Intro al uso de {tidymodels}

RLadies Santiago, Chile

Sara Acevedo

Marzo 2020

Notas importantes antes de empezar

No olviden revisar el **código de conducta**. Este es un ambiente seguro y no se tolera el acoso: https://github.com/rladies/starter-kit/wiki/Code-of-Conduct#spanish

Presentaciones en Xaringan:

- https://github.com/semiramisCJ/taller_xaringan_RLadiesMty2020
- https://github.com/sporella/xaringan_github

El código estará disponible en GitHub: https://github.com/RladiesChile

Notas importantes antes de empezar

Plan para esta sesión: 60 min

Cosas que **SI** veremos hoy

- Paquetes y sus usos
- Funciones más importantes
- Implementar un modelo lineal
- Visualización básica

Cosas que **NO** veremos hoy

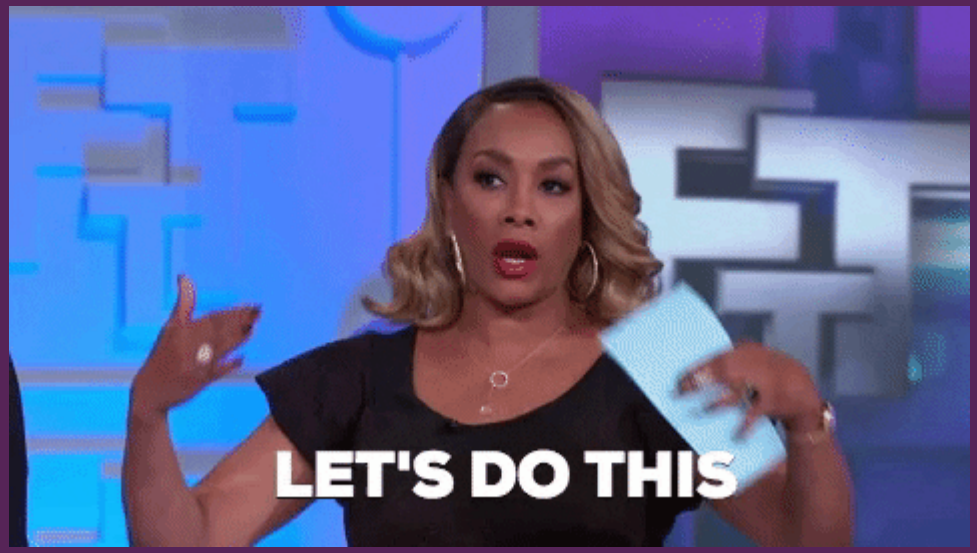
- Limpieza de datos avanzada
- Modelos complejos o ensemble
- Machine learning
- Visualización avanzada

Notas importantes antes de empezar

Si hay cosas que no entiendes del taller

- Es normal, quizás iremos algo rápido
- El código y la presentación quedará disponible
- Habrá espacios para preguntas

Empecemos



Paquetes Tidymodels

- Sintaxis tidyverse
- Reproducibilidad de datos
- Developer Max Kuhn {library(caret)}
- Hoy usaremos rsample, parsnip, recipes y yardstick
- Otros: corrr, dials, workflows, tune

Pre-Process → Train → Validate

Figura: https://rviews.rstudio.com/2019/06/19/a-gentle-intro-to-tidymodels/

- Instalar los paquetes **tidyverse**, **tidymodels**, junto con sus dependencias

```
install.packages(c("tidyverse", "tidymodels")), dependencies = TRUE)
```

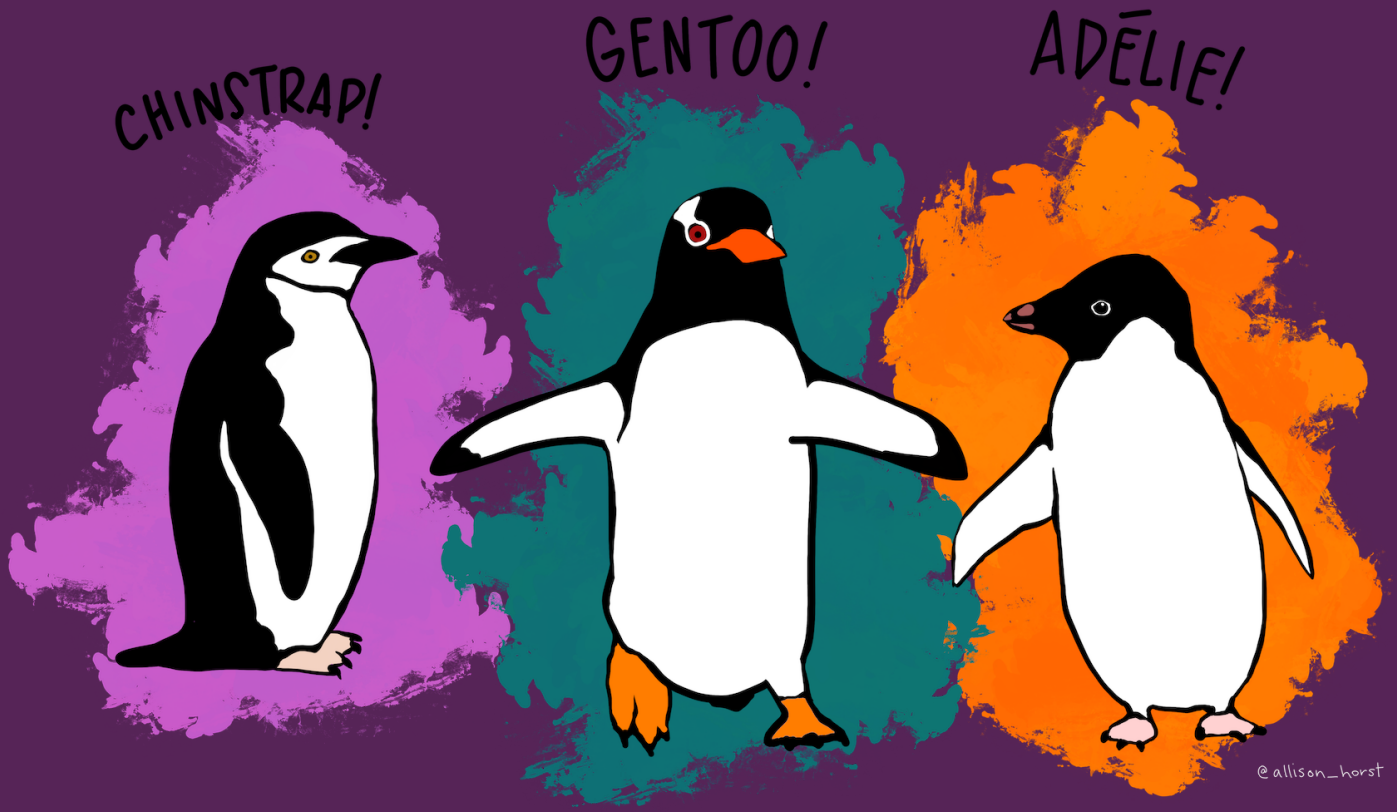
- Instalar el paquete **remotes** y **ggsignif**, junto con sus dependencias

```
install.packages(c("remotes", "ggsignif")), dependencies = TRUE)
```

- Instalar desde github el paquete **datos** y **corrr**

```
remotes::install_github("cienciadedatos/datos")
remotes::install_github("tidymodels/corrr")
```

Base de datos: pinguinos



Artwork by @allison_horst

library() y glimpse()

```r
# librerias
library(tidyverse)
library(tidymodels)
library(remotes)
library(datos)
library(ggsignif)
library(corrr)
# estilo ggplot
theme_set(theme_bw())
# cargar la database
pinguinos <- datos::pinguinos
# echar un vistazo
dplyr::glimpse(pinguinos)
```

```
## Rows: 344
## Columns: 8
## $ especie         <fct> Adelia, Adelia, Adelia, Adelia, Adelia, Adelia, Adeli…
## $ isla            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen…
## $ largo_pico_mm   <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, 4…
## $ alto_pico_mm    <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, 2…
## $ largo_aleta_mm  <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186,…
## $ masa_corporal_g <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, 4…
## $ sexo            <fct> macho, hembra, hembra, NA, hembra, macho, hembra, mac…
## $ anio            <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007,…
```

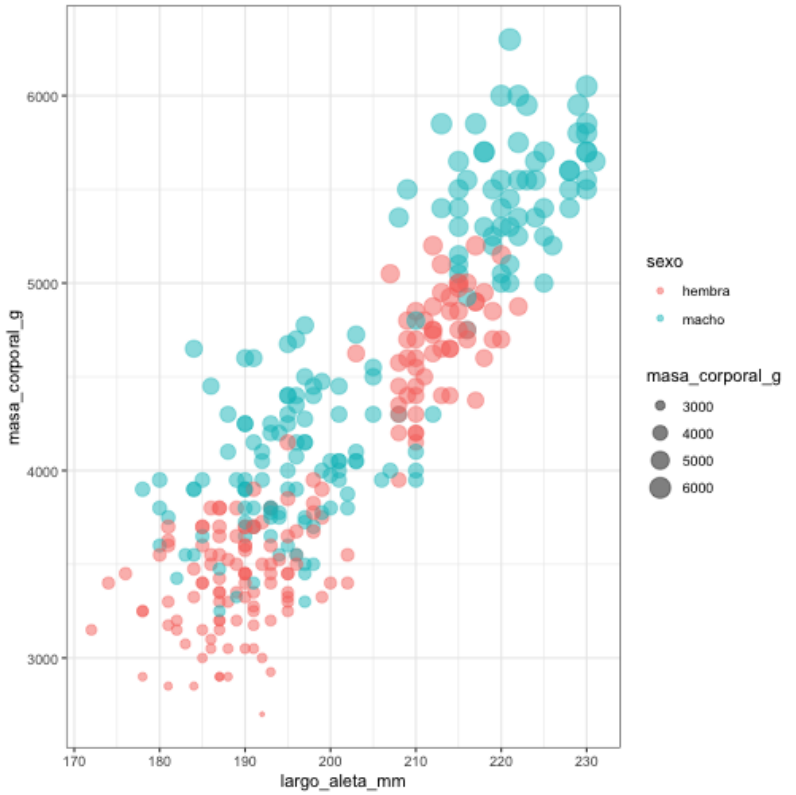
Un poco de limpieza

```
# arbitrariamente eliminaremos
pinguinos_db <-  pinguinos %>%
                 drop_na() %>% # las observaciones con datos ausentes
                 select(-anio) # la columna anio
# revisamos nuestro nuevo archivo
glimpse(pinguinos_db)
```

```
## Rows: 333
## Columns: 7
## $ especie         <fct> Adelia, Adelia, Adelia, Adelia, Adelia, Adelia, Adeli…
## $ isla            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen…
## $ largo_pico_mm   <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6,…
## $ alto_pico_mm    <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2,…
## $ largo_aleta_mm  <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 185…
## $ masa_corporal_g <int> 3750, 3800, 3250, 3450, 3650, 3625, 4675, 3200, 3800,…
## $ sexo            <fct> macho, hembra, hembra, hembra, macho, hembra, macho, …
```

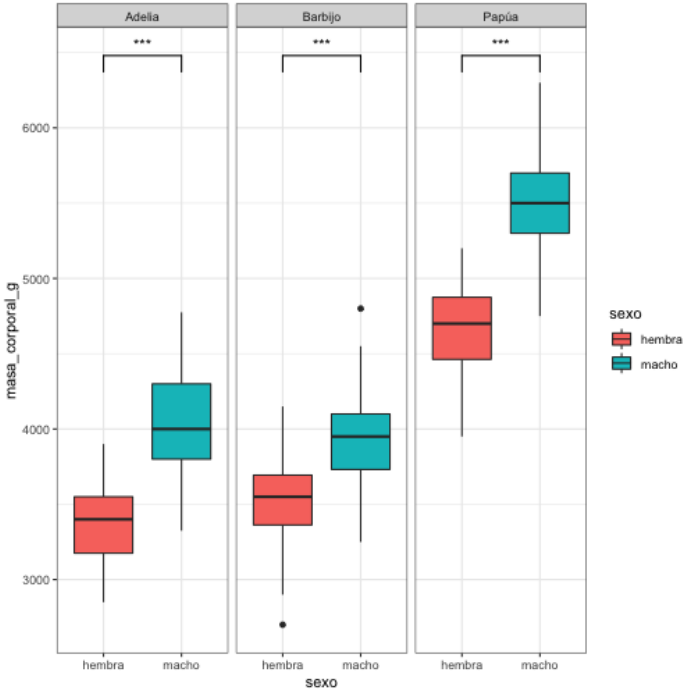
Exploramos datos visualmente

```
pinguinos_db %>% ggplot(aes(x=largo_aleta_mm, y=masa_corporal_g,
                            color = sexo, size =masa_corporal_g)) +
                 geom_point(alpha=0.5)
```



Diferencias macho y hembra por especie

```
pinguinos_db %>% ggplot(aes(x=sexo, y=masa_corporal_g, fill=sexo)) +
                         geom_boxplot() +
                         facet_wrap(~especie) +
                         geom_signif(comparisons = list(c("macho", "hembra")),
                         map_signif_level=TRUE,
                         test = "t.test")
```



Correlación entre variables numéricas

```
pinguinos_db %>%
  select(-especie,-sexo,-isla) %>%
  corrr::correlate()
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'

## # A tibble: 4 x 5
##   rowname         largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##   <chr>                   <dbl>        <dbl>          <dbl>           <dbl>
## 1 largo_pico_mm          NA           -0.229          0.653           0.589
## 2 alto_pico_mm           -0.229       NA             -0.578          -0.472
## 3 largo_aleta_mm          0.653       -0.578         NA               0.873
## 4 masa_corporal_g         0.589       -0.472          0.873          NA
```

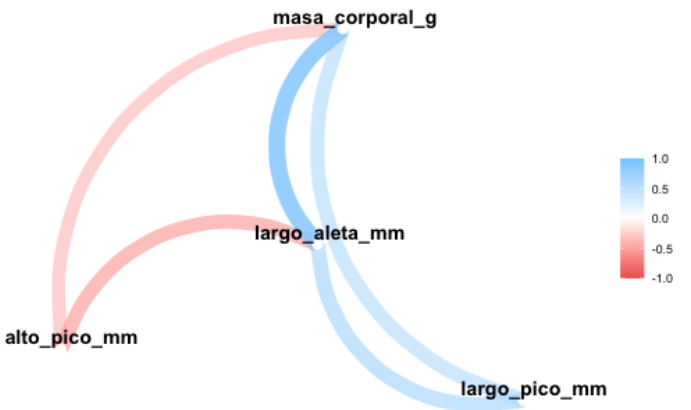
Correlación entre variables numéricas

```
pinguinos_db %>%
  select(-especie,-sexo,-isla) %>%
  corrr::correlate() %>%
  rearrange() %>%  # ordena las correlaciones
  shave() %>%# limpia las correlaciones repetidas
  fashion()
```

```
##           rowname largo_aleta_mm masa_corporal_g largo_pico_mm alto_pico_mm
## 1  largo_aleta_mm
## 2 masa_corporal_g            .87
## 3   largo_pico_mm            .65             .59
## 4    alto_pico_mm           -.58            -.47          -.23
```

Correlación entre variables numéricas

```
pinguinos_db %>%
  select(-especie,-sexo,-isla) %>%
  corrr::correlate() %>%
  network_plot()
```



Objetivo

- Predecir la masa corporal de un pinguino, en base a sus caracteristicas físicas
- Interpretar los resultados que obtendremos

Primer paso: dividir el dataset en entrenamiento y testeo

https://github.com/rstudio/hex-stickers/blob/master/thumbs/rsample.png

Dividimos el dataset en 80% entrenamiento y 20% testeo

```
set.seed(1234)
division      <-  initial_split(data = pinguinos_db, prop = .8)
entrenamiento <-  training(division)
testeo        <-  testing(division)
```

```
nrow(entrenamiento)
```

```
## [1] 267
```

```
nrow(testeo)
```

```
## [1] 66
```

Balanceo de datos

```
entrenamiento %>% count(sexo)
```

```
## # A tibble: 2 x 2
##   sexo       n
## * <fct>  <int>
## 1 hembra   126
## 2 macho    141
```

```
testeo        %>% count(sexo)
```

```
## # A tibble: 2 x 2
##   sexo       n
## * <fct>  <int>
## 1 hembra    39
## 2 macho     27
```

Balanceo de datos

```
set.seed(1234)
division      <-  initial_split(data = pinguinos_db, prop = .8, strata = sexo)
entrenamiento <-  training(division)
testeo        <-  testing(division)
```

Balanceo de datos

```
entrenamiento %>% count(sexo)
```

```
## # A tibble: 2 x 2
##   sexo       n
## * <fct>  <int>
## 1 hembra   133
## 2 macho    135
```

```
testeo        %>% count(sexo)
```

```
## # A tibble: 2 x 2
##   sexo       n
## * <fct>  <int>
## 1 hembra    32
## 2 macho     33
```

Segundo paso: crear una receta

https://github.com/rstudio/hex-stickers/blob/master/thumbs/recipes.png

Creamos una receta para usar nuestras variables

```
masa_recipe <-recipe(masa_corporal_g ~ ., data = entrenamiento) %>%
              step_corr(all_numeric()) %>%
              step_dummy(all_nominal()) %>%
              prep()

masa_recipe
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Training data contained 268 data points and no missing data.
##
## Operations:
##
## Correlation filter removed no terms [trained]
## Dummy variables from especie, isla, sexo [trained]
```

Creamos una receta para usar nuestras variables

```
entrenamiento_juice <- masa_recipe %>%
                       juice()

testeo_bake         <- masa_recipe %>%
                       bake(testeo)
```

Creamos una receta para usar nuestras variables

```
head(entrenamiento_juice, 3)
```

```
## # A tibble: 3 x 9
##   largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g especie_Barbijo
##           <dbl>        <dbl>          <int>           <int>            <dbl>
## 1          39.1         18.7            181            3750                0
## 2          39.5         17.4            186            3800                0
## 3          40.3         18              195            3250                0
## # … with 4 more variables: especie_Papúa <dbl>, isla_Dream <dbl>,
## #   isla_Torgersen <dbl>, sexo_macho <dbl>
```

```
head(testeo_bake, 3)
```

```
## # A tibble: 3 x 9
##   largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g especie_Barbijo
##           <dbl>        <dbl>          <int>           <int>            <dbl>
## 1          39.3         20.6            190            3650                0
## 2          38.9         17.8            181            3625                0
## 3          37.7         18.7            180            3600                0
## # … with 4 more variables: especie_Papúa <dbl>, isla_Dream <dbl>,
## #   isla_Torgersen <dbl>, sexo_macho <dbl>
```

Tercer paso: usar recetas y entrenar nuestros datos

https://github.com/rstudio/hex-stickers/blob/master/thumbs/parsnip.png

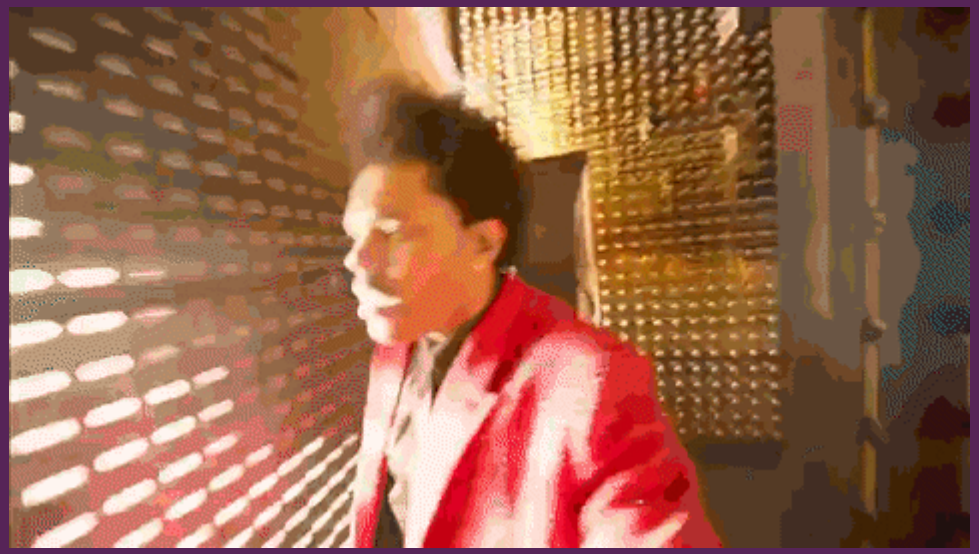
Creamos nuestro modelo

```
modelo_lineal <- linear_reg() %>%
                 set_engine("lm") %>%
                 set_mode("regression")

translate(modelo_lineal)
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
##
## Model fit template:
## stats::lm(formula = missing_arg(), data = missing_arg(), weights = missing_arg())
```

Estoy perdid@, son muchos objetos y funciones



Recapitulemos

- Datos:

```
entrenamiento #80%
testeo #20%
```

- Receta: creamos datos dummies

```
masa_recipe
```

- Nuevos set de datos

```
entrenamiento_juice
testeo_bake
```

- Creamos un modelo linear

```
modelo_lineal
```

```
ml_ajuste <- modelo_lineal %>%
             fit(masa_corporal_g ~ ., data = entrenamiento_juice)

ml_ajuste
```

```
## parsnip model object
##
## Fit time:  5ms
##
## Call:
## stats::lm(formula = masa_corporal_g ~ ., data = data)
##
## Coefficients:
##     (Intercept)    largo_pico_mm     alto_pico_mm   largo_aleta_mm
##        -1064.38            13.48            56.98            15.82
## especie_Barbijo    especie_Papúa       isla_Dream   isla_Torgersen
##         -204.94          1010.01           -15.61           -30.17
##      sexo_macho
##          439.81
```

```
lm_prediccion <- ml_ajuste %>%
                 predict(testeo_bake) %>%
                 bind_cols(testeo_bake)

lm_prediccion
```

```
## # A tibble: 65 x 10
##    .pred largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g
##    <dbl>         <dbl>        <dbl>          <int>           <int>
##  1 4054.          39.3         20.6            190            3650
##  2 3307.          38.9         17.8            181            3625
##  3 3796.          37.7         18.7            180            3600
##  4 3725.          38.8         17.2            180            3800
##  5 3219.          39.5         16.7            178            3250
##  6 3970.          39.6         18.8            190            4600
##  7 4159.          42.3         21.2            191            4150
##  8 4046.          40.6         18.8            193            3800
##  9 3306.          35.7         16.9            185            3150
## 10 4061.          42.8         18.5            195            4250
## # … with 55 more rows, and 5 more variables: especie_Barbijo <dbl>,
## #   especie_Papúa <dbl>, isla_Dream <dbl>, isla_Torgersen <dbl>,
## #   sexo_macho <dbl>
```

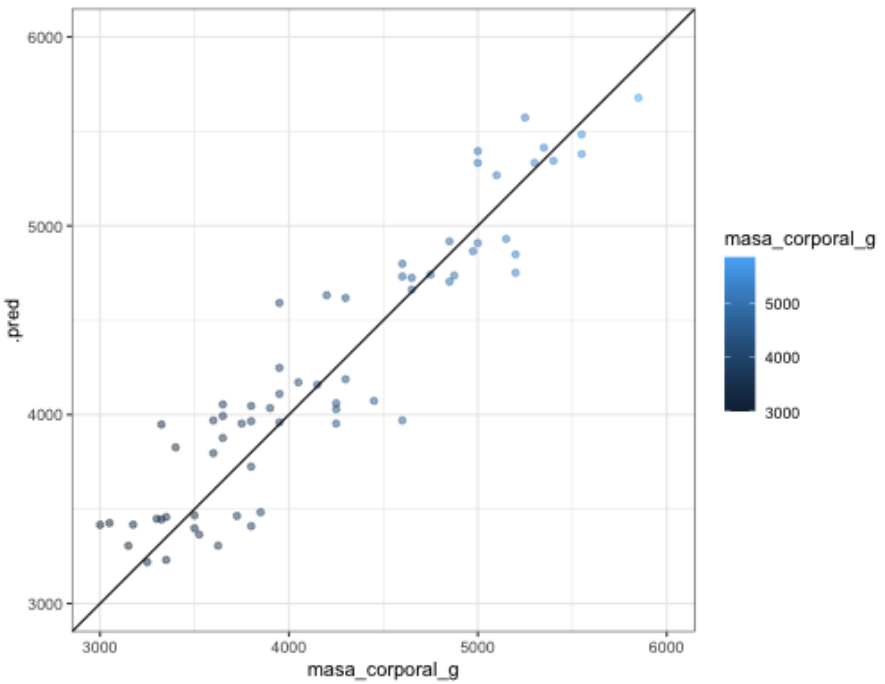
Yardstick: evaluar el modelo

https://github.com/rstudio/hex-stickers/blob/master/thumbs/yardstick.png?raw=true

```
lm_prediccion %>% metrics(truth = masa_corporal_g, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard     269.
## 2 rsq     standard       0.871
## 3 mae     standard     221.
```

```
lm_prediccion %>%  ggplot(aes(x=masa_corporal_g, y=.pred,
                             color=masa_corporal_g)) +
                   geom_point(alpha=0.5) +
                   geom_abline() +
                   coord_equal() +
                   ylim(c(3000,6000)) +
                   xlim(c(3000,6000))
```



Mas información, códigos y talleres

- Tidymodels.org
- Latin R
- Linear and Bayesian Regression Models with tidymodels package, Masumbuko Semba
- Tidymodel and glmnet, Jun Kang *https://rpubs.com/EmilOWK/tidymodels_demo
- Canal de youtube de Silvia Silge