Clase Tidymodels

Matias Bajac

2024-09-01

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.5      v recipes      1.0.10
## v dials        1.2.1      v rsample      1.2.1
## v dplyr        1.1.4      v tibble       3.2.1
## v ggplot2      3.5.0      v tidyr        1.3.1
## v infer        1.0.7      v tune         1.2.1
## v modeldata    1.4.0      v workflows    1.1.4
## v parsnip      1.2.1      v workflowsets 1.1.0
## v purrr        1.0.2      v yardstick    1.3.1
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(here)
```

```
## here() starts at /Users/matiasbajac/Desktop/Aprendizaje-Estadistico-
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library()
```

parte 2 Leer los dastos y simplificar

```
nrc = read_csv(here("Clases","Laboratorio 1", "nrc.csv" ))
```

```
## Rows: 61 Columns: 68
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): Broad.Field, Field, Institution.Name, Program.Name, Program.Websit...
## dbl (62): Program.ID, Regional.Code.1.NE.2.MW.3.SA.4.SC.5.W, Program.Size.Qu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# --- Simplificamos los nombres  y seleccionamos algunas variables
nrc <- nrc %>%
  select(rank = R.Rankings.5th.Percentile,
         research = Research.Activity.5th.Percentile,
         student = Student.Support.Outcomes.5th.Percentile,
         diversity = Diversity.5th.Percentile)
```

Exploracion de datos

Hacer un grafico de la respuesta observada contra los predictores

Que aprendemos de la relacion entre estas variables?

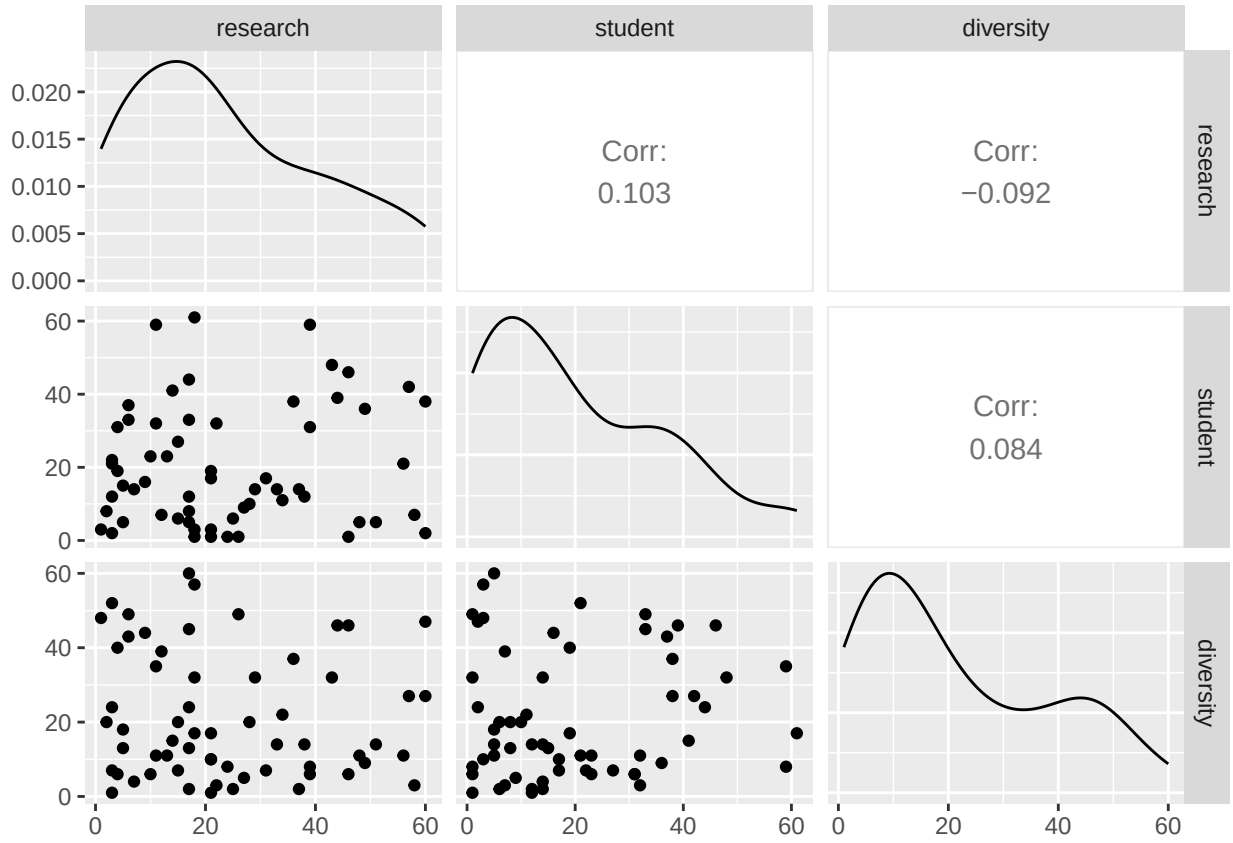
```
# rank (respuesta (y)) vs research(predictora)

a1 = ggplot(nrc,aes(x=research,y=rank)) +
  geom_point()  +
  geom_smooth(se=FALSE)

## rank vs student

a2 = ggplot(nrc,aes(x=student,y=rank)) +
  geom_point() +
  geom_smooth(se = FALSE)

a3 = ggplot(nrc, aes(x=diversity,y=rank)) + geom_point() +
  geom_smooth(se=FALSE)

## todos en una fila
```

La relacion entre rank y research es fuerte y positiva

Poca relacion con los otros predictores

Podemos usar un scatterplot matrix

```
ggpairs(nrc,columns=c(2,3,4))
```



No hay relacion entre los pares de predictores, no hay atipicos ni agrupamiento.

La distribucion de cada predictor es un poco asimetrica a la derecha

Parte 4 Especificar el modelo con parsnip

Paso 1: especificar el tipo de modelo

*Especificar el modelo (ej: modelos para regresion lineal)

*Usualmente basado en la estructura matematica

Paso 2: Especificar el motor (engine)

*Especificar el paquete o sistea para ajustar el modelo.

- La mayoria de las veces refleja el paquete

Paso 3 Declarar el modo de ser necesario

- Fijar la clase de problem, usualmente influye como se recopia la respuesta ej: si la respuesta es numerica: set_mode("regression), si la respuesta es categorica set_mode("classification")
- Este paso no se necesita si mode ya se definio en el paso 1

```
lm_mod <-
  parsnip::linear_reg() %>% # Paso 1:  Especificamos el tipo de modelo
  parsnip::set_engine("lm") # Paso 2: Especificamos el motor (engine)
lm_mod
```

```
## Linear Regression Model Specification (regression)
##
```

```
## Computational engine: lm
```

set_mode() en este caso no es necesario porque el tipo de modelo fue especificado en el paso 1

Posibles motores para regresion lineal

Existen distintas formas de especificar un modelo de regresion lineal seleccionando distintos motores

Podemos ver los motores disponibles para un posible modelo que esta disponible por defecto

```
show_engines("linear_reg")
```

```
## # A tibble: 7 x 2
##   engine mode
##   <chr>  <chr>
## 1 lm     regression
## 2 glm    regression
## 3 glmnet regression
## 4 stan   regression
## 5 spark  regression
## 6 keras  regression
## 7 brulee regression
```

Parte 5: Ajustamos el modelo

Una vez que los detalles del modelo fueron especificados, la estimacion se puede hacer con la funcion fit() (mediante expresion de tipo formula y los datos)

La formula es escrita como y ~ x donde y es el nombre de la respuesta y x es el nombre de la predictora

```
## Ajusto  el  modelo

lm_fit = lm_mod %>%  # modelo de parnsip
  parsnip::fit(rank ~ research + student + diversity, ## formula

                       data = nrc) # data frame

## Alternativamente para ajustar el modelo se puede usar fit_t (especifico predictoras y respuesta)

lm_xy_fit <-
  lm_mod %>%
  fit_xy(
    x = nrc %>% select(research, student, diversity),
    y = nrc %>% select(rank)
  )


lm_fit
```

```
## parsnip model object
##
##
## Call:
## stats::lm(formula = rank ~ research + student + diversity, data = data)
##
## Coefficients:
## (Intercept)     research      student    diversity
```

```
##      6.1025       0.5645       0.0919      -0.0573
```

El resultado de este ajuste es un **objeto** modelo de **parsnip**. Estos objetos contienen el ajuste del modelo y alguna informacion particular de **parsnip**. Se puede acceder al resumen del ajuste con:

```
lm_fit$fit %>%
summary()
```

```
##
## Call:
## stats::lm(formula = rank ~ research + student + diversity, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.048  -5.721  -3.112   4.270  40.310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.10254    3.35924   1.817   0.0745 .
## research     0.56455    0.08187   6.896 4.74e-09 ***
## student      0.09190    0.08569   1.072   0.2880
## diversity   -0.05730    0.08283  -0.692   0.4919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 57 degrees of freedom
## Multiple R-squared:  0.4796, Adjusted R-squared:  0.4523
## F-statistic: 17.51 on 3 and 57 DF,  p-value: 3.534e-08
```

Ajuste del modelo y vemos resultados

```
# alternativamente
lm_fit %>%
  extract_fit_engine() %>%
summary()
```

```
##
## Call:
## stats::lm(formula = rank ~ research + student + diversity, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.048  -5.721  -3.112   4.270  40.310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.10254    3.35924   1.817   0.0745 .
## research     0.56455    0.08187   6.896 4.74e-09 ***
## student      0.09190    0.08569   1.072   0.2880
## diversity   -0.05730    0.08283  -0.692   0.4919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.77 on 57 degrees of freedom
## Multiple R-squared:  0.4796, Adjusted R-squared:  0.4523
```

```
## F-statistic: 17.51 on 3 and 57 DF,  p-value: 3.534e-08
```

Parte 6: Explicar la relacioon entre la predictoras y la respuesta

El reporte de los coeficientes estmiados.

Se puede usar el paquete **broom** para extraer la info clave del objeto modelo en formato ordenado.

La funvcion **tidy()** retorna los parametros estimados de un objeto **lm**

```
## coeficientes estimados en formato ordenado

broom::tidy(lm_fit)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic       p.value
##   <chr>          <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   6.10      3.36       1.82  0.0745
## 2 research      0.565     0.0819     6.90  0.00000000474
## 3 student       0.0919    0.0857     1.07  0.288
## 4 diversity    -0.0573    0.0828    -0.692 0.492
```

```
lm_fit %>%  tidy()
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic       p.value
##   <chr>          <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   6.10      3.36       1.82  0.0745
## 2 research      0.565     0.0819     6.90  0.00000000474
## 3 student       0.0919    0.0857     1.07  0.288
## 4 diversity    -0.0573    0.0828    -0.692 0.492
```

Explicar la relacion entre las predictoras y la respuesta. Tiene sentido esta interpretacion? a amyores valores en research mayor es el rank

Coeficientes para **research**

Manteniendo todas las otras variables ctes (ceterius paribus), cuando **resertch** aumenta en una unidad , **rank** auenta en 0.5645 (peor) en promedio

Deberia teener mas sentido que mas research se asocie a un mejor rank (mas chico)

Es porque rank se basa en otras variables

Graficar esas variables con rank deberian dar asoc negativas

Parte 7: Whisker plot de los coeficientes

Es comun que uno vea los resultados de un regresion en tablas aunque hay muchas recomendaciones sobre visualizar los resultados y su efectividad respecto a presentarlos en tablas

Una forma de repreentar los resultados de regresion es hacer un grafico de punto y bigote (dot and whisker plot)

El paquete d**dotwhisker** permite hacer de formato sencilla estas viz para presentar y comparar resultados de modelos de regresion

Se puede hacer para graficar los coeficientes estimados u otras cantidades de interes (ej prob predichas) con un solo modelo o para diferentes modelos

Las estimiaciones son presentadas con puntos y los intervalos de confianza como bigotes

```
# Coeficientes estimados en formato ordenado
  lm_fit %>%
  dotwhisker::dwplot()
```

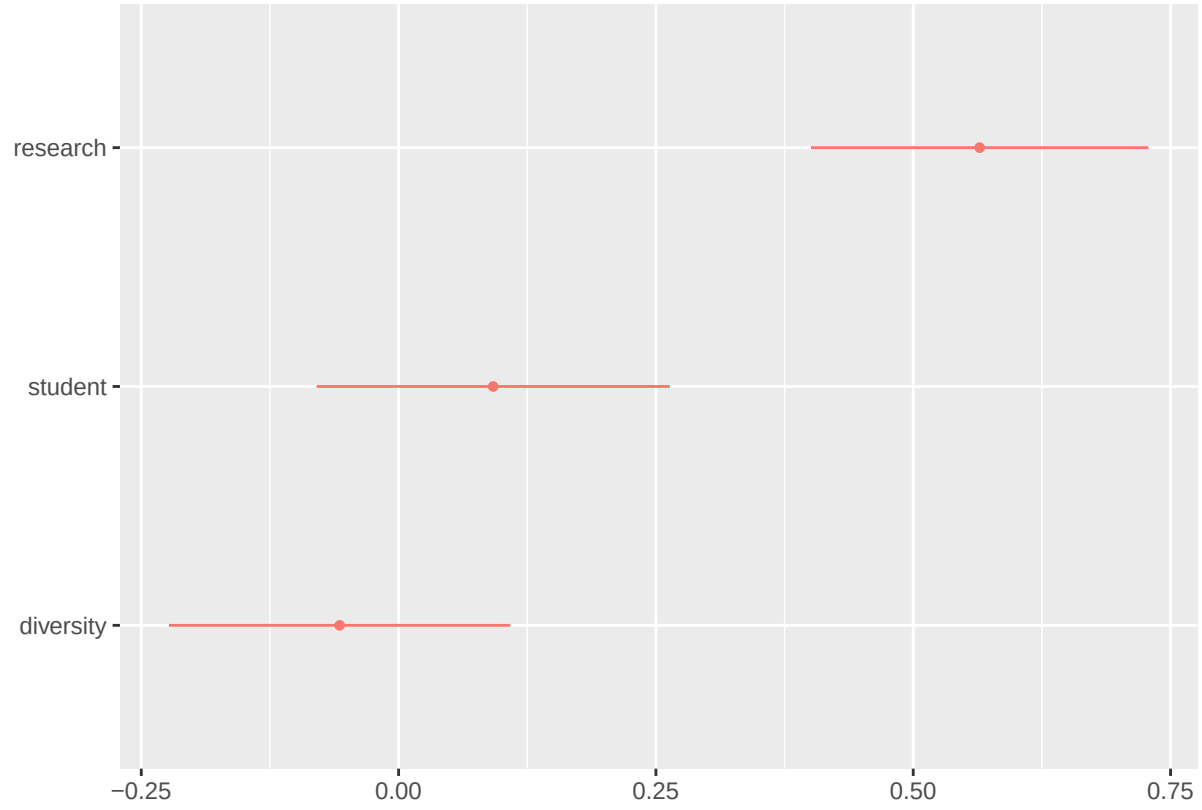
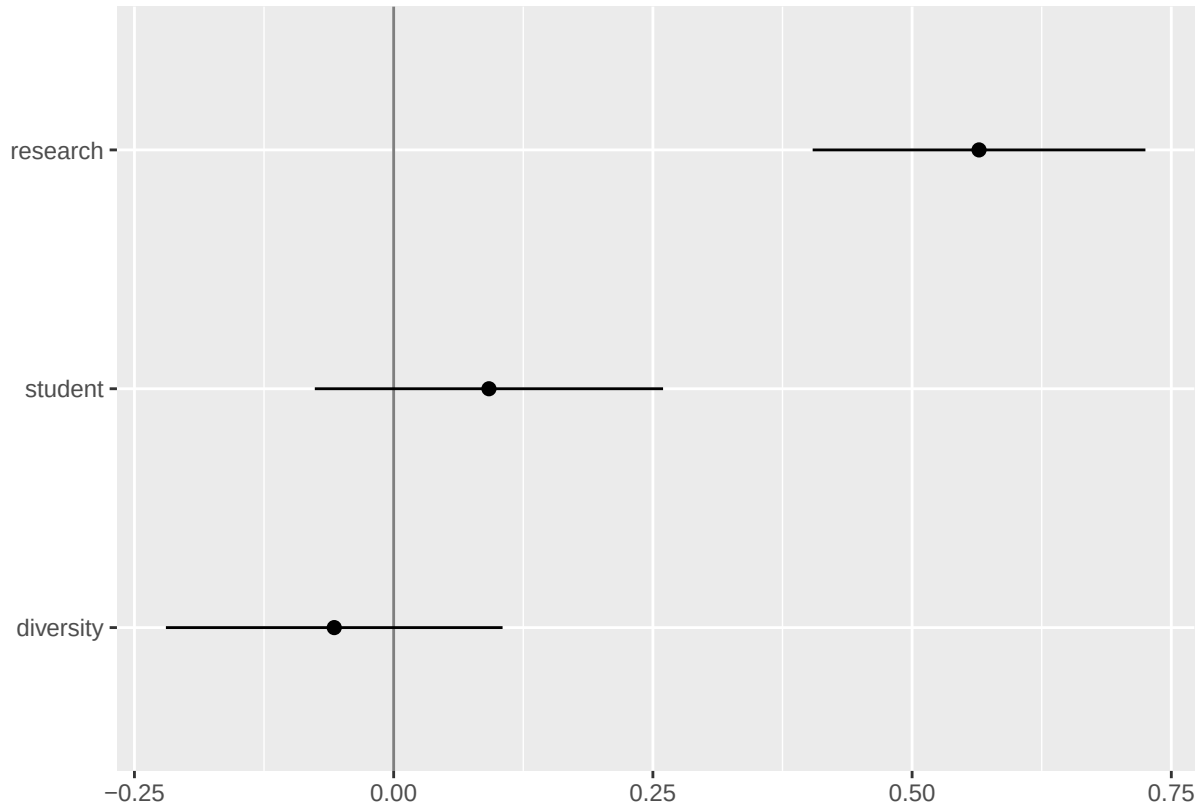


Grafico con dwplot() personalizado

Cambiamos el color e incluimos una linea de referencia vertical en el cero:

```
  # Dot-and-whisker plot
broom::tidy(lm_fit) %>%
  dotwhisker::dwplot(dot_args = list(size = 2, color = "black"),
                     whisker_args = list(color = "black"),
                     vline = geom_vline(xintercept = 0,
                                        colour = "grey50"))
```



Que aprendimos del grafico?

student y **diversity** se solapan con el 0, sugiere que estos predictores no son significtivqmente distintos de cero

El coeficiente de **research** es significativamente distinto de 0 (t grande) es la unica importante en el modelo.

Relacion positiva entre **research** y **rank**

Otras funciones de broom

```
## Para extraer residuos y valores ajustados
## para modelos de parnsip debemos dar datos para hacer predicciones

broom::augment(lm_fit,
               new_data = nrc)
```

```
## # A tibble: 61 x 6
##    .pred .resid  rank research student diversity
##    <dbl>  <dbl> <dbl>    <dbl>   <dbl>     <dbl>
##  1 18.1  17.9      36       26       1        49
##  2 10.9  36.1      47        4      31         6
##  3  9.71  1.29     11        6      33        49
##  4 20.9  -8.90     12       18      61        17
##  5 16.2   2.85     19       17      33        45
##  6 13.5  -4.52      9       10      23         6
##  7  8.84 -0.842     8        3      12         1
##  8 13.3  -6.27      7       18       3        57
```

```
##  9  7.81  2.19     10        4      19        40
## 10 24.8   1.24     26       31      17         7
## # i 51 more rows
```

El ajuste es bueno?

```
## summary mas potente
broom::glance(lm_fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic      p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>        <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.480         0.452  10.8      17.5 0.0000000353     3  -229.  469.  480.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

El modelo explica el 45% de la variabilidad de rank

Parte 9: Diagnostico

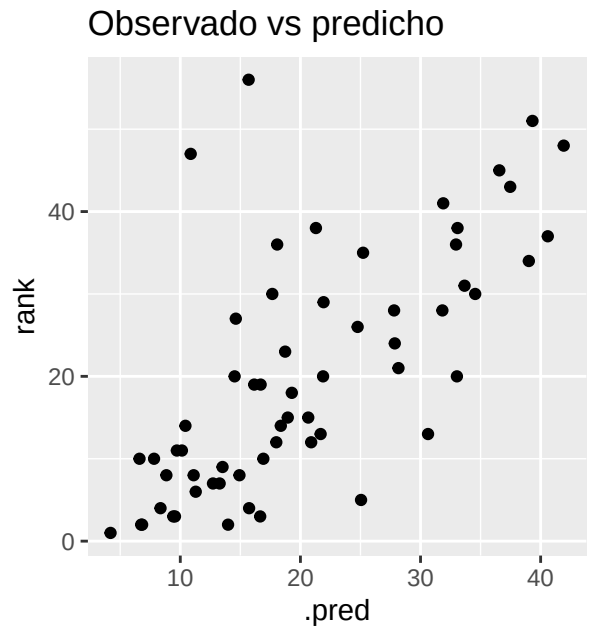
Explorar el ajuste visualmente Graficar los valores predichos vs los observados, residuos contra ajustados y predichos contra los predictores]

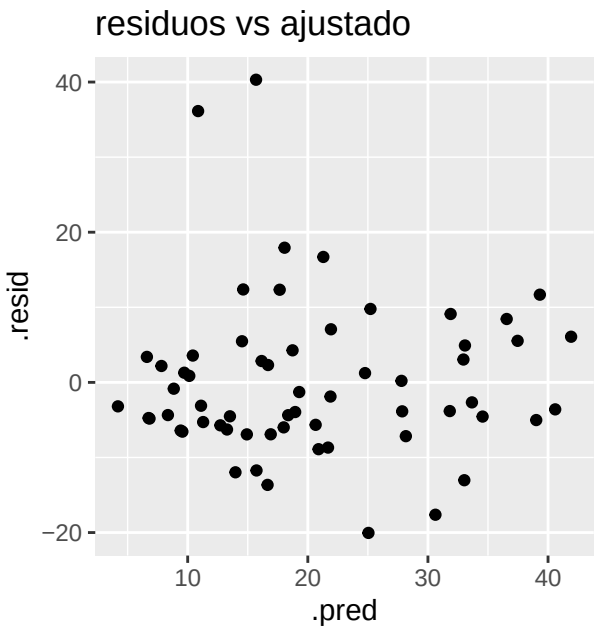
Residuos vs ajustado: se usa para identificar patrones que evidencien no linealidad de los datos

```
#Extraer los residuos y ajustados
nrc_all <- broom::augment(lm_fit, new_data = nrc)

# Ajustado  vs observado
p1 <-
  ggplot(nrc_all, aes(x = .pred, y = rank)) +
  geom_point() +
  labs(title = "Observado vs predicho") +
  theme(aspect.ratio=1)
# residuos vs. ajustado
p2 <-
  ggplot(nrc_all, aes(x = .pred, y = .resid)) +
  geom_point() +
  labs(title = "residuos vs ajustado") +
  theme(aspect.ratio=1)
require(patchwork)
```

```
## Loading required package: patchwork
```

```
p1 + p2
```





Observado vs predicho muestra ajuste razonable

Hay dos atipios, se revelan mas con el plot de los residuos

Estos son programas con rankings pobres pero que predicen mejore de lo que son

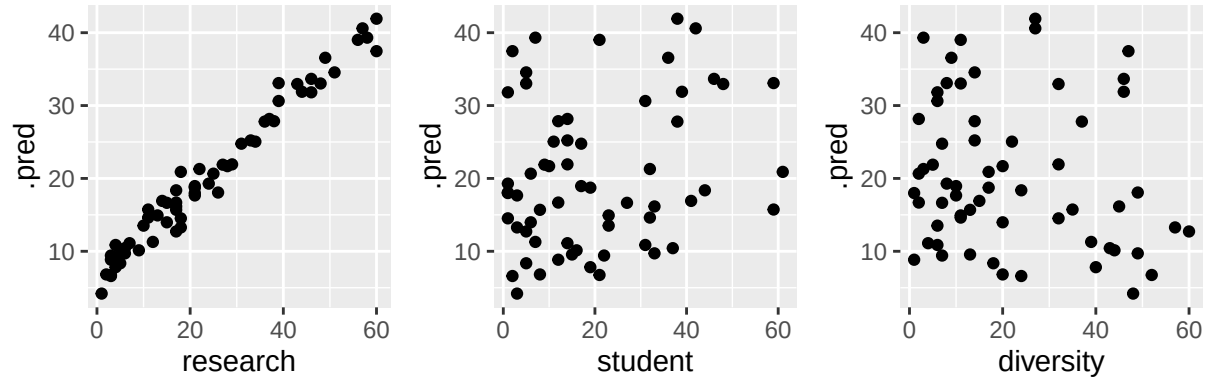
Atipicos

```
# ---saco outliers
nrc_all %>% filter(.pred < 20 & rank > 40)
```

```
## # A tibble: 2 x 6
##   .pred .resid  rank research student diversity
##   <dbl>  <dbl> <dbl>    <dbl>   <dbl>     <dbl>
## 1  10.9   36.1    47        4      31         6
## 2  15.7   40.3    56       17       8        13
```

```
# Observed (actual) vs predicted (fitted)
# Research
p3 <-
  ggplot(nrc_all, aes(x = research, y = .pred)) +
  geom_point() +theme(aspect.ratio=1)
# Student outcomes
p4 <-
  ggplot(nrc_all, aes(x = student, y = .pred)) +
  geom_point() +theme(aspect.ratio=1)
# Diversity
p5 <-
  ggplot(nrc_all, aes(x = diversity, y = .pred)) +
```

```
  geom_point() +theme(aspect.ratio=1)
p3 + p4 + p5
```



Hay una rel fuerte entre research y los valores predicos, que muestras que principalmente que el modelo usa research como predictora

##Paso 10: Generamos una data para predecir

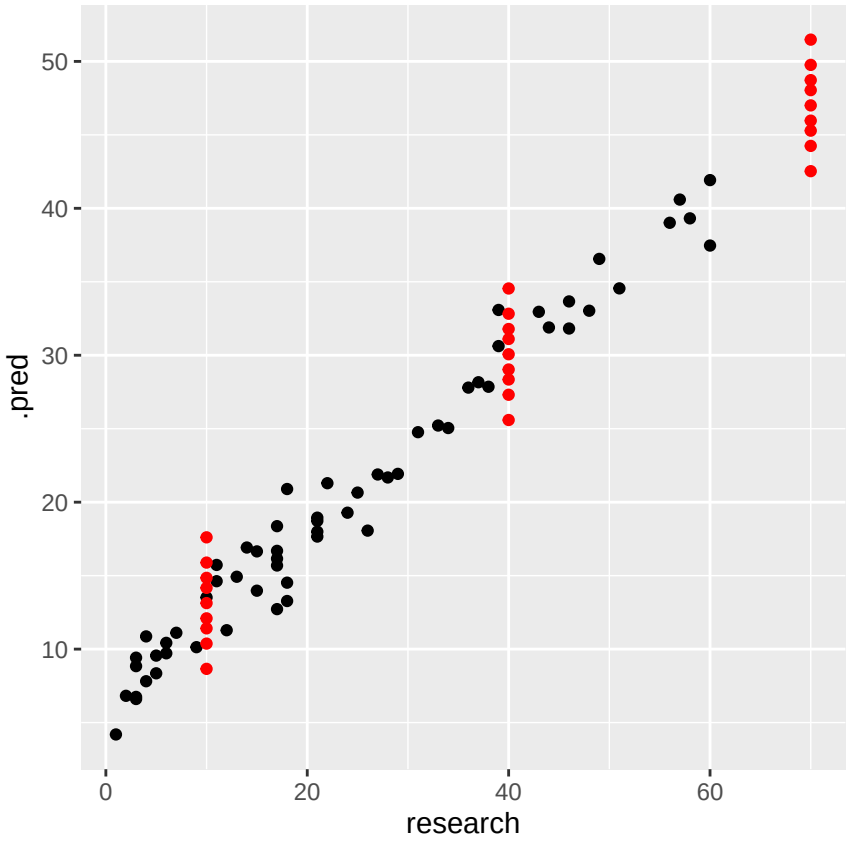
```
# --- Prediciendo nuevos datos
# Generar un nuevo conjunto de datos usamos expand_grid() )
# Mirar https://tidyr.tidyverse.org/reference/expand_grid.html
new_points <- expand_grid(research = c(10, 40, 70),
                          student = c(10, 40, 70),
                          diversity = c(10, 40, 70))


new_points
```

```
## # A tibble: 27 x 3
##    research student diversity
##       <dbl>   <dbl>     <dbl>
##  1       10      10        10
##  2       10      10        40
##  3       10      10        70
##  4       10      40        10
##  5       10      40        40
##  6       10      40        70
##  7       10      70        10
```

```
##  8       10      70        40
##  9       10      70        70
## 10       40      10        10
## # i 17 more rows
```

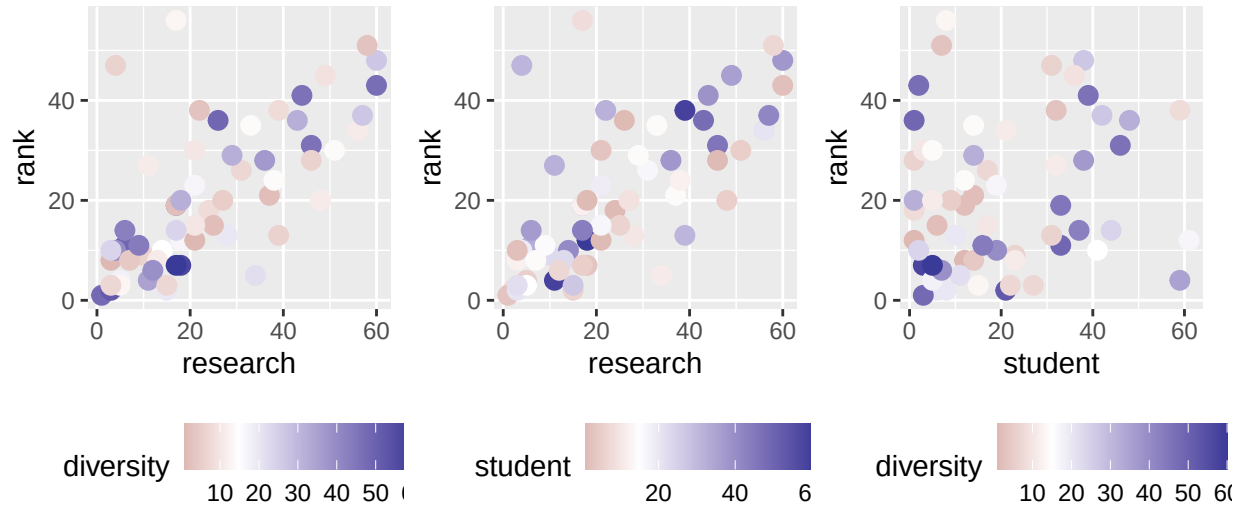
```
# Predecimos nuevos conjuntos de datos unsando el modelo corriente

mean_pred <- predict(lm_fit, new_data = new_points)

# Derivamos los intervalos de confianza

conf_int_pred <- predict(lm_fit,
                         new_data = new_points,
                         type = "conf_int")

# Extraemos los residuos y los valores ajustados,
#agregamos  argument en un data frame
new_points <- broom::augment(lm_fit, new_data = new_points)
```

```
#  Grafico los datos predichos y los observados, coloreo los nuevos puntos
ggplot() +
  geom_point(data = nrc_all, aes(x = research, y = .pred)) +
  geom_point(data = new_points, aes(x = research, y = .pred),
             colour = "red") + theme(aspect.ratio = 1)
```




```
a6 <- ggplot(nrc, aes(x = research, y = rank, color = diversity)) +
  geom_point(size = 3) +
```

```
 scale_color_gradient2(midpoint = median(nrc$diversity)) +
  theme(aspect.ratio = 1, legend.position = 'bottom')

a7 <- ggplot(nrc, aes(x = research, y = rank, color = student)) +
  geom_point(size = 3) +
 scale_color_gradient2(midpoint = median(nrc$student)) +
  theme(aspect.ratio = 1,legend.position = 'bottom')


a8 <- ggplot(nrc, aes(x = student, y = rank, color = diversity)) +
  geom_point(size = 3) +
 scale_color_gradient2(midpoint = median(nrc$diversity)) +
  theme(aspect.ratio = 1, legend.position = 'bottom')

a6 + a7 + a8
```




```
lm_fit_int <-
  lm_mod |> # modelo de parnisp
  parsnip::fit(rank ~ research*diversity , # Formula
               data = nrc) # Data frame
lm_fit_int |>
  tidy()
```

```
## # A tibble: 4 x 5
##   term               estimate std.error statistic p.value
##   <chr>                 <dbl>     <dbl>     <dbl>   <dbl>
```

```
## 1 (Intercept)        10.9       4.05         2.69 0.00929
## 2 research            0.429     0.140        3.07 0.00323
## 3 diversity          -0.188     0.136       -1.38 0.173
## 4 research:diversity  0.00636   0.00497      1.28 0.206
```

```
# Subdivido la muestra en entrenamiento y testeo usando el paquete rsample
set.seed(22)
train_test_split <- rsample::initial_split(nrc, prop = 2/3)

nrc_train <- rsample::training(train_test_split)
nrc_test <- rsample::testing(train_test_split)
```

```
#Ajustamos el modelo con los datos de entrenamiento
nrc_lm_fit <-
  lm_mod |>
  fit(rank ~ ., data = nrc_train)

# resumimos el modelo
tidy(nrc_lm_fit)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic    p.value
##   <chr>          <dbl>     <dbl>     <dbl>      <dbl>
## 1 (Intercept)   2.77       4.48      0.619 0.540
## 2 research      0.536      0.101     5.33  0.00000547
## 3 student       0.201      0.107     1.88  0.0680
## 4 diversity    -0.0259     0.104    -0.249 0.805
```

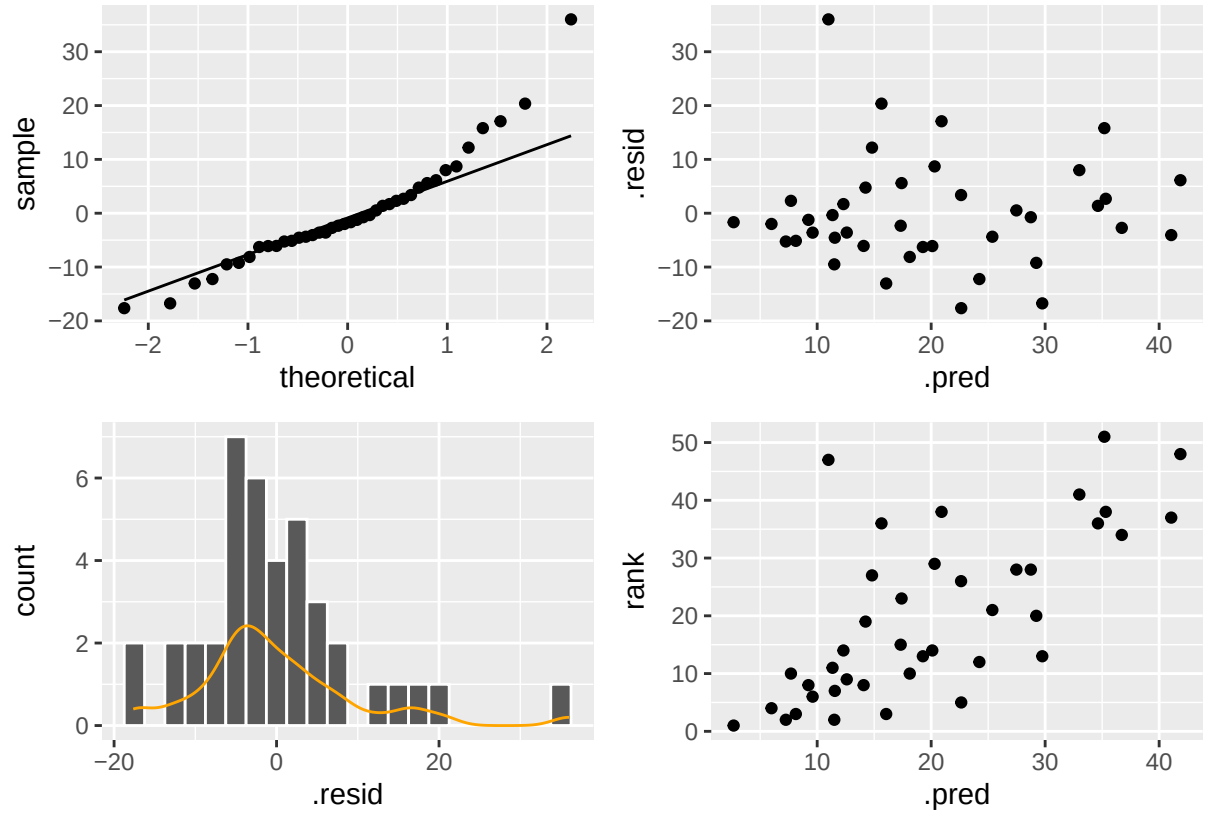
```
# broom::augment() is an easy way to get predicted
# values and residuals
nrc_lm_train_pred <- augment(nrc_lm_fit, nrc_train)
nrc_lm_test_pred <- augment(nrc_lm_fit, nrc_test)
metrics(nrc_lm_test_pred, truth = rank,
        estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      11.6
## 2 rsq     standard       0.416
## 3 mae     standard       7.11
```

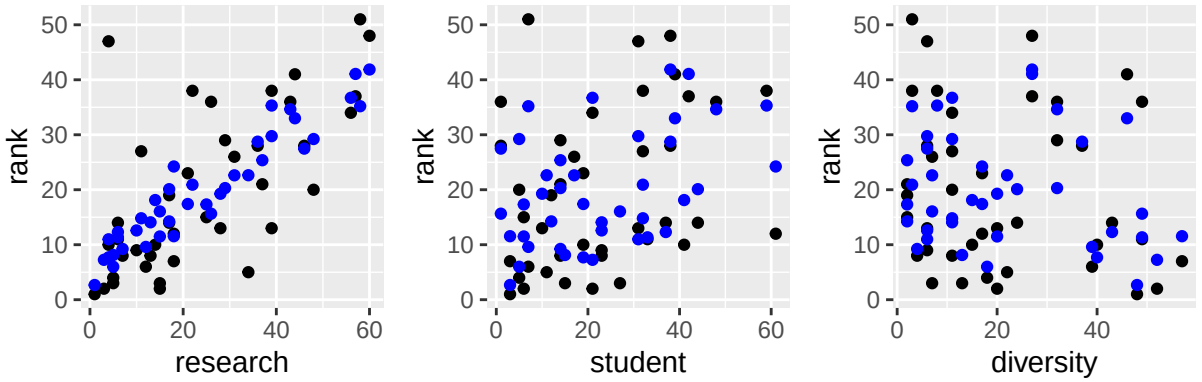
```
# Plot fitted and residuals of training data
p_f <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = .pred, y = rank))
p_e <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = .pred, y = .resid))
p_h <- ggplot(nrc_lm_train_pred, aes(x = .resid)) +
  geom_histogram(binwidth=2.5, colour="white") +
  geom_density(aes(y=..count..), bw = 2, colour="orange")
p_q <- ggplot(nrc_lm_train_pred, aes(sample = .resid)) +
  stat_qq() +
  stat_qq_line() +
  xlab("theoretical") + ylab("sample")
p_q + p_e + p_h + p_f
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
```

```
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

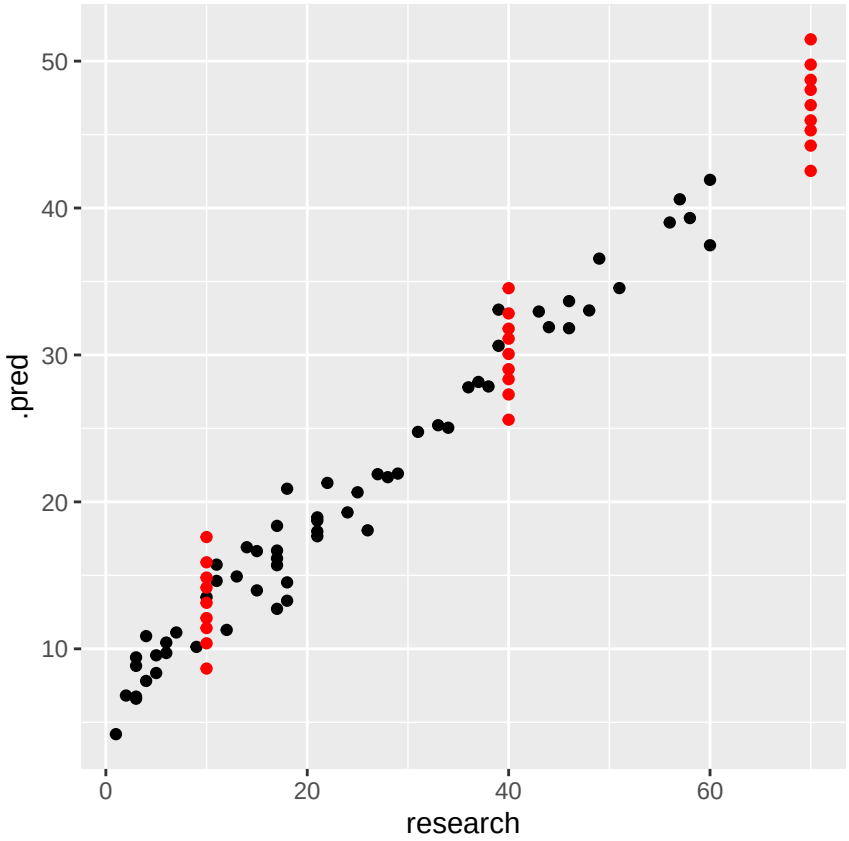



```
 # Grafico el modelo ajustado contra cada predictor
p1 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = research, y = rank)) +
  geom_point(aes(x = research, y = .pred),
             colour="blue") +theme(aspect.ratio = 1)
p2 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = student, y = rank)) +
  geom_point(aes(x = student, y = .pred),
             colour="blue") +theme(aspect.ratio = 1)
p3 <- ggplot(nrc_lm_train_pred) +
  geom_point(aes(x = diversity, y = rank)) +
  geom_point(aes(x = diversity, y = .pred),
             colour="blue") +theme(aspect.ratio = 1)
p1 + p2 + p3
```



**Hacer un grafico de los valores predichos y los valores observados y los nuevos datos colorealos en rojo

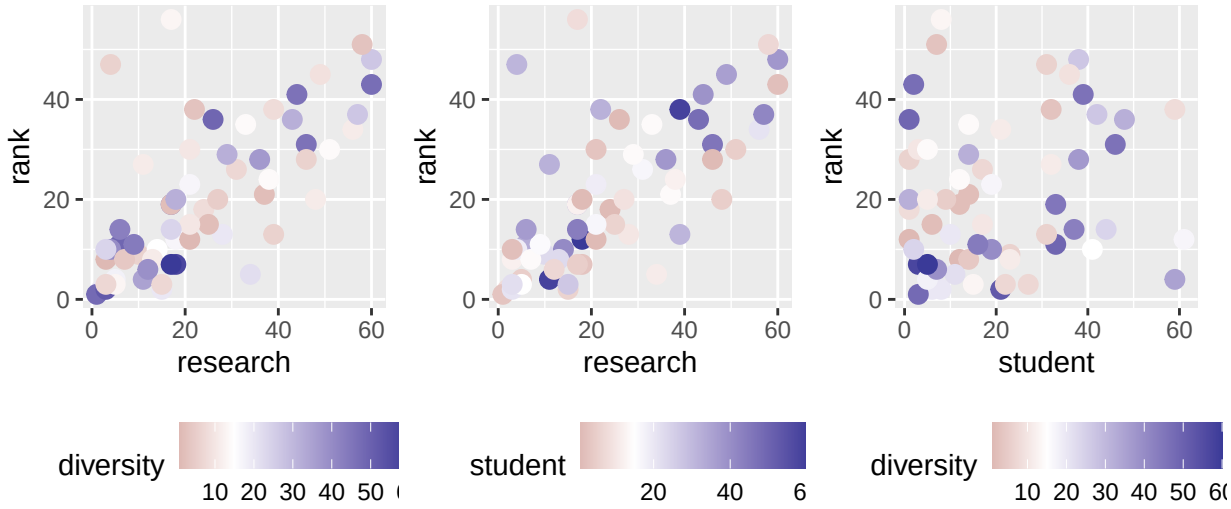
```
#  Grafico los datos predichos y los observados, coloreo los nuevos puntos
ggplot() +
  geom_point(data = nrc_all, aes(x = research, y = .pred)) +
  geom_point(data = new_points, aes(x = research, y = .pred),
             colour = "red") + theme(aspect.ratio = 1)
```



2

Explotar graficamente si hay alguna relacion entre las variables

```
a6 <- ggplot(nrc, aes(x = research, y = rank, color = diversity)) +
  geom_point(size = 3) +
 scale_color_gradient2(midpoint = median(nrc$diversity)) +
  theme(aspect.ratio = 1, legend.position = 'bottom')

a7 <- ggplot(nrc, aes(x = research, y = rank, color = student)) +
  geom_point(size = 3) +
 scale_color_gradient2(midpoint = median(nrc$student)) +
  theme(aspect.ratio = 1,legend.position = 'bottom')


a8 <- ggplot(nrc, aes(x = student, y = rank, color = diversity)) +
  geom_point(size = 3) +
 scale_color_gradient2(midpoint = median(nrc$diversity)) +
  theme(aspect.ratio = 1, legend.position = 'bottom')

a6 + a7 + a8
```



3

Incluir la interaccion en el modelo de diversidad y research

```
nrc_lm_fit <-
  lm_mod |> # modelo de parnisp
  parsnip::fit(rank ~ research*diversity , # Formula
               data = nrc) # Data frame
lm_fit_int |>
  tidy()
```

```
## # A tibble: 4 x 5
##   term               estimate std.error statistic p.value
##   <chr>                 <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)         10.9       4.05       2.69 0.00929
## 2 research             0.429     0.140      3.07 0.00323
## 3 diversity           -0.188     0.136     -1.38 0.173
## 4 research:diversity   0.00636   0.00497    1.28 0.206
```

interacccion : el efecto de una variable depende de la otra

forma mas conveniente de sacar el conkjunto de datos

entrenamiento y testeo estimaciones y predicciones

‘‘

no es bueno en ternimos de la variab explicada

obitene las predicciones y los residuos augment

ell modelo captura la estr de los datos