Analisis overfitting

Jimena Murillo

2022-06-15

Paquetes

```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
```

```
library(neuralnet)
```

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

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

Datos

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


Alajuela <- basecanton %>% filter(Canton == "Alajuela") %>%

  dplyr::select(Year,Month,Nino12SSTA, Nino3SSTA, Nino4SSTA,Nino34SSTA,Nino34SSTA1, Nino34SSTA2, Nino34S

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


if(anyNA(Alajuela)){
  Alajuela <- na.omit(Alajuela)
}

#Escala


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

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

Alajuela2 <- apply(Alajuela, 2, normalize)

Fecha = paste(Alajuela$Year, Alajuela$Month)
Fechas_nom = c(2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012, 2011)
```

```
model.gen = function(X_train,y_train, X_test){

## Generar un Wrapper para el learning dropout
# R6 wrapper class, a subclass of KerasWrapper
ConcreteDropout <- R6::R6Class("ConcreteDropout",

  inherit = KerasWrapper,

  public = list(
```

```r
  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
```

```
  ))
}



# prior length-scale
l <- 1e-4
# initial value for weight regularizer
wd <- l^2/230
# initial value for dropout regularizer
dd <- 2/230



## Modelo Dropout


# we use one-dimensional input data here, but this isn't a necessity
input_dim <- 32
# this too could be > 1 if we wanted
output_dim <- 1


input <- layer_input(shape = input_dim)

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
)


## 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)


## Entrenar al modelo


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

history <- model %>% fit(
  X_train,
  y_train,
  epochs = 100,
  batch_size = 18,
```

```r
  validation_split = 0.1,
  shuffle = F
)


num_MC_samples <- 300

samples = list()

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

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

MC_samples.tot <- array(0, dim = c(num_MC_samples, nrow(Alajuela2[,-33]), 2 * output_dim))
for (k in 1:num_MC_samples) {
  MC_samples.tot[k, , ] <- denorm((model %>% predict(Alajuela2[,-33])))
}


samples[[1]] <- MC_samples.pd
samples[[2]] <- MC_samples.tn
samples[[3]] <- MC_samples.tot

return (samples)

}
```

```r
#Train y test

Fechas = c(1, 0.95, 0.90, 0.85, 0.80, 0.75, 0.70, 0.65, 0.60, 0.55, 0.50)


Eval.pd = matrix(NA, nrow = length(Fechas), ncol = 2)
Eval.tn = matrix(NA, nrow = length(Fechas), ncol = 2)
Eval.tot = matrix(NA, nrow = length(Fechas), ncol = 2)

p1 = list()
p2 = list()
p3 = list()

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

data_train = as.data.frame(Alajuela2) %>% filter(Year < Fechas[i])#PARA ENTRENAR HASTA 2018
data_test = as.data.frame(Alajuela2) %>% filter(Year >= Fechas[i])

X_train = as.matrix(data_train[,-ncol(data_train)])
y_train = as.matrix(data_train[,ncol(data_train)])
```

```
X_test = as.matrix(data_test[,-ncol(data_test)])
y_test = as.matrix(data_test[,ncol(data_test)])

samples = list()
samples = model.gen (X_train, y_train, X_test)

## Generar intervalo de confianza
output_dim  = 1

MC_samples.pd = samples[[1]]

means <- NULL
means <- MC_samples.pd[, , 1:output_dim]
# average over the MC samples
predictive_mean <- apply(means, 2, mean)

epistemic_uncertainty <- apply(means, 2, var)

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

y_test = denorm(y_test)


df1 <- data.frame(
  x = Fecha[(236-nrow(X_test)):235],
  y = y_test,
  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)
)



everyother1 <- function(x) x[(seq_along(Fecha) + 5)%%12 == 6]


p1[[i]] = ggplot(df1, aes(x = x, y = y, group = 1)) + geom_line(colour = "blue") +
  geom_line( aes(x = x, y = y_pred, colour = "red"))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_blank(), axis.text.x = element_text(angle = 45), legend.position = "none" )
  scale_x_discrete(breaks = everyother1) + labs (x = "Fecha", y = "Riesgo Relativo") +
  ggtitle(paste("Predicciones desde año:", Fechas_nom[i], sep = " "))+
  geom_ribbon(aes(ymin = u_overall_lower, ymax = u_overall_upper), alpha = 0.3) +
```

```r
  ylim(-3, 10)

metricas <- function(tabla){
  NRMSE <- mean((tabla$y_pred-tabla$y)^2)/mean(tabla$y)
  NIS_95 <- mean((tabla$u_overall_upper-tabla$u_overall_lower)+
                   (2/0.05)*(tabla$u_overall_lower-tabla$y)*(tabla$y<tabla$u_overall_lower)+
                   (2/0.05)*(tabla$y-tabla$u_overall_upper)*(tabla$y>tabla$u_overall_upper))/mean(tabla$
  return(data.frame(NRMSE,NIS_95))
}

Eval.pd[i, 1:2] = as.numeric(metricas(df1))





#### PREDICCIONES CON X_TRAIN / VALORES APROXIMADOS ####

MC_samples.tn = samples[[2]]

means <- NULL
means <- MC_samples.tn[, , 1:output_dim]
# average over the MC samples
predictive_mean <- apply(means, 2, mean)

epistemic_uncertainty <- apply(means, 2, var)


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


df2 <- data.frame(
  x = Fecha[1:nrow(X_train)],
  y = denorm(y_train),
  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)
)



p2[[i]] = ggplot(df2, aes(x = x, y = y, group = 1)) + geom_line(colour = "blue") +
  geom_line( aes(x = x, y = y_pred, colour = "red"))+
```

```
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_blank(), axis.text.x = element_text(angle = 45), legend.position = "none" )
  scale_x_discrete(breaks = everyother1) + labs (x = "Fecha", y = "Riesgo Relativo") +
  ggtitle(paste("Valores aproximados de training hasta el año:", Fechas_nom[i], sep = " "))+
  geom_ribbon(aes(ymin = u_overall_lower, ymax = u_overall_upper), alpha = 0.3)+
  ylim(-3, 10)


Eval.tn[i, 1:2] = as.numeric(metricas(df2))


#### VALORES AJUSTADOS TOTALES ####

MC_samples.tot = samples[[3]]

means <- NULL
means <- MC_samples.tot[, , 1:output_dim]
# average over the MC samples
predictive_mean <- apply(means, 2, mean)

epistemic_uncertainty <- apply(means, 2, var)


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


df3 <- data.frame(
  x = Fecha,
  y = Alajuela$RR,
  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)
)



p3[[i]] = ggplot(df3, aes(x = x, y = y, group = 1)) + geom_line(colour = "blue") +
  geom_line( aes(x = x, y = y_pred, colour = "red"))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_blank(), axis.text.x = element_text(angle = 45), legend.position = "none" )
  scale_x_discrete(breaks = everyother1) + labs (x = "Fecha", y = "Riesgo Relativo") +
  ggtitle(paste("Valores aproximados vs. RR observado, training antes del año:", Fechas_nom[i], sep = "
  geom_ribbon(aes(ymin = u_overall_lower, ymax = u_overall_upper), alpha = 0.3)+
```

```
  ylim(-3, 10)


Eval.tot[i, 1:2] = as.numeric(metricas(df3))

}
```

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

Resultados

```
Metricas = cbind (Eval.pd, Eval.tn, Eval.tot)
colnames(Metricas) = c("Test NMRSE", "Test NIS", "Train NMRSE", "Train NIS", "Total NMRSE", "Total NIS")
rownames(Metricas) = c("2021", "2020 +", "2019 +", "2018 +", "2017+", "2016+", "2015+", "2014+", "2013+
as.data.frame(Metricas)
```

```
##        Test NMRSE Test NIS Train NMRSE Train NIS Total NMRSE Total NIS
## 2021   0.09455641 4.233738   0.4025551  3.833167   0.3946659  3.837756
## 2020 + 0.73012480 5.391953   0.3165140  3.907108   0.3351875  3.979888
## 2019 + 0.43746535 4.367302   0.4148680  4.064297   0.4207408  4.093328
## 2018 + 0.30552904 3.991890   0.3735781  3.910866   0.3703808  3.929527
## 2017+  0.50416820 3.675390   0.4642659  3.744027   0.4690183  3.709109
## 2016+  1.09394454 6.597728   0.8897232  5.438602   0.9654608  5.868470
## 2015+  0.62003708 3.214939   0.4048946  4.116765   0.4980019  3.697439
## 2014+  0.70549982 3.627712   0.5499143  3.896548   0.6267063  3.821908
## 2013+  0.55478125 3.478053   0.4219397  4.450496   0.4868812  3.868262
## 2012+  0.50417198 3.261502   0.4513857  4.413017   0.4809989  3.743890
## 2011+  0.92300533 4.962695   0.6762020  4.206985   0.8334777  4.703411
```

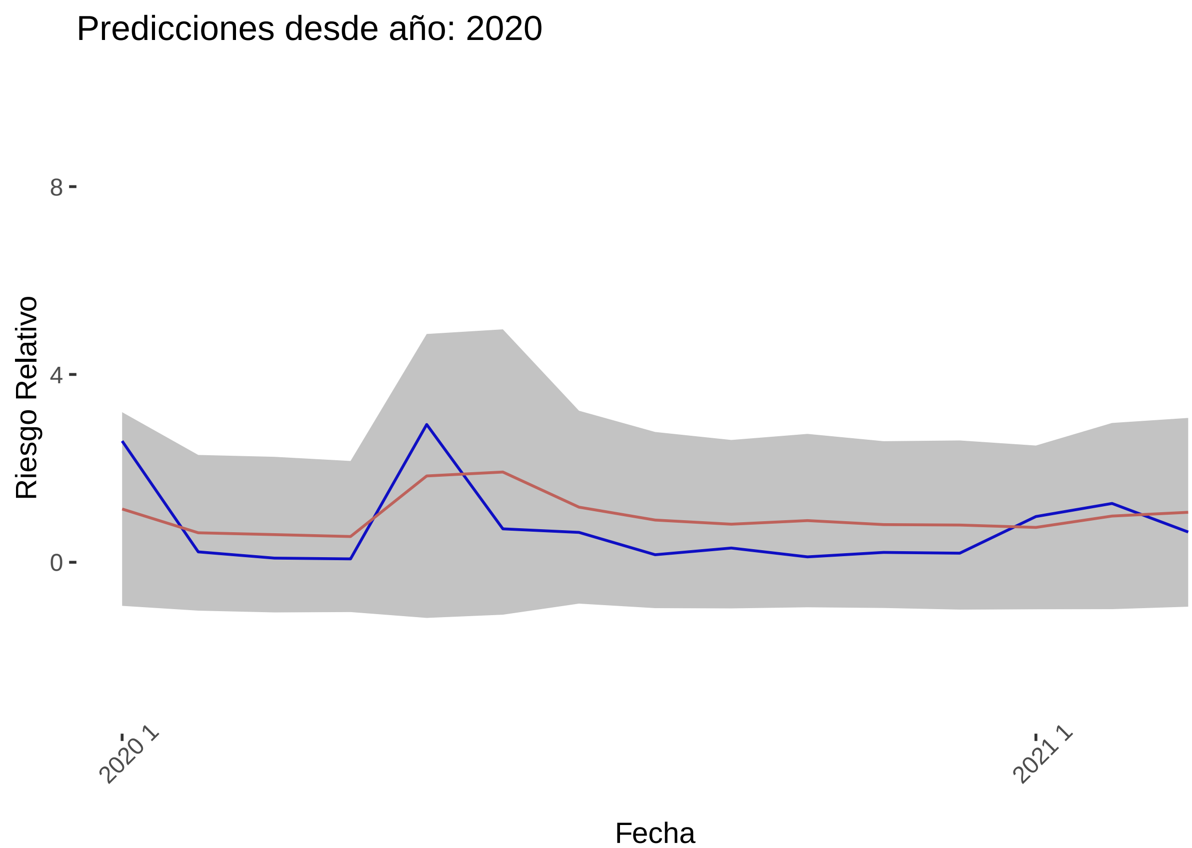
Gráficos

```
p1
```

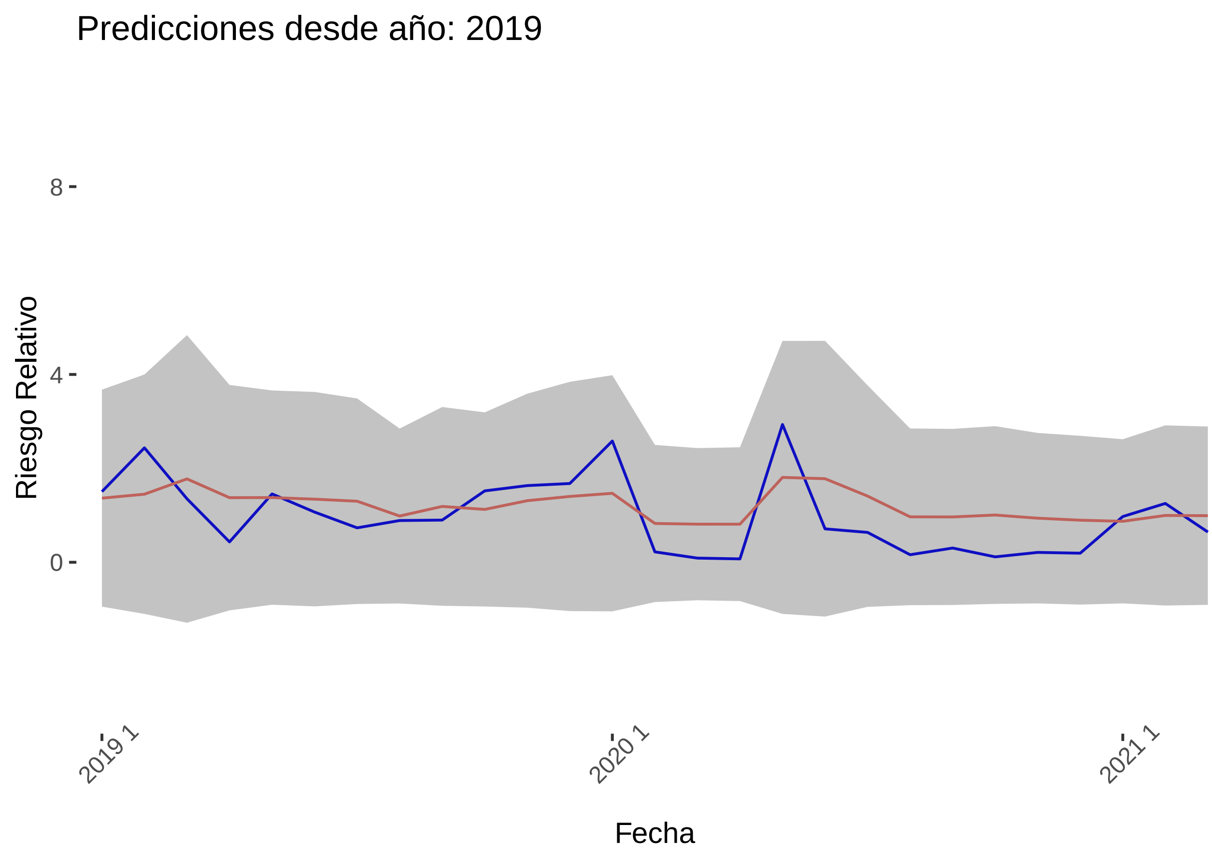
```
## [[1]]
```


Predicciones desde año: 2021
Riesgo Relativo
8
4
0
2021 1
Fecha

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

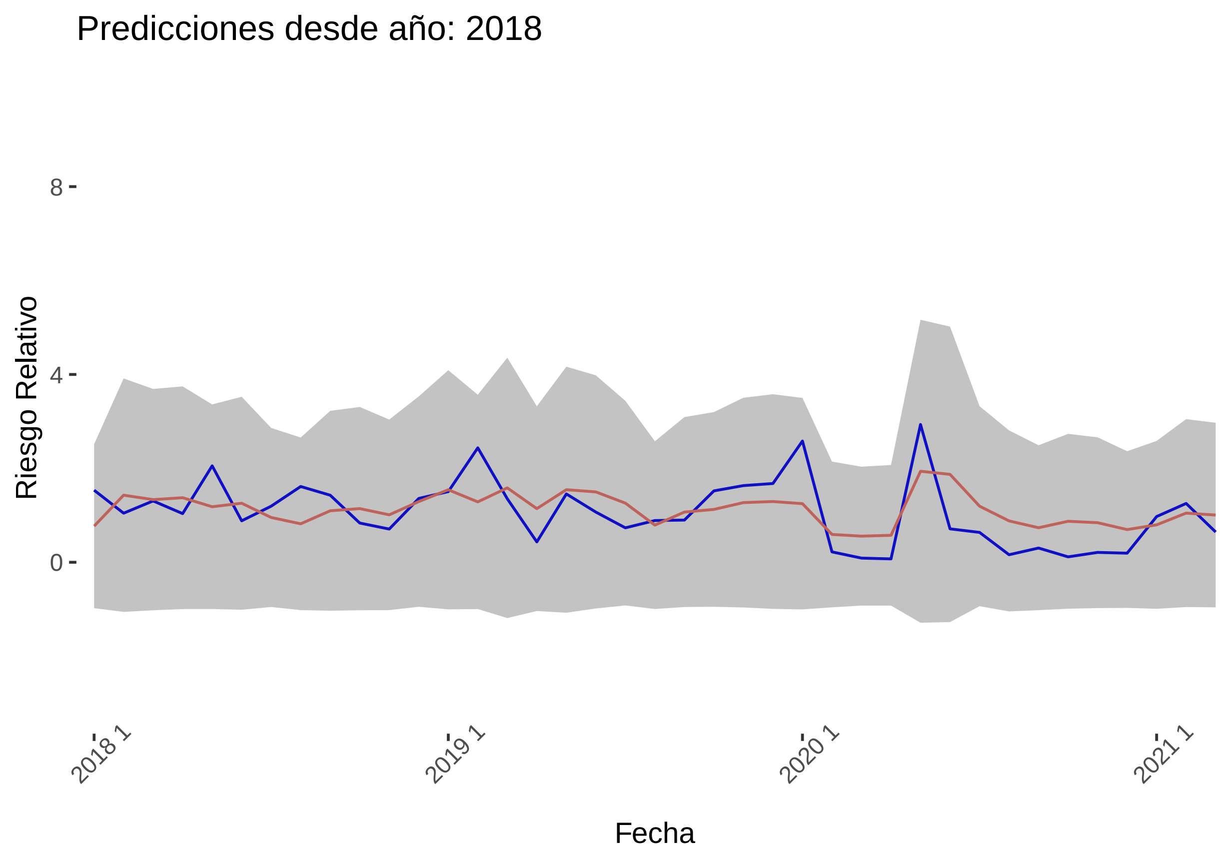

Predicciones desde año: 2020
Riesgo Relativo
8
4
0
2020 1
2021 1
Fecha

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



Predicciones desde año: 2019
8
4
0
Riesgo Relativo
2019 1
2020 1
2021 1
Fecha

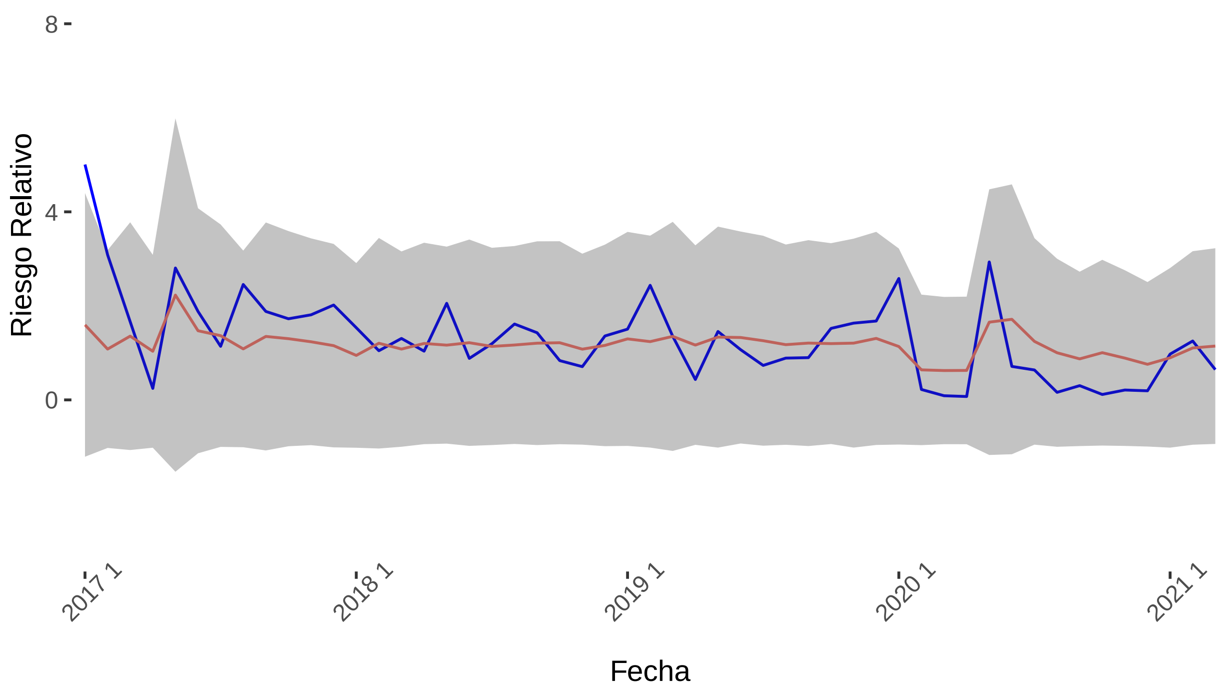

```
##
## [[4]]
```



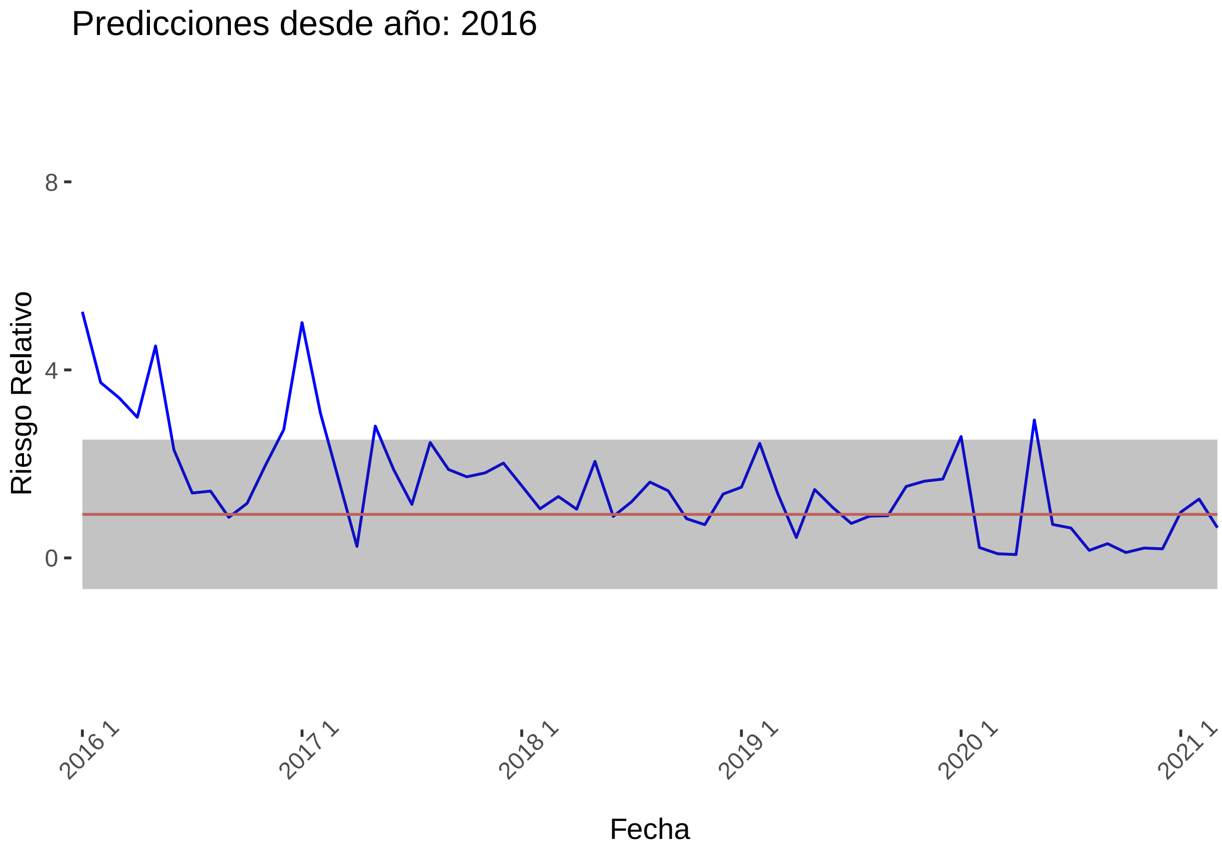
Predicciones desde año: 2018
8
4
0
Riesgo Relativo
2018 1
2019 1
2020 1
2021 1
Fecha


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

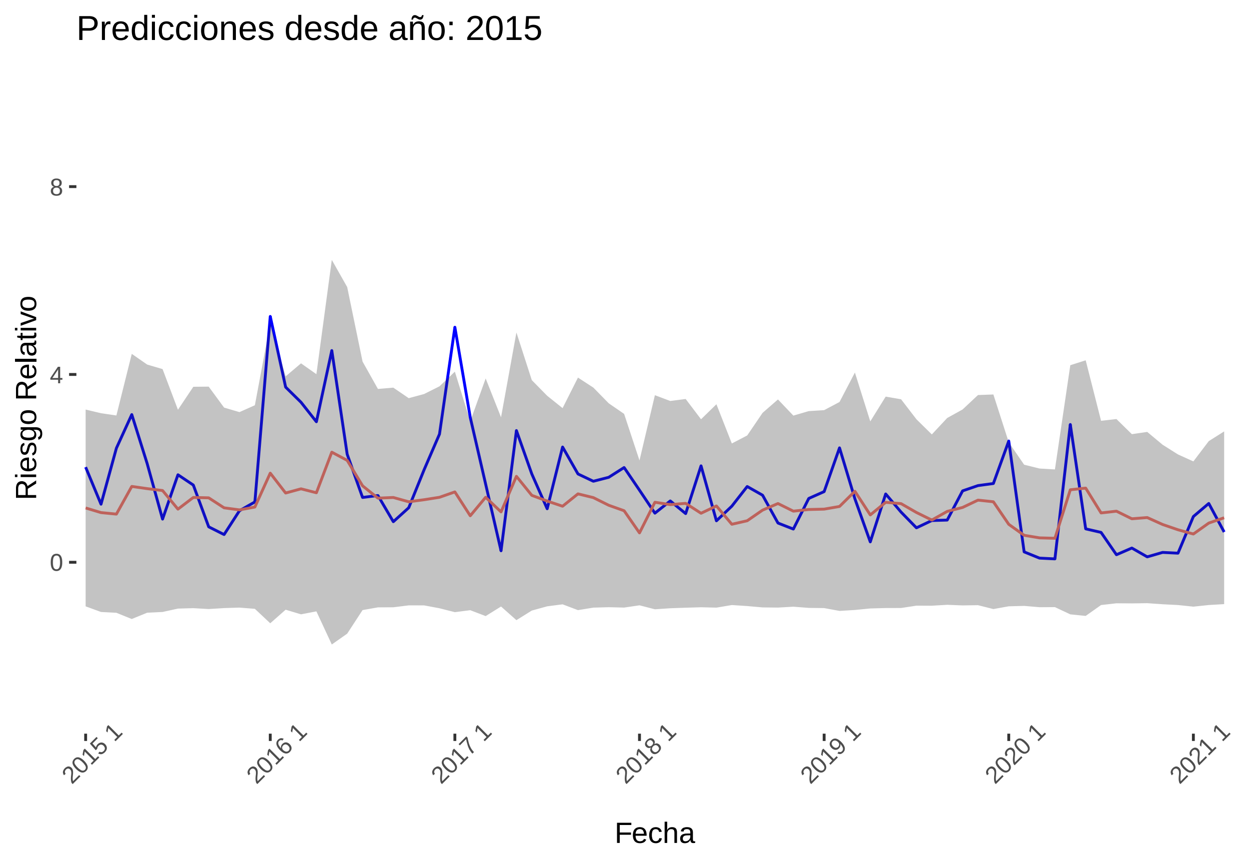
Predicciones desde año: 2017




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

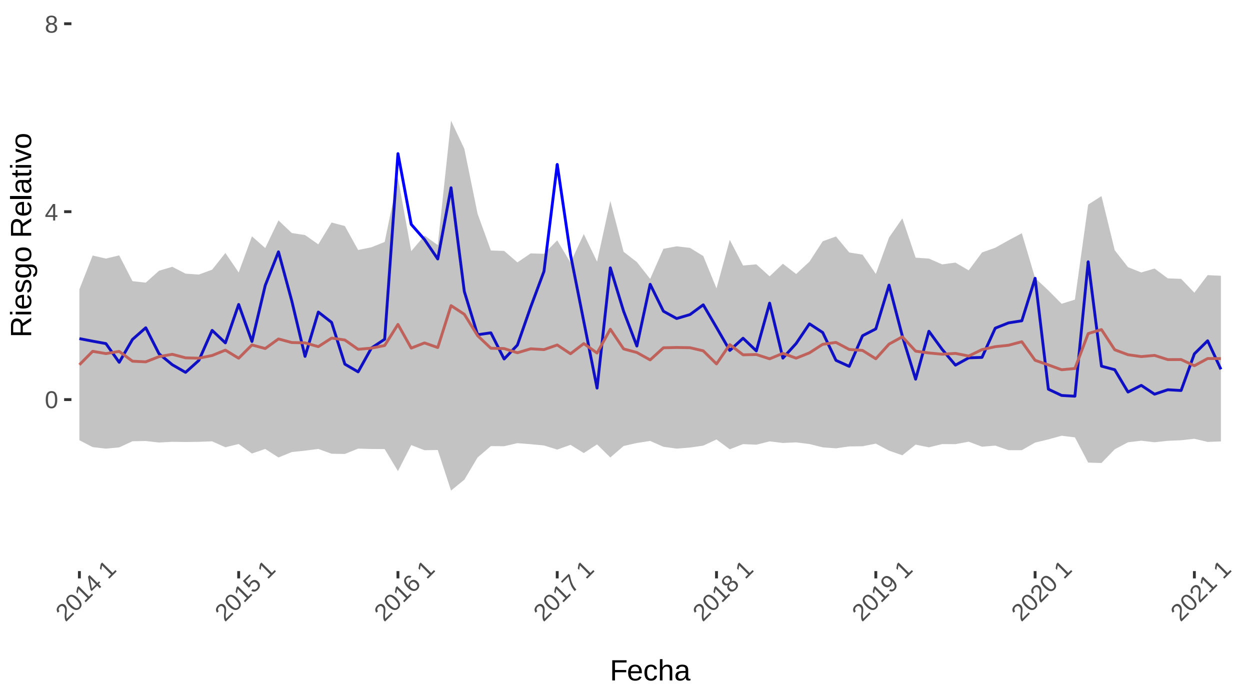

Predicciones desde año: 2016
Riesgo Relativo
8
4
0
2016 1
2017 1
2018 1
2019 1
2020 1
2021 1
Fecha

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

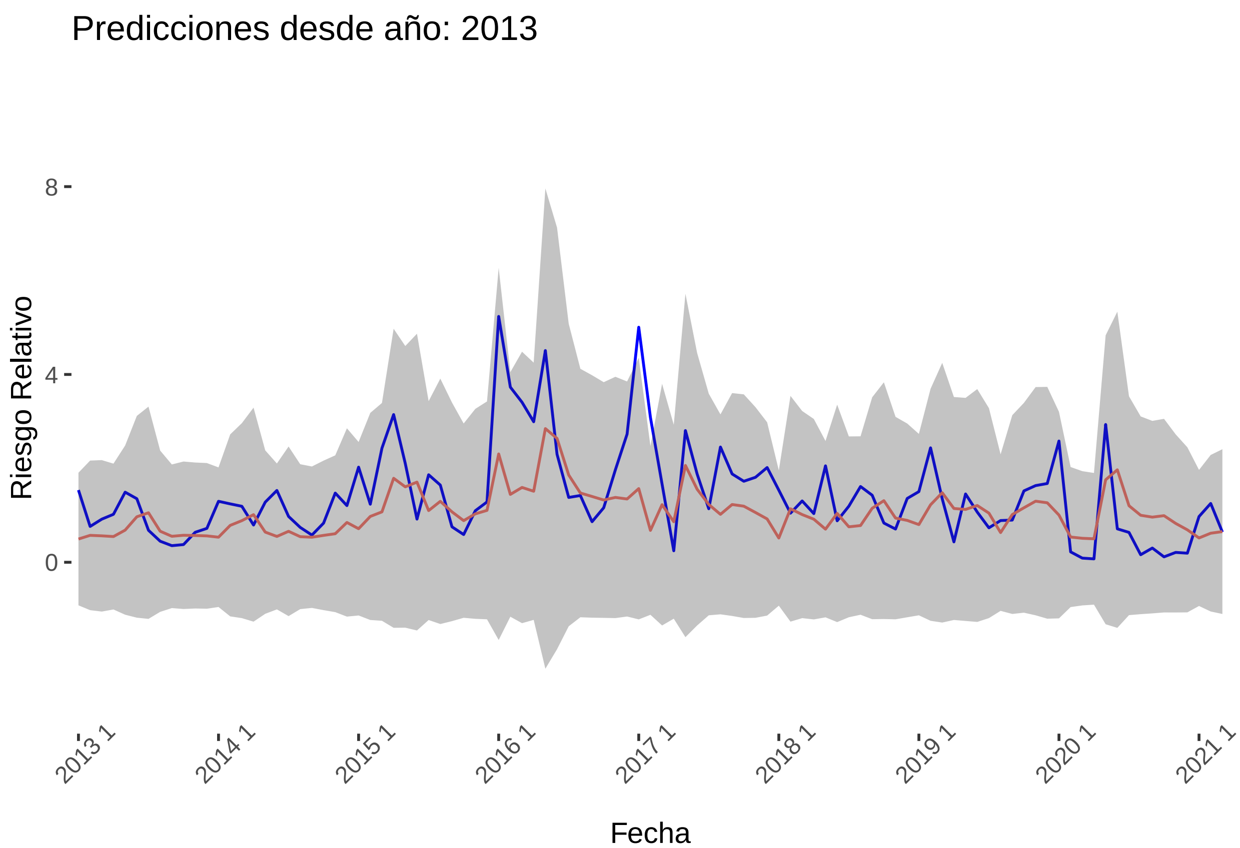

Predicciones desde año: 2015
Riesgo Relativo
8
4
0
2015 1
2016 1
2017 1
2018 1
2019 1
2020 1
2021 1
Fecha

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

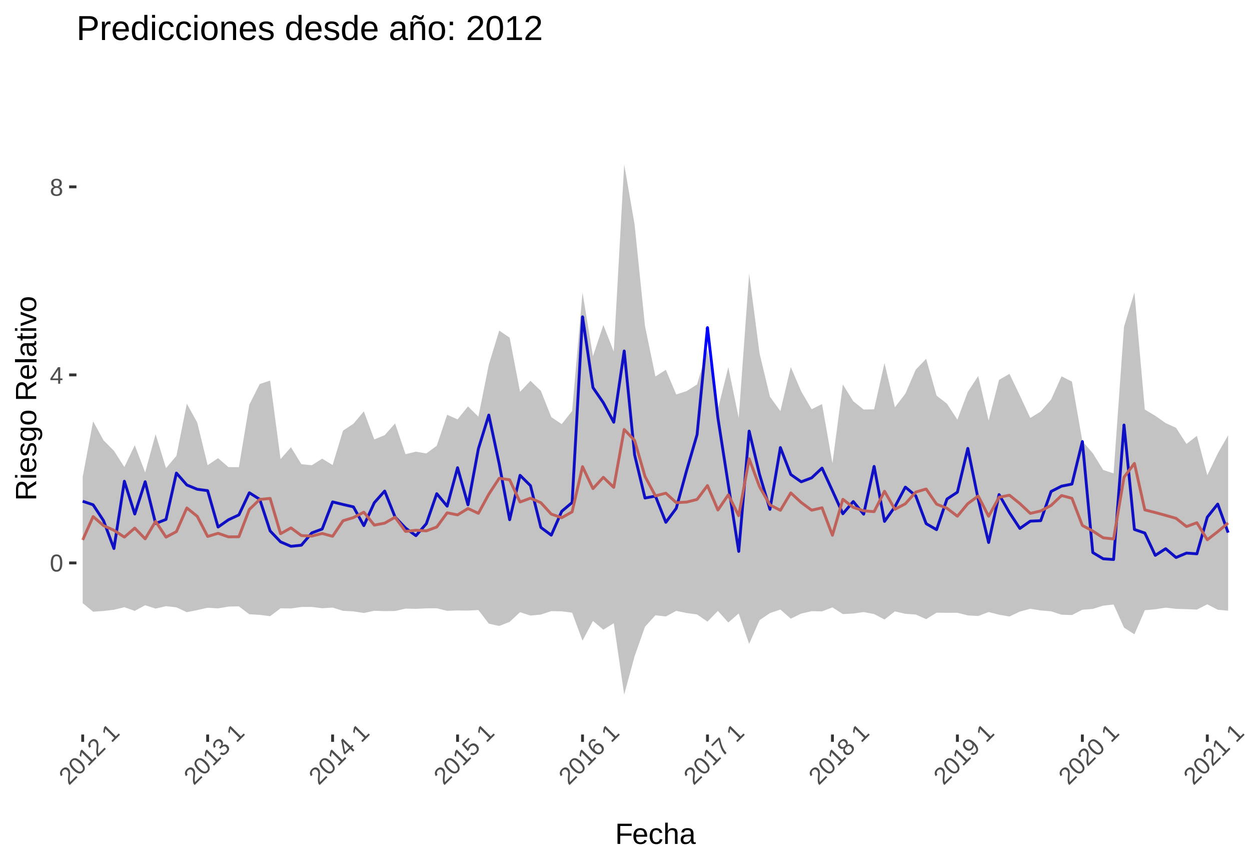
Predicciones desde año: 2014




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


Predicciones desde año: 2013
8
4
0
Riesgo Relativo
2013 1
2014 1
2015 1
2016 1
2017 1
2018 1
2019 1
2020 1
2021 1
Fecha

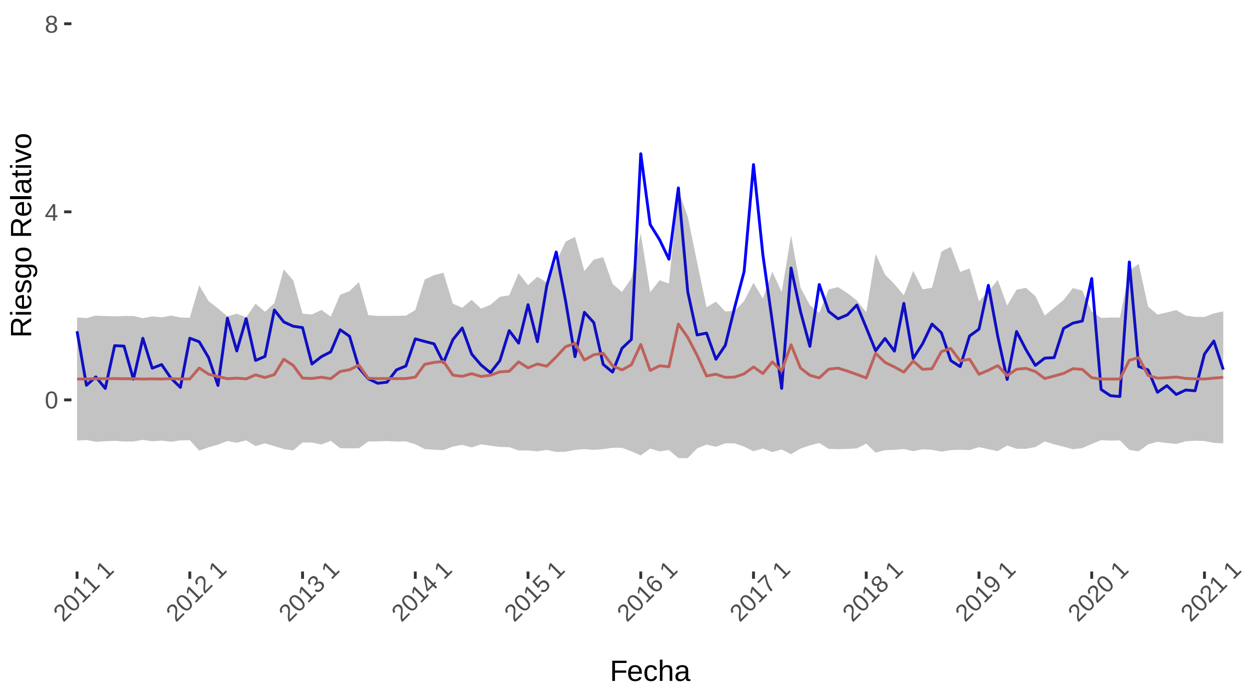
```
##
## [[10]]
```



Predicciones desde año: 2012
Riesgo Relativo
8
4
0
2012 1
2013 1
2014 1
2015 1
2016 1
2017 1
2018 1
2019 1
2020 1
2021 1
Fecha


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

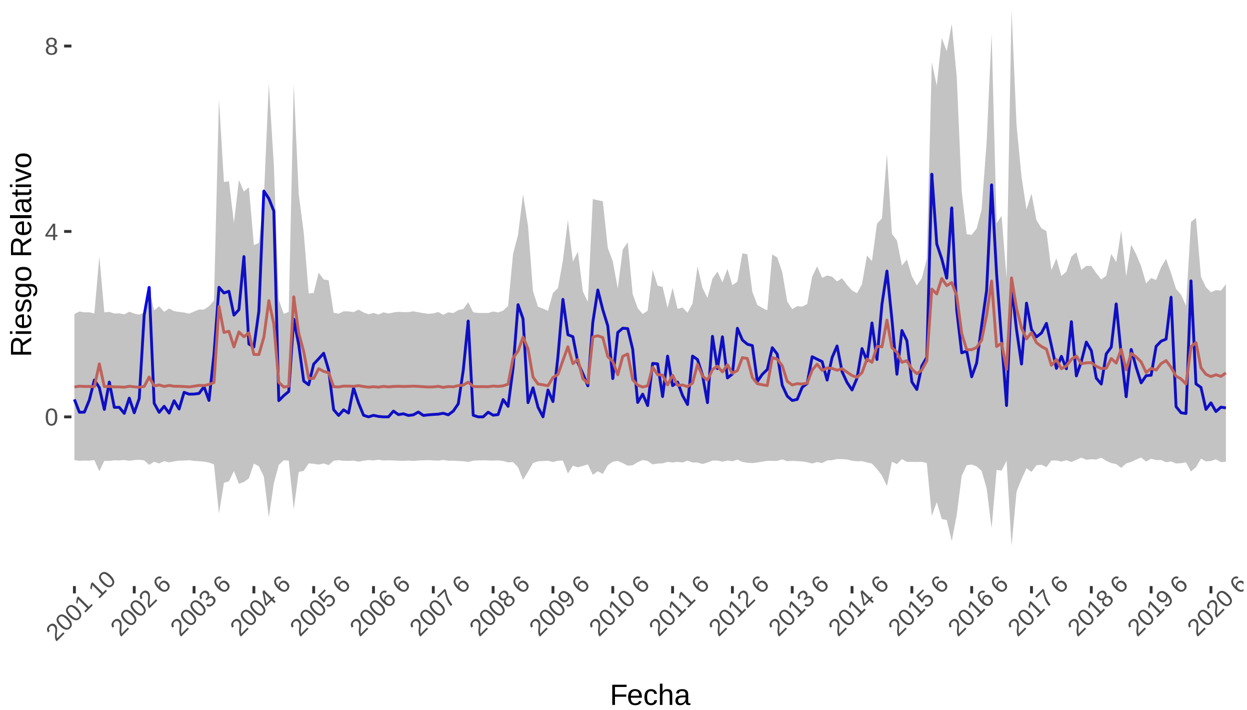
Predicciones desde año: 2011




```
p2
```

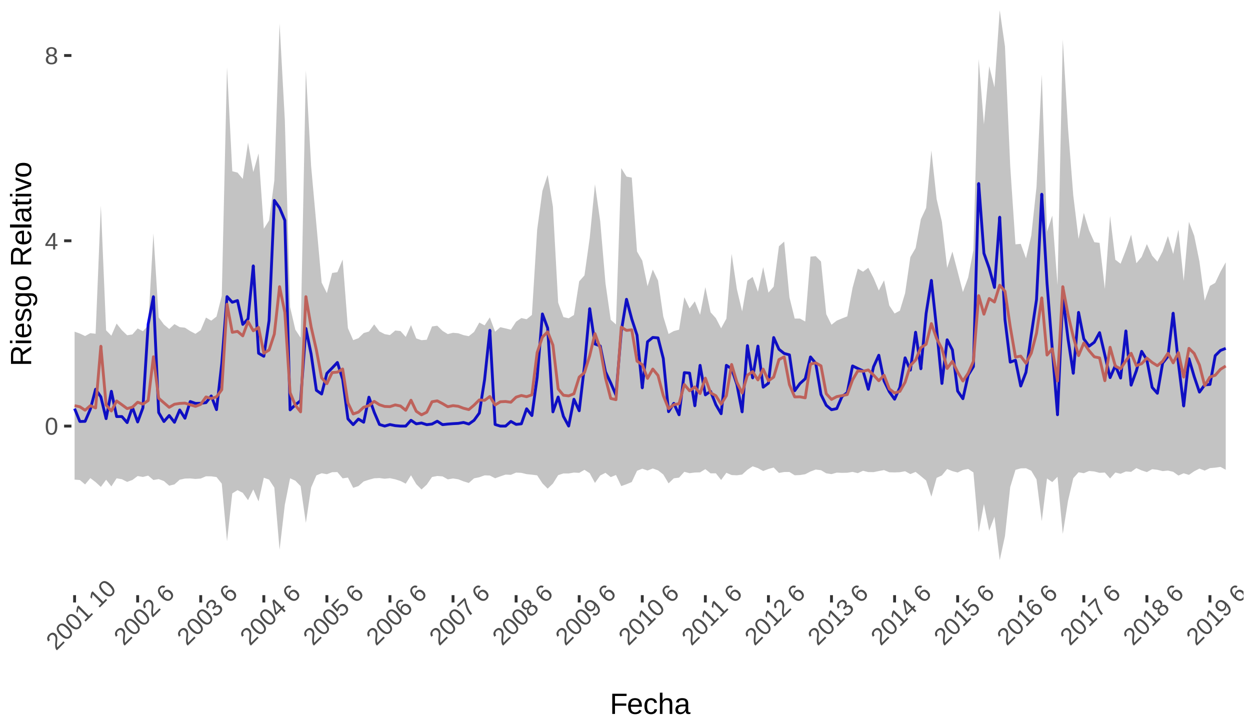
```
## [[1]]
```

Valores aproximados de training hasta el año: 2021

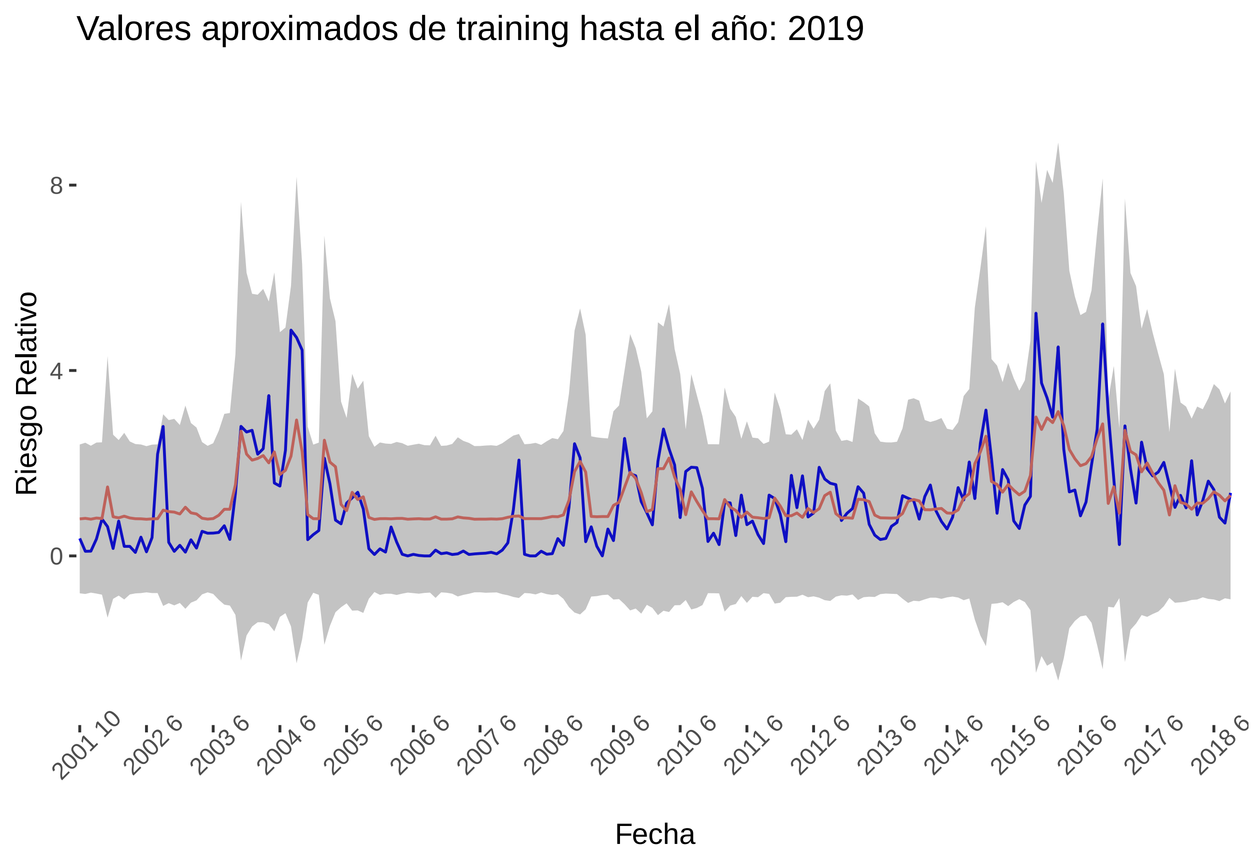



```
##
## [[2]]
```

Valores aproximados de training hasta el año: 2020

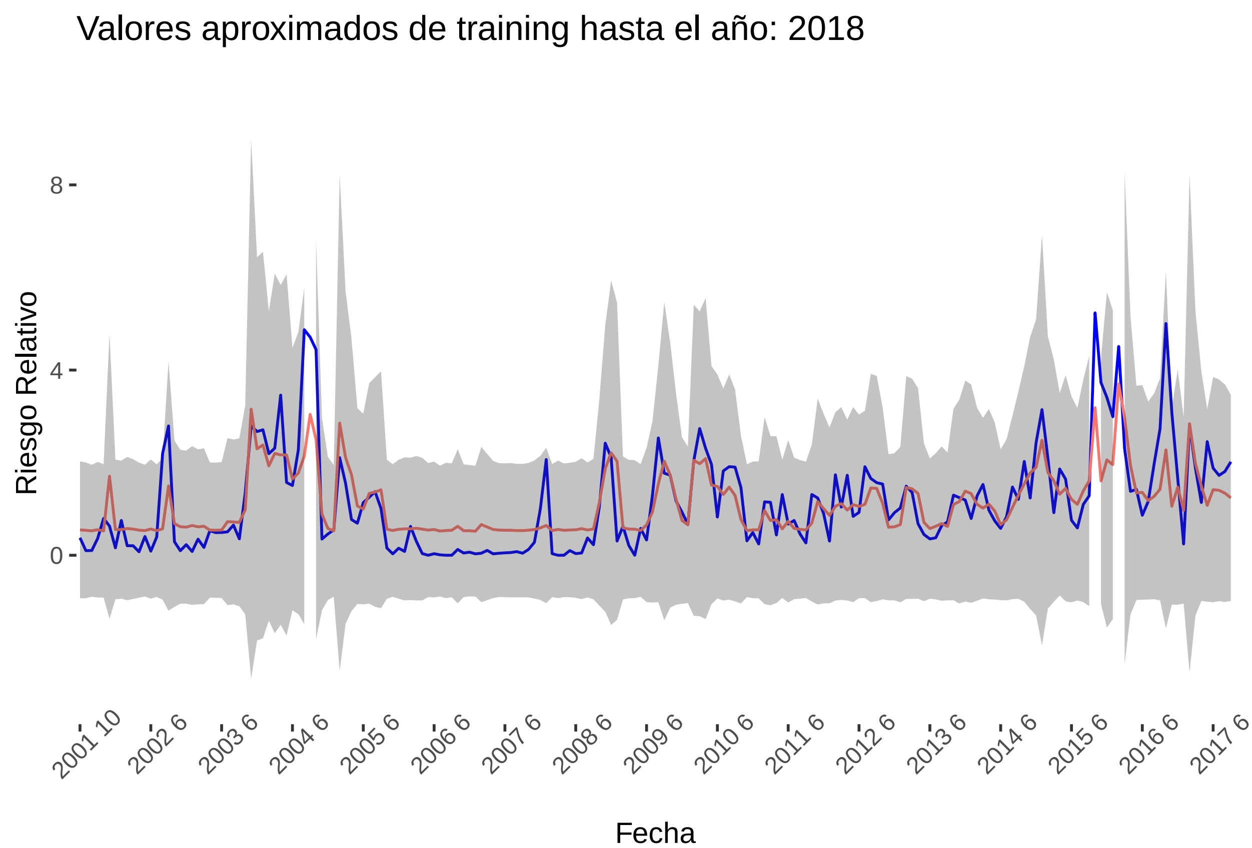



```
##
## [[3]]
```



Valores aproximados de training hasta el año: 2019
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
Fecha


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


Valores aproximados de training hasta el año: 2018
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
Fecha

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

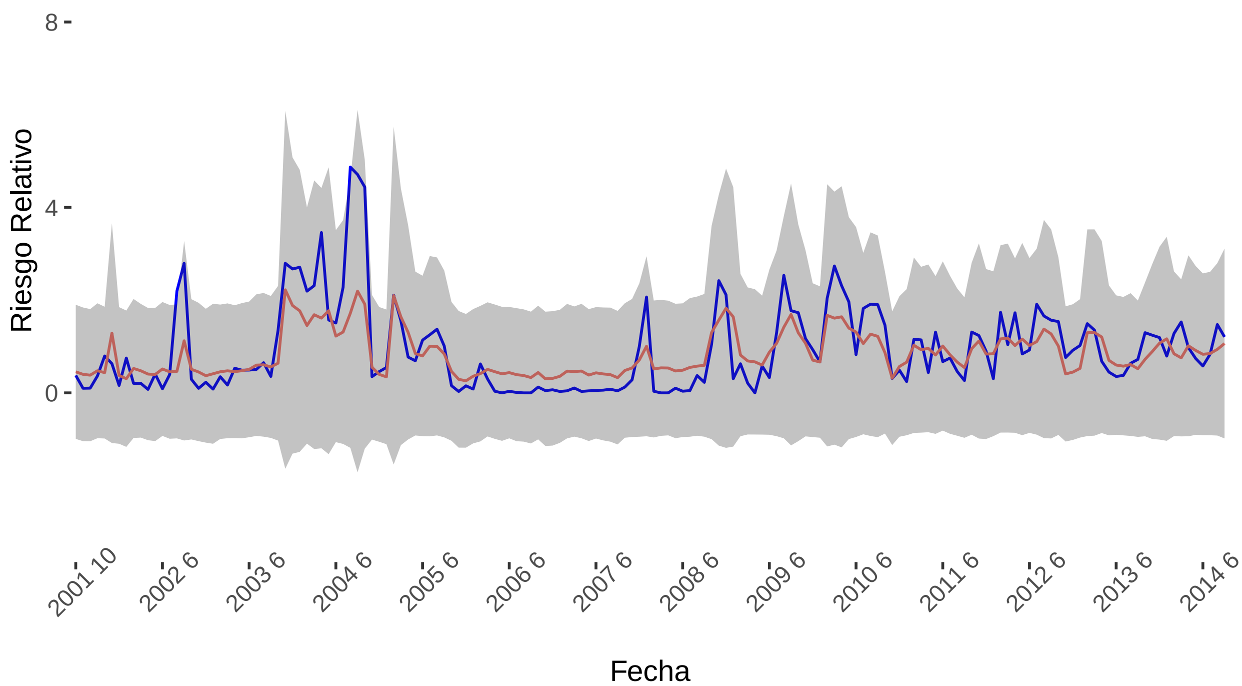
Valores aproximados de training hasta el año: 2017
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
Fecha


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

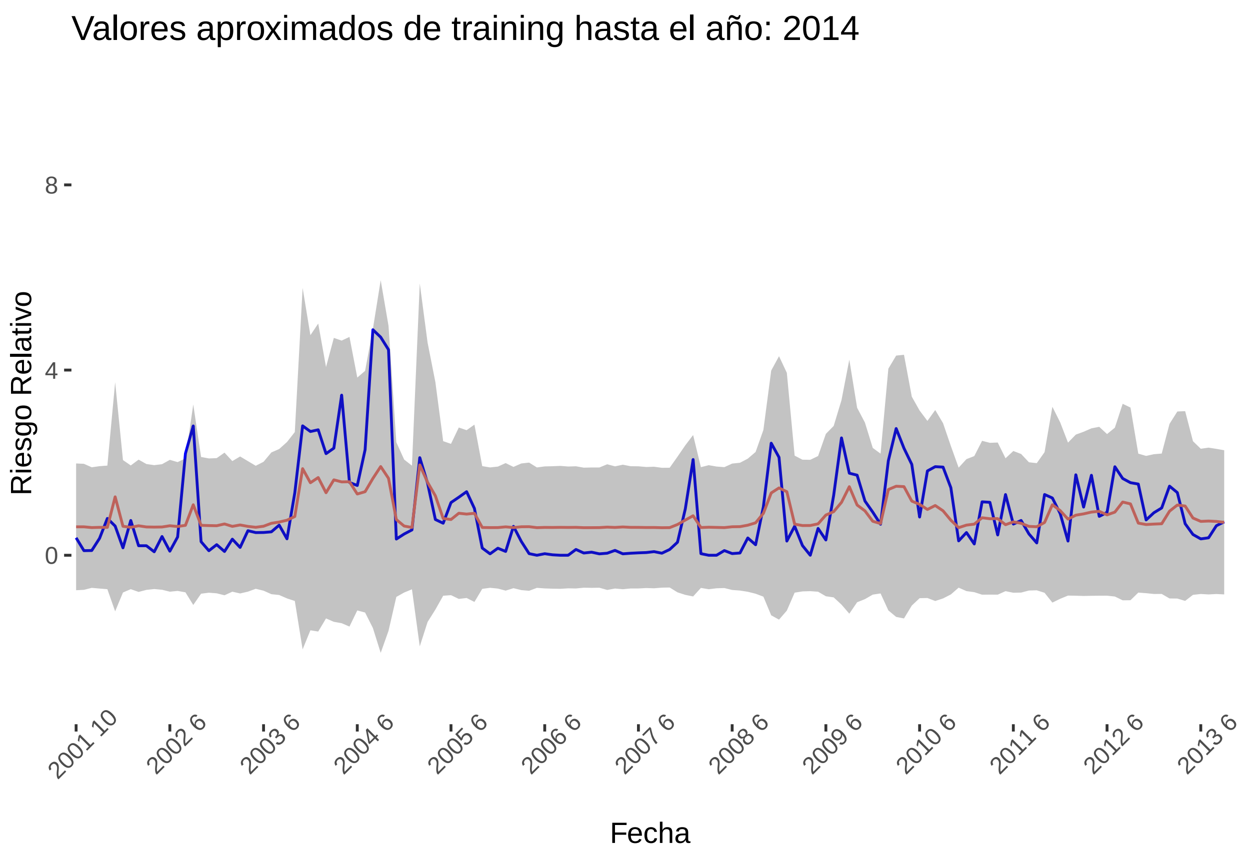

Valores aproximados de training hasta el año: 2016
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
Fecha

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

Valores aproximados de training hasta el año: 2015



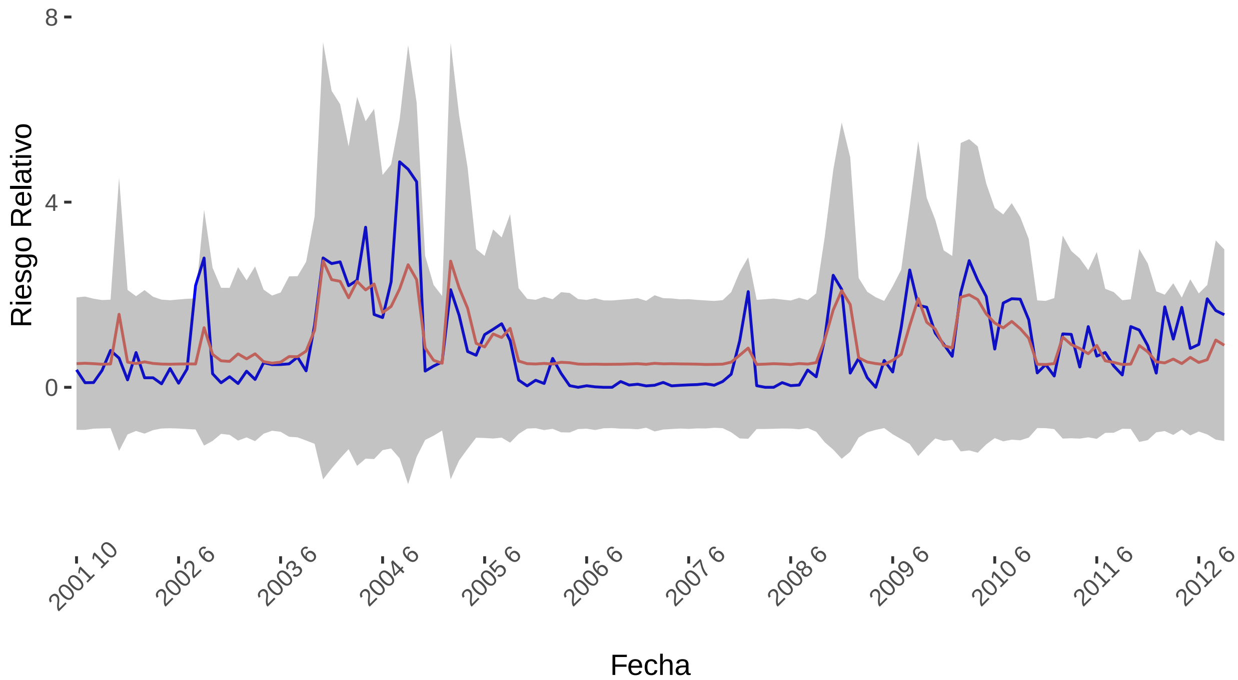

```
##
## [[8]]
```



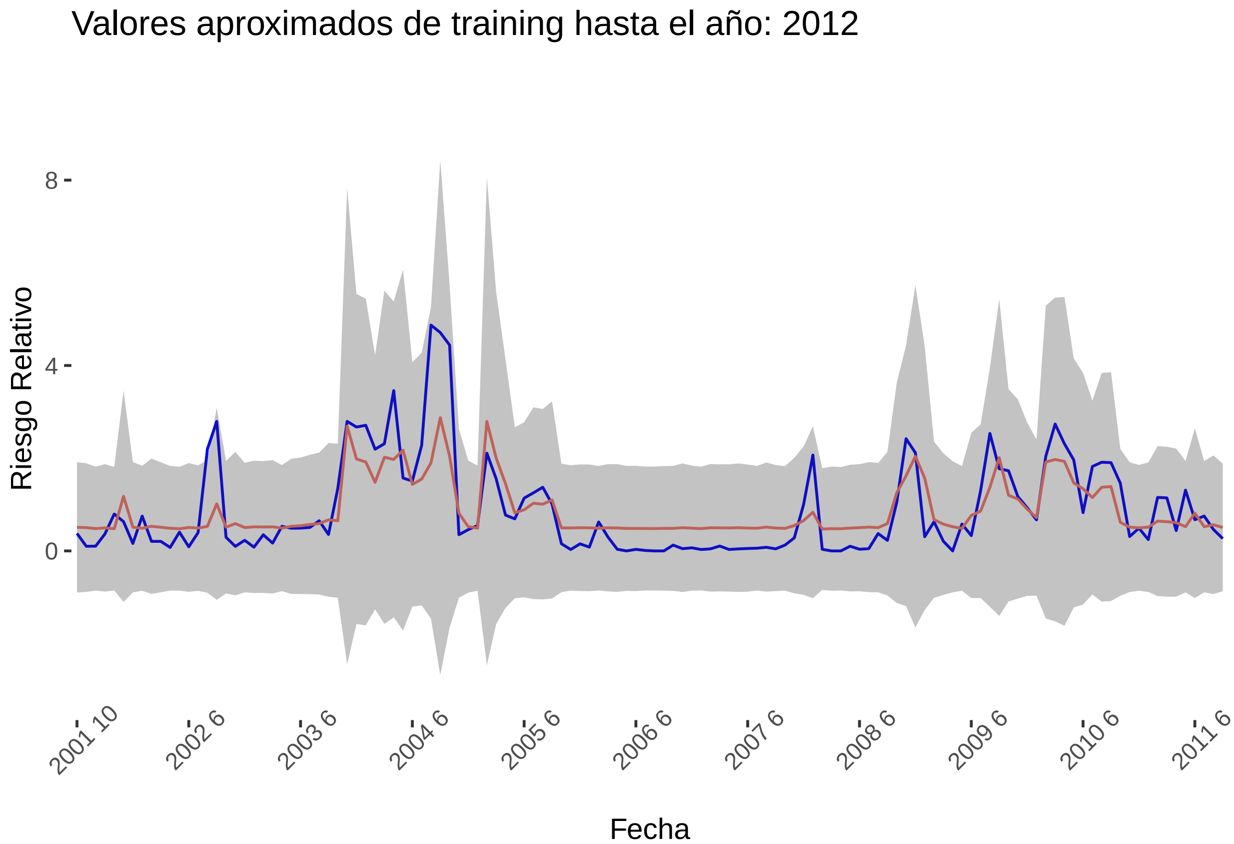
Valores aproximados de training hasta el año: 2014
8
4
0
Riesgo Relativo
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
Fecha


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

Valores aproximados de training hasta el año: 2013




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



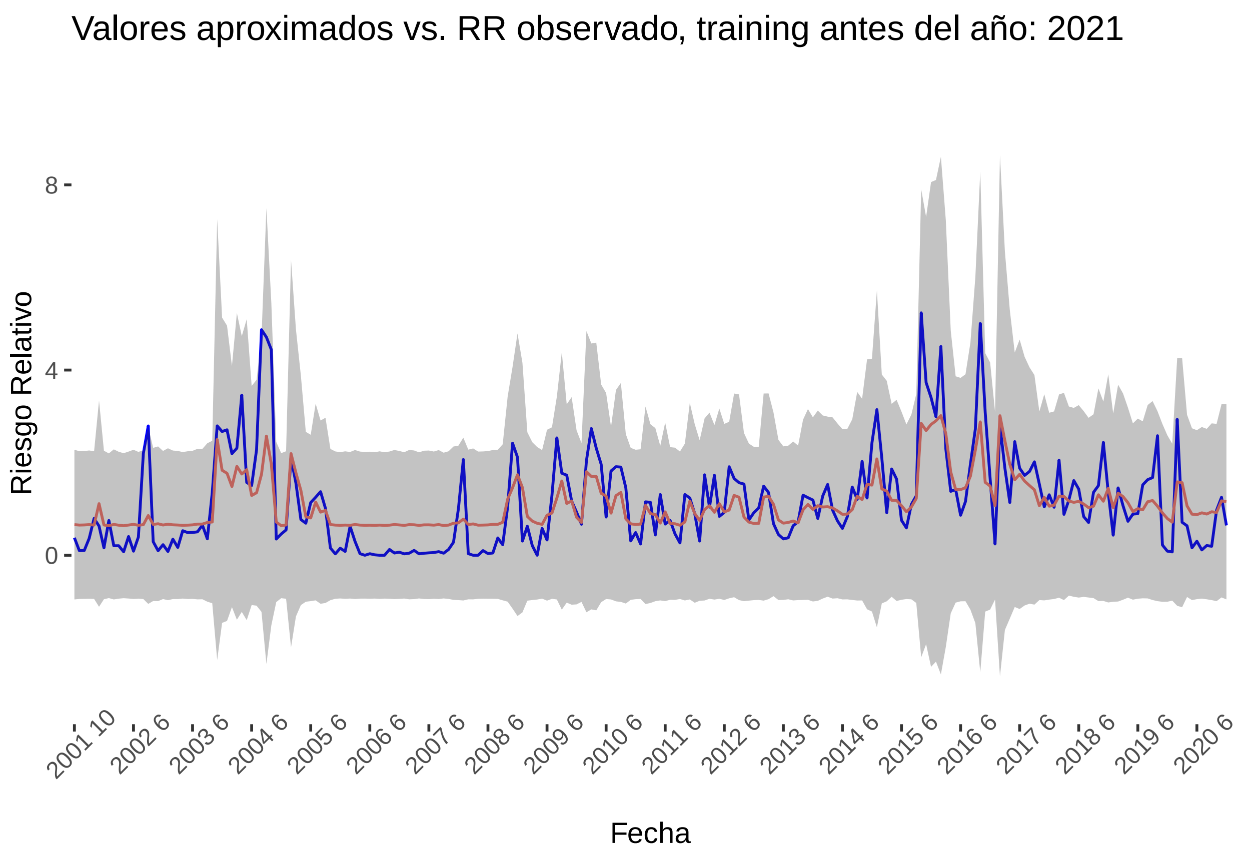
Valores aproximados de training hasta el año: 2012
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
Fecha


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

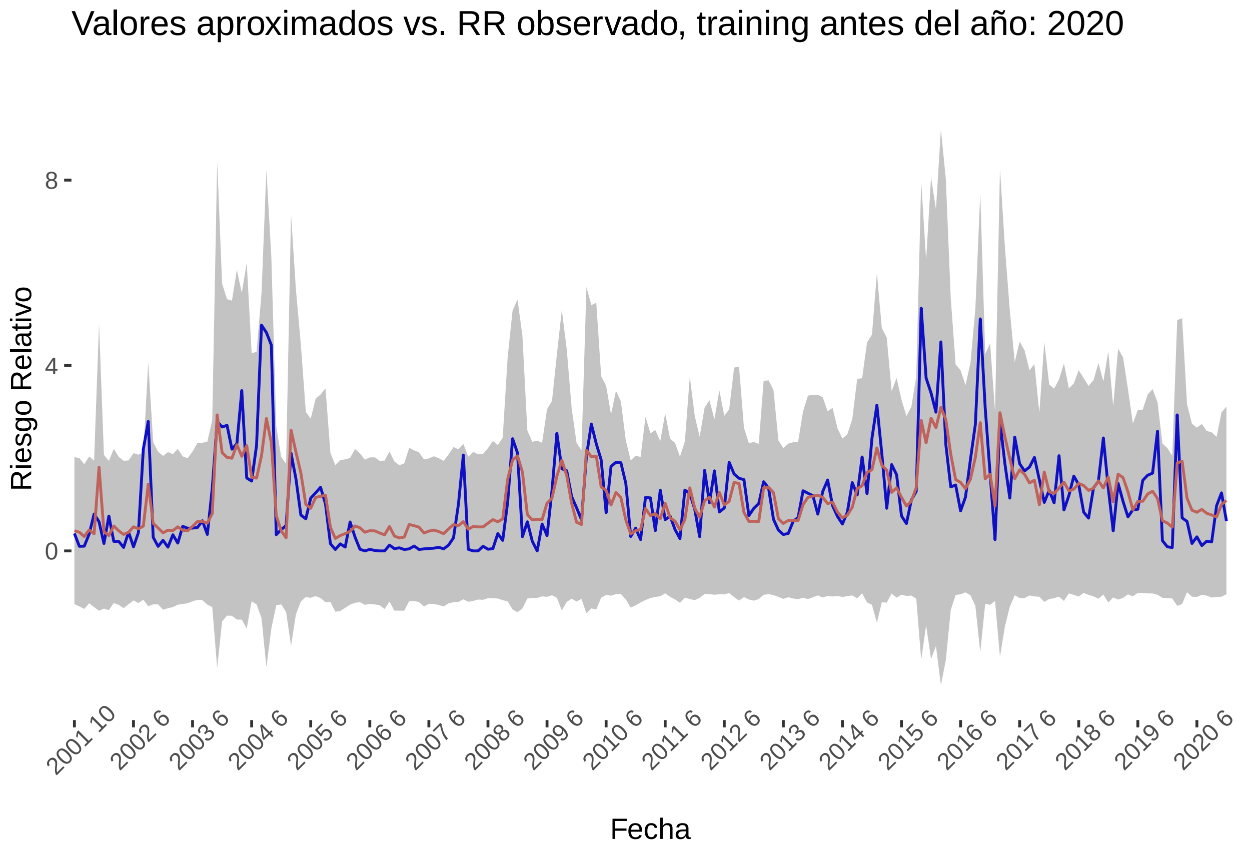
Valores aproximados de training hasta el año: 2011
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
Fecha


```
p3
```

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

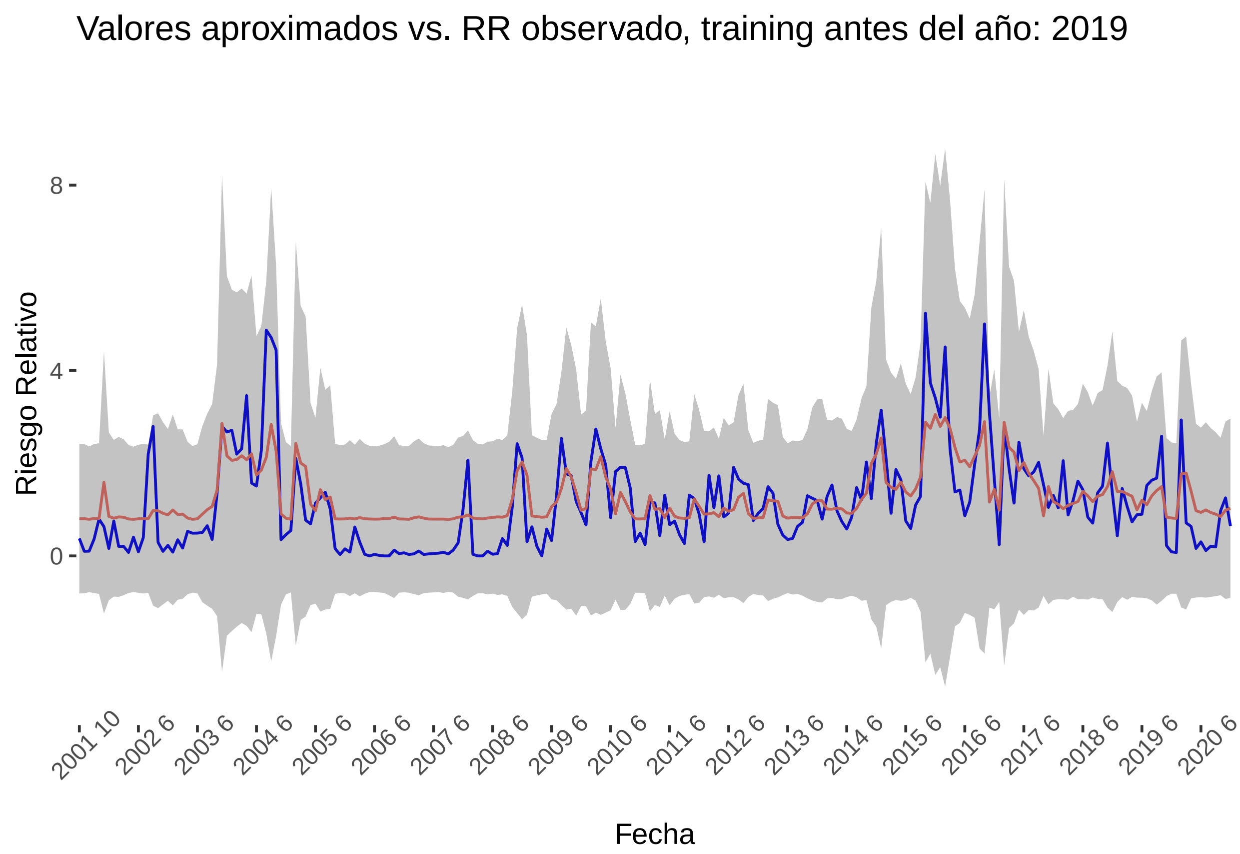

Valores aproximados vs. RR observado, training antes del año: 2021
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha

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



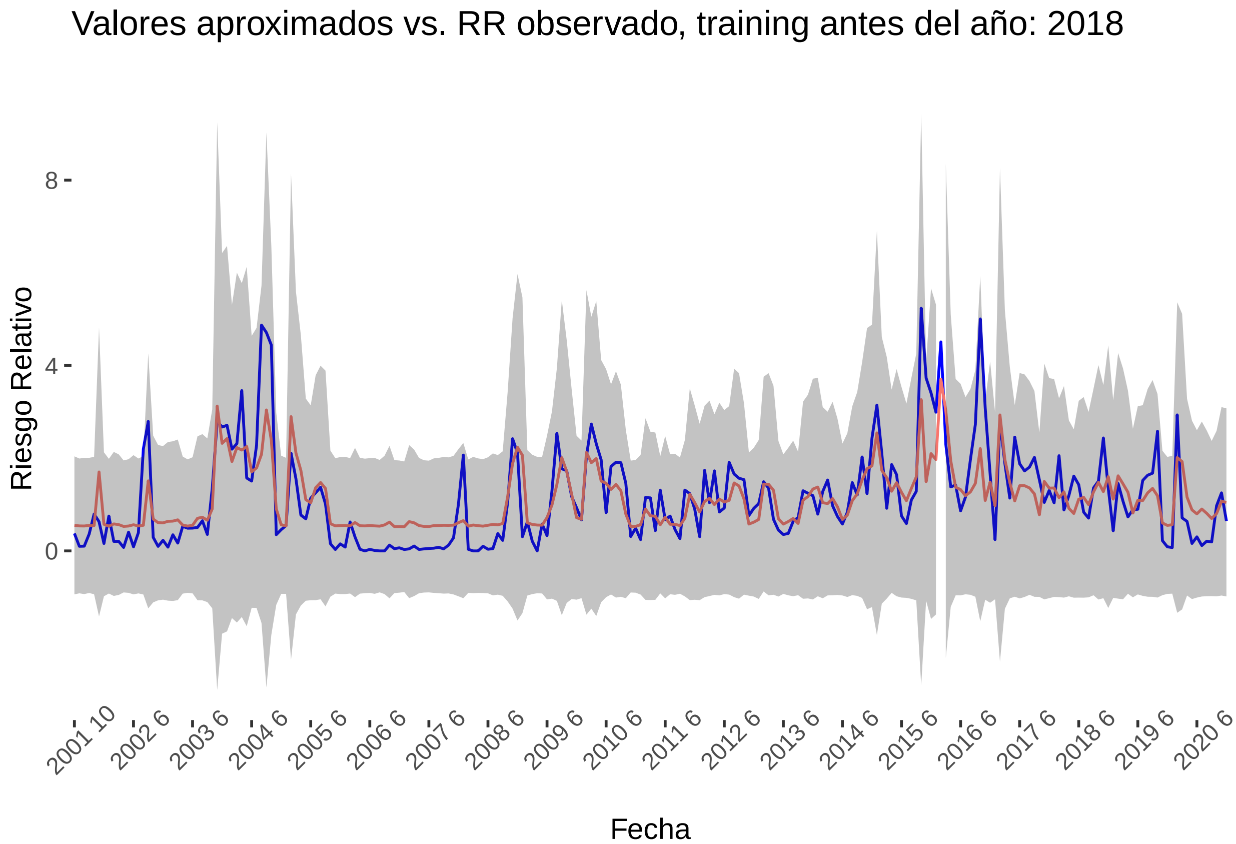
Valores aproximados vs. RR observado, training antes del año: 2020
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha


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



Valores aproximados vs. RR observado, training antes del año: 2019
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha

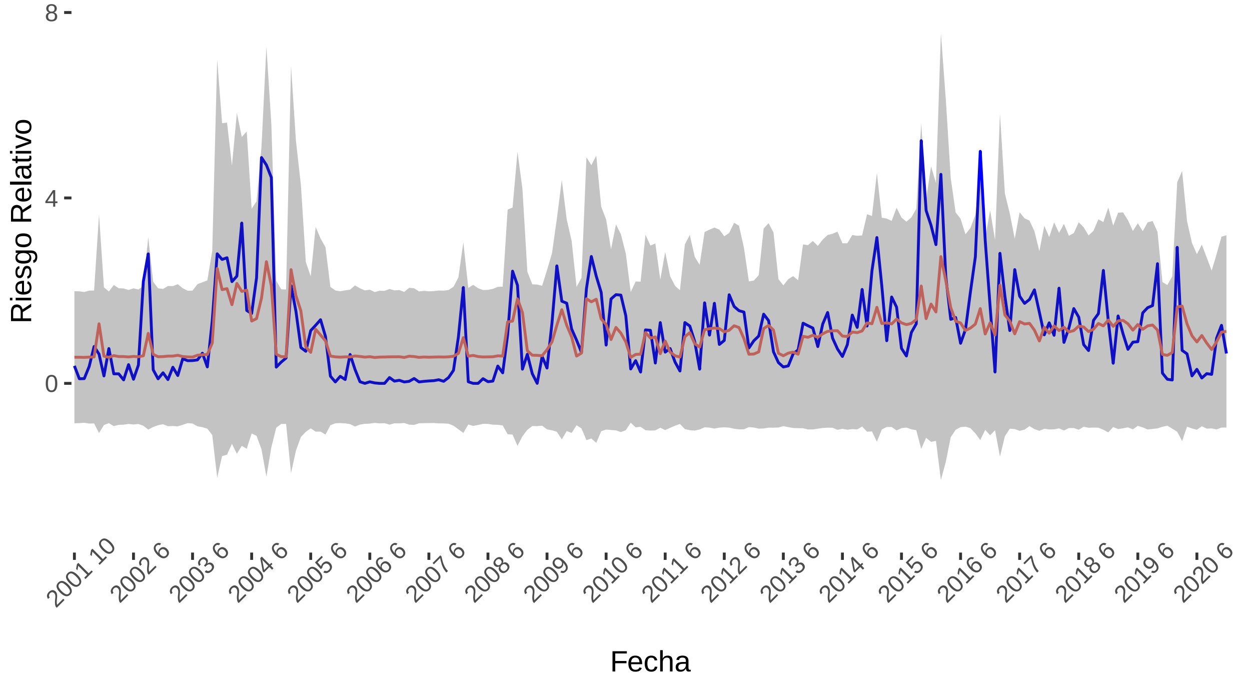

```
##
## [[4]]
```



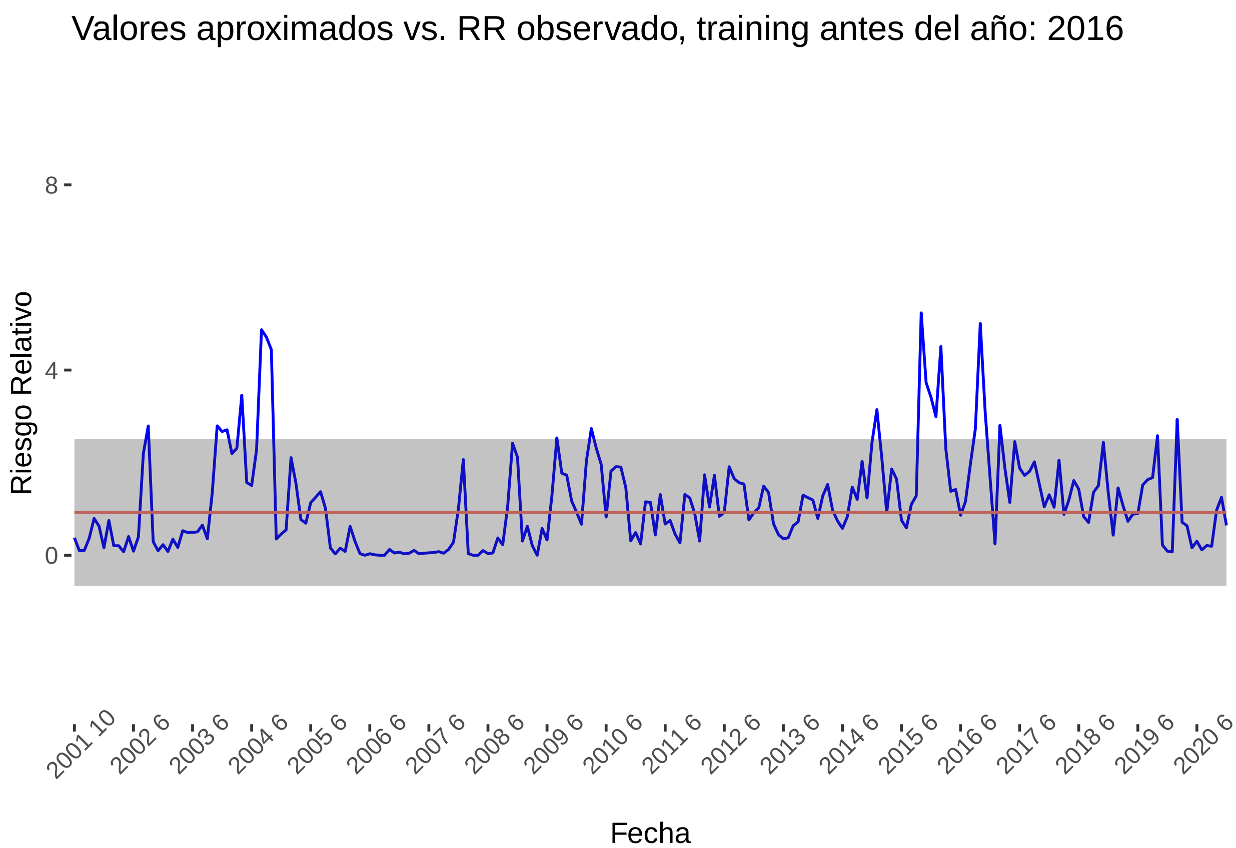
Valores aproximados vs. RR observado, training antes del año: 2018
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha


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

Valores aproximados vs. RR observado, training antes del año: 2017

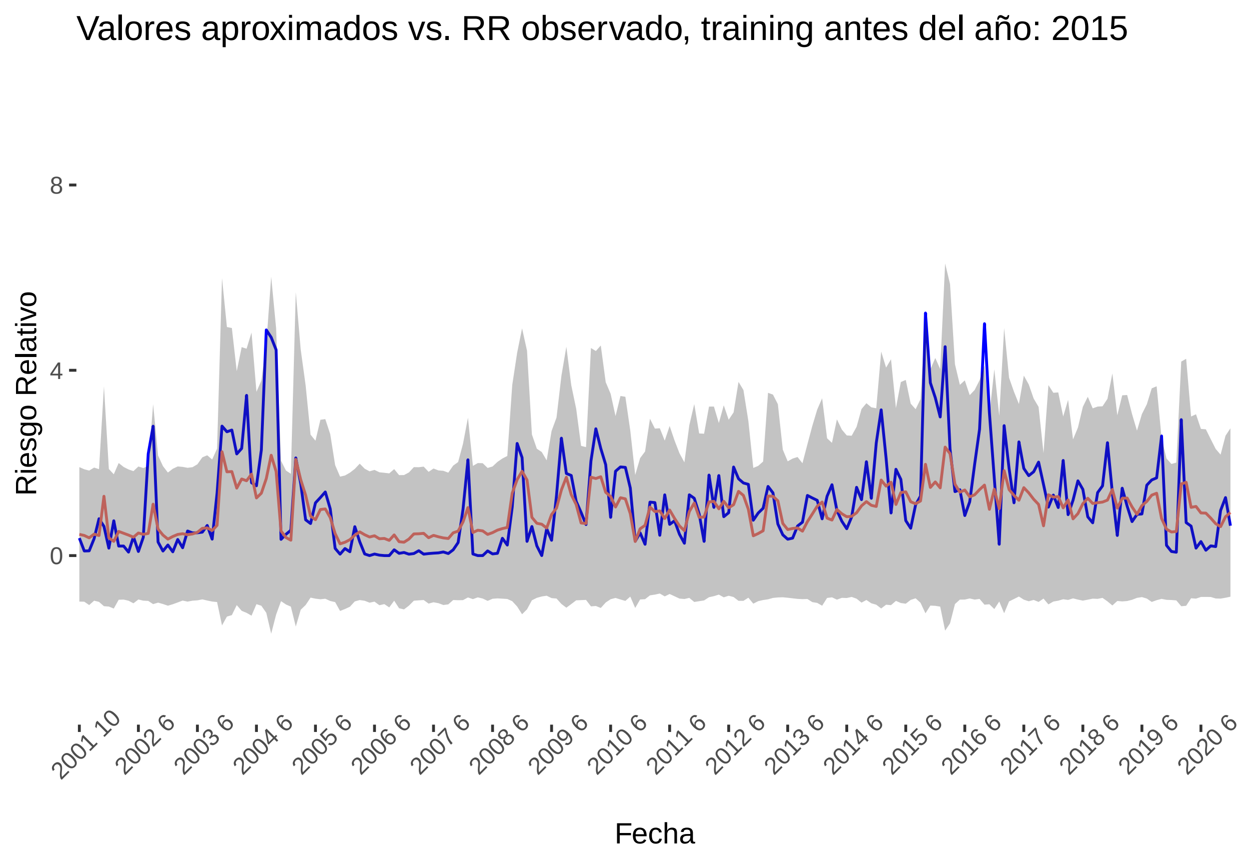



```
##
## [[6]]
```



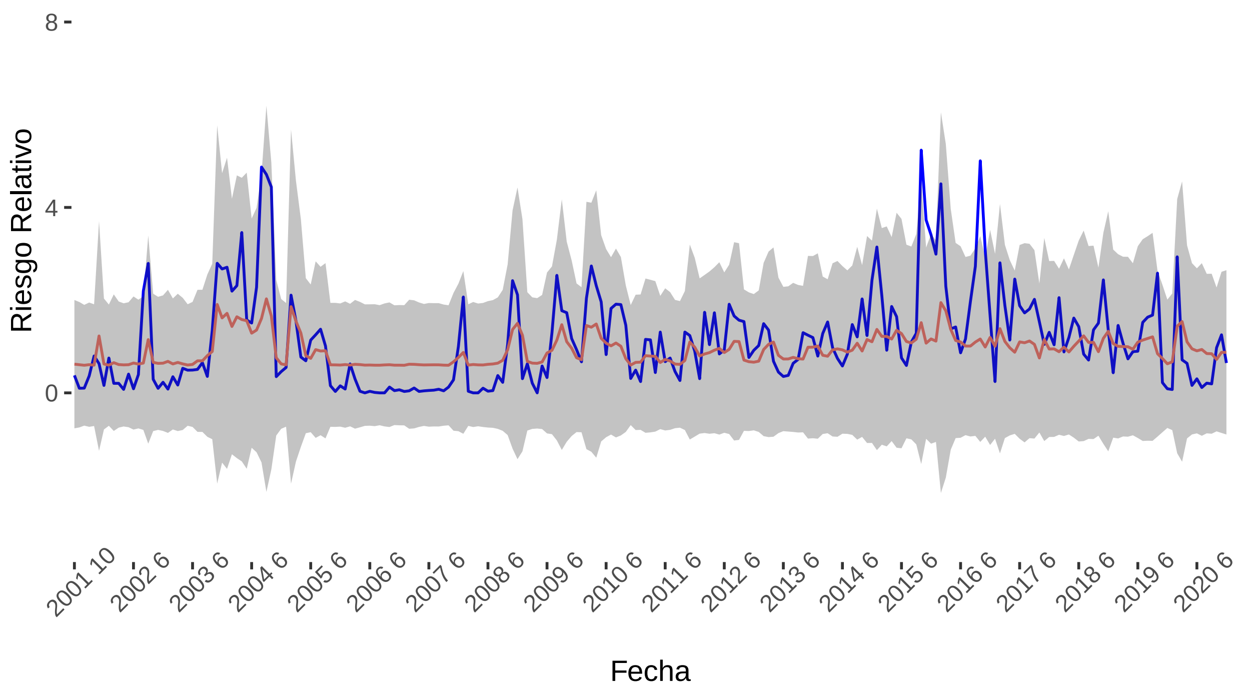
Valores aproximados vs. RR observado, training antes del año: 2016
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha


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


Valores aproximados vs. RR observado, training antes del año: 2015
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha

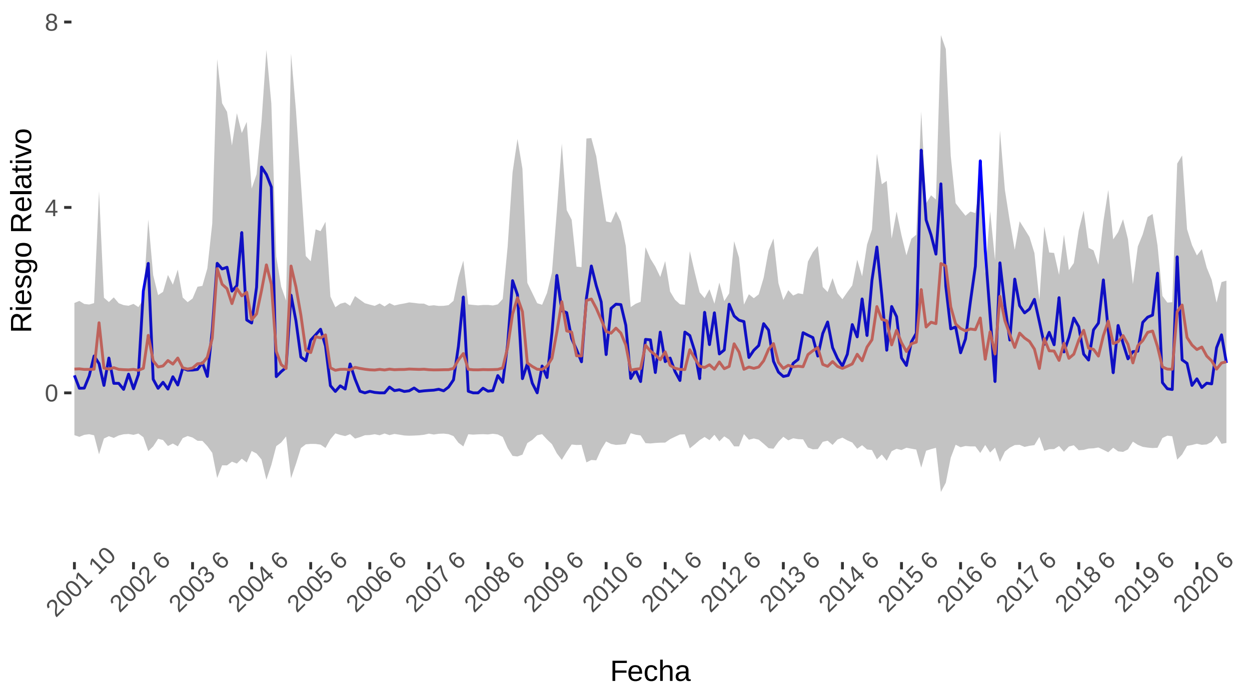
```
##
## [[8]]
```

Valores aproximados vs. RR observado, training antes del año: 2014



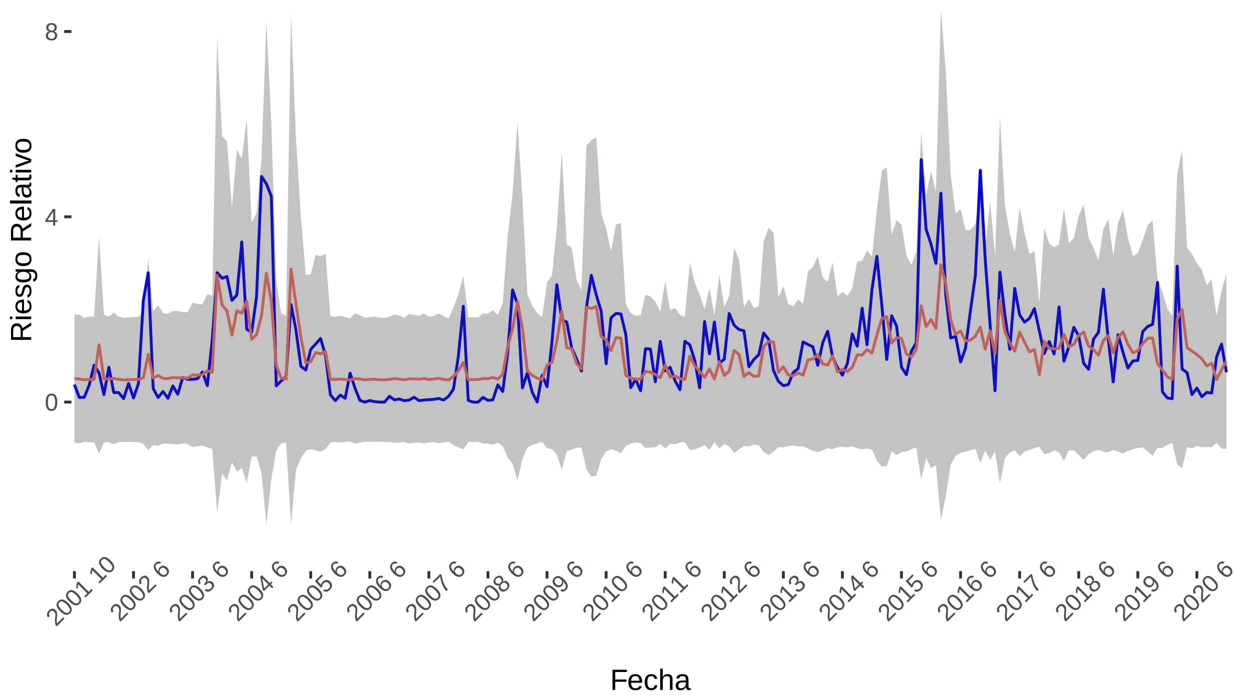

```
##
## [[9]]
```

Valores aproximados vs. RR observado, training antes del año: 2013

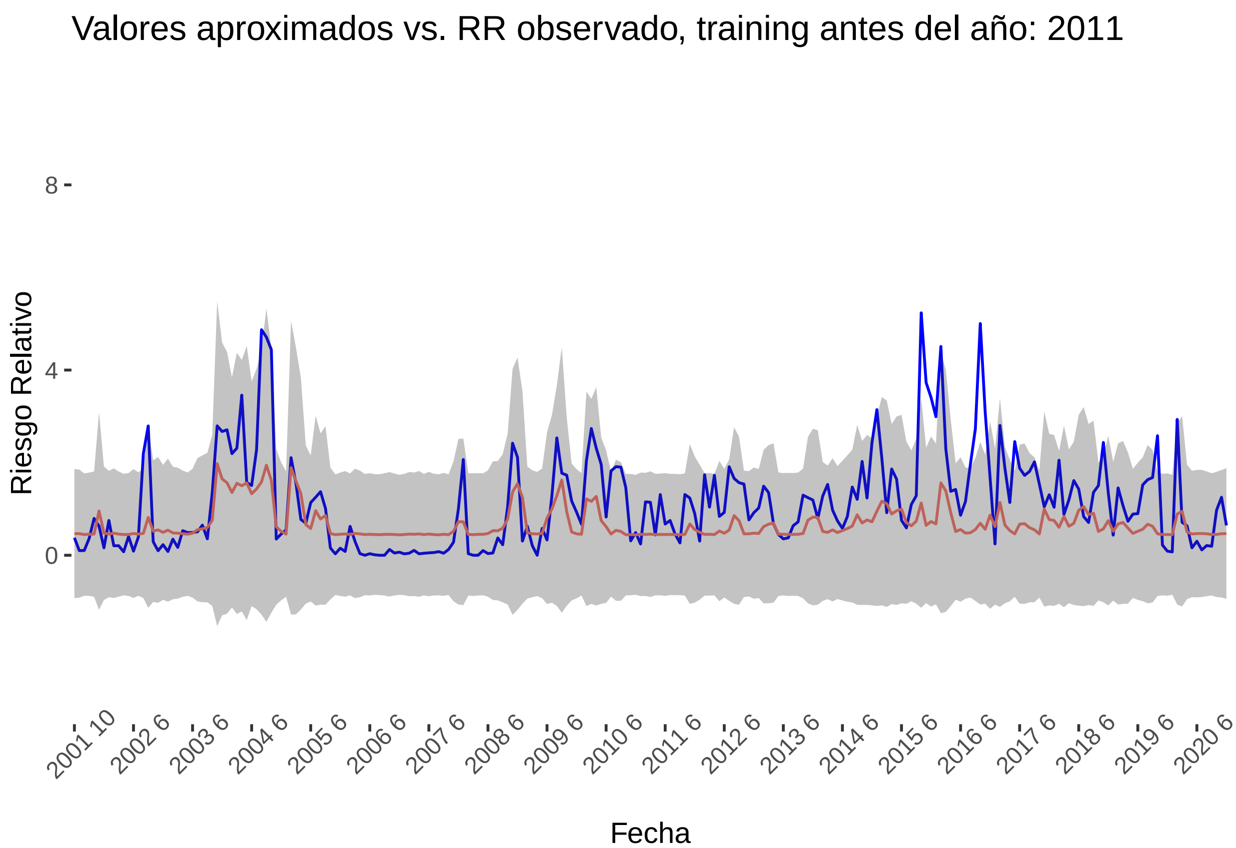



```
##
## [[10]]
```

Valores aproximados vs. RR observado, training antes del año: 2012




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



Valores aproximados vs. RR observado, training antes del año: 2011
Riesgo Relativo
8
4
0
2001 10
2002 6
2003 6
2004 6
2005 6
2006 6
2007 6
2008 6
2009 6
2010 6
2011 6
2012 6
2013 6
2014 6
2015 6
2016 6
2017 6
2018 6
2019 6
2020 6
Fecha