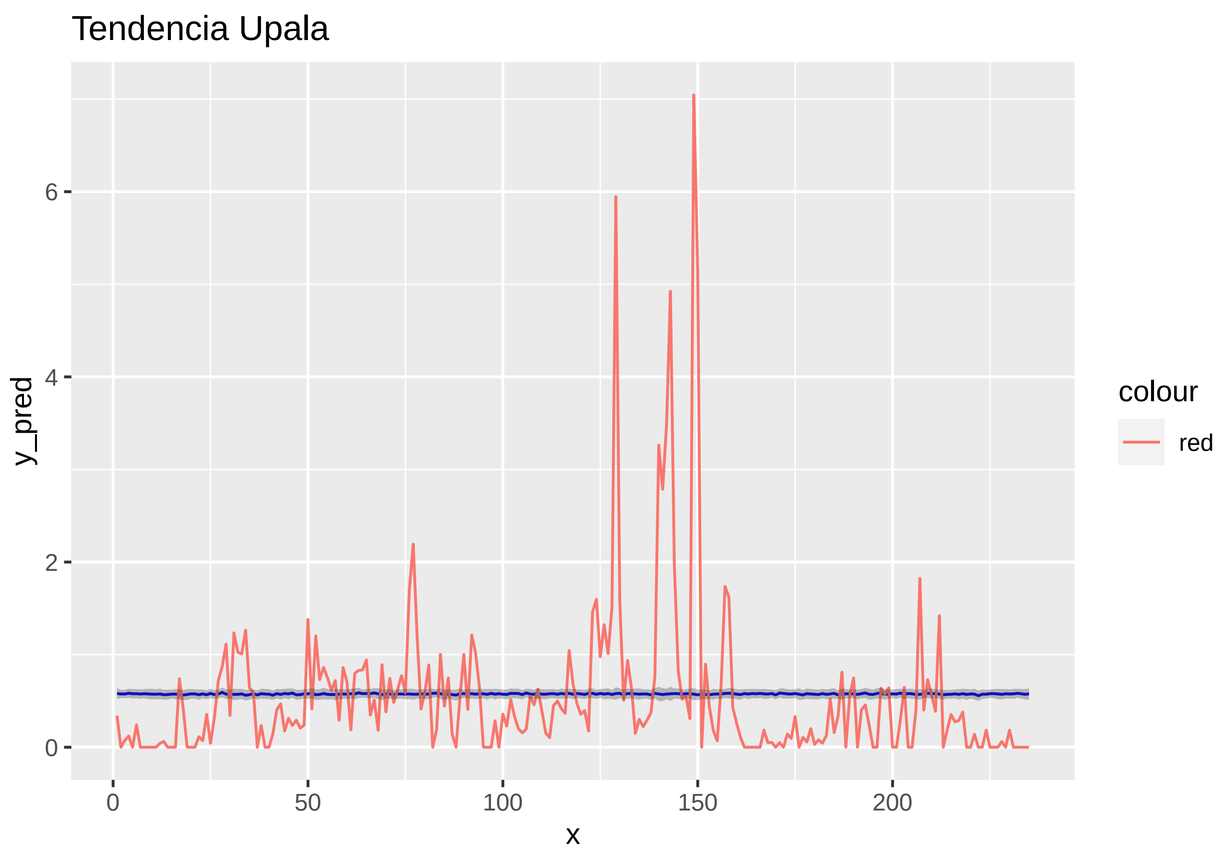

```
##
## [[30]]
```


Tendencia Talamanca
y_pred
0
10
20
0
50
100
150
200
x
colour
red

```
##
## [[31]]
```




```
##
## [[32]]
```



Tendencia Upala
y_pred
6
4
2
0
0
50
100
150
200
x
colour
red