Introducción a la Ciencia de Datos

Guía de trabajos prácticos N°7

Introducción al modelado de datos.Modelo de regresión lineal

En esta guía, vamos a dar los primeros pasos en el vasto mundo del **modelado de datos**. Esto es un paso clave en el proceso de desarrollo de un proyecto de ciencia de datos, y una herramienta que seguirán desarrollando a lo largo de toda la carrera.

Modelar datos sirve principalmente para cuatro cosas:

1. Cuantificar (es decir, ponerle números) a un patrón o una relación que hayamos encontrado a través de un análisis de datos.
2. Explorar patrones en los datos. Podemos usar modelos para quitar las tendencias más obvios de un conjunto de datos y explorar la existencia de patrones más sutiles.
3. Resumir la información presente en un conjunto de datos para poder comunicar las conclusiones más fácilmente.
4. Realizar predicciones sobre valores de observaciones que no tenemos.

Para modelar datos, lo primero que necesitamos es especificar qué queremos modelar. En el caso de modelos supervisados, como los que vamos a ver acá, esto significa especificar qué variable (o variables, en principio puede ser más de una) queremos usar como *objetivo* para que nuestro modelo describa. El segundo paso, es definir un modelo. En su visión más simple, esto no es más que una expresión matemática que vincula las variables que no son el *objetivo*, que llamaremos variables explicativas o predictoras, o covariables, pero que tienen muchos otros nombres.

En esta materia solo veremos modelos de *regresión*, es decir donde la variable *objetivo* es continua. No discutiremos los modelos de *clasificación*, que son aquellos en los que la variable *objetivo* es discreta. Para entrar a este mundo, vamos a estudiar una familia de modelos muy importante, la más usada, y que tiene gran utlidad: los modelos lineales. En particular, hoy vamos a ver modelos lineales simples.

En R estos modelos se implementan en el paquete `stats`, en la función `lm` (de *linear model*). Además de esto, en esta guía mostramos el uso de varias funciones prácticas del paquete `modelr`.

Empecemos entonces, cargando `modelr` (`stats` se carga automáticamente) y nuestro querido `tidyverse`.

```
library(tidyverse)
library(modelr)
```

Parte 1. Modelo lineal simple

Lo básico Volvamos un dataset de la primera clase: iris. Recuerden que está incluido en el paquete `datasets` de R, que se carga automáticamente. De manera que al abrir R studio ya debería estar cargado.

Pueden ver algunas características de los datos con la función `summary`.

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
## 
```

Para esta primera experiencia, vamos a estudiar cómo varía el ancho de pétalos de flores de la especie Versicolor con respecto a su largo.

1. Filtren el dataset para quedarse solo con los datos de Versicolor; guarden el resultado en una variable llamada `df`. Hagan un gráfico del ancho de los pétalos en función de su largo.

Observen la tendencia que existe entre ambas variables. Parece razonable ajustar estos datos con una recta. Para eso, hacemos uso de la función `lm`. Esta función usa una mini sintaxis particular para definir los modelos a ajustar. En este caso, tenemos que decirle cuál es nuestra variable *target* (`Petal.Width`) y cuál es la variable explicativa (`Petal.Length`). Esto en `lm` se expresa asi usando la virgulilla de la ñ:

```
lm(Petal.Width ~ Petal.Length, data=df)
```

Noten que además de la fórmula, hay que pasar el dataset que usamos para el ajuste en el argumento `data`. Con esta sintaxis, lo que estamos haciendo es ajustar un modelo de la forma:

$$Petal.Width = a + b\, Petal.Length \quad .$$

La tarea de la función `lm` es encontrar el valor de los **parametros** `a` y `b` que "mejor ajustan" los datos.

2. Corran el código de arriba y guarden el resultado en la variable `mod` (de modelo), que vamos a usar más adelante.

La primero que podemos hacer con un modelo ajustado es usar la función `summary`. Fíjense que muchos objetos de R pueden usarse como argumentos de `summary`, que se adapta a cada uno. En este caso, nos da mucha información sobre el modelo:

```
summary(mod)
```

```
##
## Call:
## lm(formula = Petal.Width ~ Petal.Length, data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.273031 -0.073373 -0.006479  0.086271  0.295231
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.08429    0.16070  -0.525    0.602
## Petal.Length  0.33105    0.03750   8.828 1.27e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1234 on 48 degrees of freedom
## Multiple R-squared:  0.6188, Adjusted R-squared:  0.6109
## F-statistic: 77.93 on 1 and 48 DF,  p-value: 1.272e-11
```

¡Acá hay muchísima información! Concentrémonos en los valores de los coeficientes encontrados por la función (también pueden obtenerse con `coef(mod)`). Hay toda una sección dedicada a esto. En esta sección, tenemos una fila por parámetro, y para cada parámetro tenemos cuatro valores y, eventualmente, un simbolito:

- `Estimate`: el valor del parámetro que mejor ajusta los datos.
- `Std. Error`: el desvío del estimador. Es decir, el rango dentro del cual se puede mover el valor del parámetros sin que el ajuste cambie mucho (estamos siendo, a propósito, muy poco formales con estas definiciones, que ya van a ver con lujo de detalles en otras materias).
- `t value`: el **valor t**, que no es más que el cociente entre el valor del estimador y su desvío (pruébenlo).
- `Pr(>|t|)`: un **p valor** asociado con este **valor t**, que nos da una probabilidad. ¿Probabilidad de qué? Ah, es un trabalenguas, pero la respuesta es "la probabilidad de que, si no existe una relación entre ambas variables, hayamos obtenido este valor del coeficiente (o uno más extremo) por azar".
- Por último, tenemos los códigos de significancia, que aparecen como estrellitas en cada

fila, para indicar cuán significativo es cada parámetro. El código depende del **p valor**, según lo que está definido abajo.

En este caso, vemos que la pendiente tiene tres estrellitas (***), lo que significa que es un parámetro muy siginificativo. En otras palabras, sería rarísimo que, si la relación entre ambas variables no existiera, obtuviéramos un valor así de alto de la pendiente.

Las predicciones Lo primero que uno quiere hacer es ver la recta que recién ajustamos. Para eso, podemos usar la función `modelr::add_predictions`, que toma un dataset y un modelo ajustado y calcula las predicciones del modelo. El resultado aparece en una nueva variable, llamada por defecto `pred`. Corran el código de abajo para agregar las predicciones al data set.

> **Nota**: por supuesto, también podríamos haber hecho a mano el cálculo de las predicciones, tomando el valor de los coeficientes e implementando la fórmula del modelo en un `mutate`, pero esta manera es mucho más general, como vamos a ver más adelante

```
df <- df %>% add_predictions(model=mod)
```

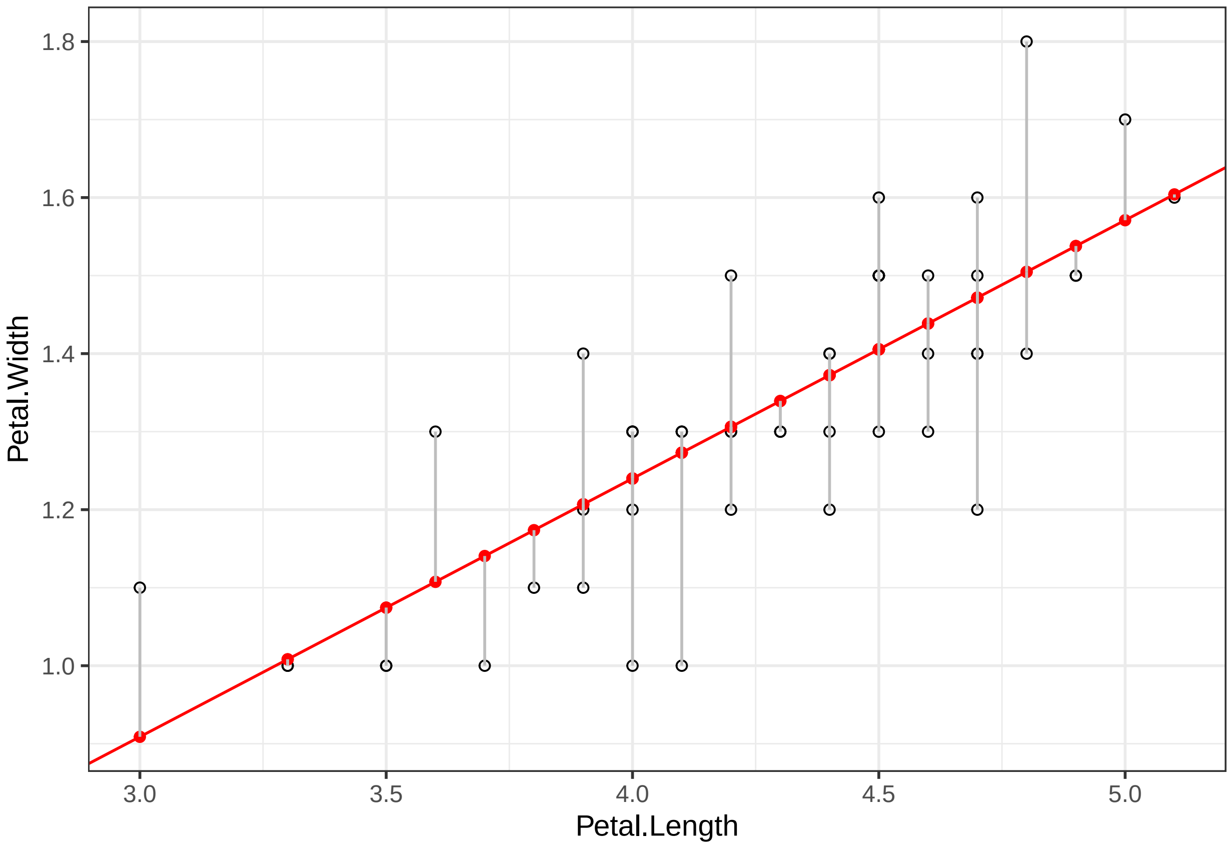
3. Creen un gráfico con los datos ajustados y agregen las predicciones como puntos rojos. Usen después `geom_line` para agregar una recta que represente la curva que ajustamos.
4. Seguramente usar tantos puntos para generar una recta es demasiado. Usen `geom_abline` (lean la docu, de ser necesario) y los valores de los coeficientes para hacer el mismo gráfico sin `geom_line`. **Ayuda**: pueden usar `coef(mod)[[1]]` y `coef(mod)[[2]]` para acceder a la ordenada al origen y la pendiente.

   > **Nota**: Fíjene que esta manera de graficar solo nos sirve si el modelo es un recta. Sin embargo, siempre se puede usar `add_predictions` **independientemtne de qué modelo estemos usando**.

Evaluación del modelo Una parte importantísima del trabajo con modelos es evaluarlos. Para eso, podemos usar varias métricas. En la salida del `summary` aparecen dos que vamos a discutir:

- El error estándar de los residuos (`Residual standard error`).
- El R^2 (`Multiple R-squared`).

Pero para esto necesitamos primero entender qué son los residuos. El residuo de una observación (*data point*) es la distancia entre el valor de la variable *target* de esa observación, y el modelo (evaludo en las variables predictoras para ese punto). Gráficamente, podemos ver los residuos como el largo de los segmentos que unen la curva del modelo con los datos:



A cada punto, le corresponde un residuo, ya sea negativo o positivo. Un gráfico que es muy importante para estudiar el comportamiento del modelo es el gráfico de los residuos:

5. Calculen los residuos como la diferencia entre cada punto y el modelo. Usen `mutate` para crear una variable nueva e incluirla en el dataset. Hagan un gráfico de los residuos en función del valor predicho (`pred`). Agreguen una línea horizontal en cero, para evaluar el modelo. ¿Encuentran algún patrón en los residuos? ¿Cuál? En base a este gráfico, ¿qué conclusión obtendrían del ajuste que hicieron?

   Nota: los residuos también pueden calcularse y agregarse a un dataframe con la función `modelr::add_residuals`.

La primera métrica da precisamente una idea de cuánto se dispersan los residuos alrededor del modelo. Se calcula como la raíz del promedio de los residuos al cuadrado, pero ajustando por la cantidad de parámetros en el modelo:

$$RMSE = \sqrt{\frac{1}{N-2}\sum_i r_i^2} \ ,$$

donde r_i es el residuo del dato i y N es la cantidad de puntos. Aparece $N-2$ porque tenemos dos parámetros en este modelo

6. Usen `modelr::add_residuals` para agregar los residuos al dataframe que venimos usando, y después usen `summarise` para calcular el valor del RMSE. Comparen con la salida del `summary` que aparece más arriba.

El R^2 es otra métrica comunmente usada para evaluar modelos; de alguna manera, representa la fracción de la dispersión de los datos que es capturada por el modelo. Su valor máximo es 1, cuando el modelo pasa por todos los puntos.

7. R base cuenta con la función `plot`, que al igual que `summarise` puede usarse con distintos objetos. Vean qué pasa cuando se usa con un modelo y con un dataframe.

Parte 2. Modelo lineal múltiple

Dos variables continuas Por supuesto, no es necesario usar una única variable para intentar modelar el target. Si tenemos más de una variable explicativa, podemos usar ambas y entrarlas en la fórmula de `lm` como `y ~ x1 + x2`, por ejemplo.

8. Elijan alguna de las otras variables del dataset de iris (filtrando solo Versicolor todavía) como variable explicativa adicional. Usen gráficos de los residuos en función de estas variables para intentar elegir una.

   Nota: no hay una respuesta clara, ninguna de las variables es muy interesante, pero una tiene un atisbo de algo mejor.

9. Usen `lm` para ajustar el modelo con esa variable adicional (llámenlo `mod2`) y vean cómo cambian los parámetros y cuáles son significativos. ¿Cómo cambia el R^2 y la dispersión de los residuos?

En base a estos resultados, seguro estamos tentadxs a elegir un modelo sobre otro. Pero la cosa es mucho más complicada, lamentablemente. Cuantos más parámetros tenga el modelo, mejor reproducirá los datos, independientemnte de qué parámetro usemos para las variables predictoras. En dos semanas veremos formas más rigurosas de **comparar y elegir modelos**.

10. Usen las funciones de `modelr` para calcular las predicciones de este nuevo modelo, sus residuos y realizar un gráfico similar al de los puntos 3 y 4. ¿Qué problema encuentran?

El problema para hacer este tipo de gráfico es que para cada valor de la primera variable predictora (`Petal.Length`), el modelo toma una infinidad de valores, dependiendo del valor de la segunda variable predictora. Una posible solución es hacer una recta para cada uno de una serie de valores de la segunda variable.

Si tienen nombres estándares para todo, prueben este código para realizar un plot para cinco valores de la segunda variable predictora (aquí `Sepal.Width`, pero tal vez ustedes usaron otro valor). Vamos a crear una grilla de valores y calcular las predicciones sobre esa grilla, en lugar de sobre

```r
# Creo una grilla de valores, con todos los valores de Petal.Length, pero
# solo cinco valores de la segunda variable
grilla <- data_grid(df, Petal.Length,
                    Sepal.Width = seq_range(Sepal.Width, n=5)) %>%
  add_predictions(mod2)

# Hago el gráfico, agrupando por valor de la segunda variable
```

```
df %>% ggplot(aes(x=Petal.Length, y=Petal.Width,
                  color=Sepal.Width)) +
  geom_point() +
  geom_line(data=grilla, aes(y=pred, group=Sepal.Width))
```

Para cada uno de los cinco valores de la segunda variable, tenemos una recta diferente. Fíjense que la pendiente de las rectas no cambia, solo la ordenada al origen. Vamos a ver mayor flexibilidad cuando incluyamos modelos con términos de interacción, en un par de semanas.

11. Para analizar los residuos de este modelo, aparece una nueva diferencia con el modelo simple. Hagan el gráfico de los residuos en función de cada una de las variables. ¿Observan algún patrón adicional?
12. Un gráfico que puede hacerse siempre, **independientemente de la cantidad de variables predictoras** que tenga el modelo, es el de los residuos en función de la predicción del modelo. Creen este gráfico y discutan las diferencias con lo del punto anterior.

Una variable continua y una discreta Ahora volvamos al dataset de Iris original y retengamos todas las especies.

11. Hagan un gráfico del *target* en función del largo del pétalo y coloreen según la especie.
12. Hagan un modelo idéntico al del punto 1. Comparen los residuos y el R^2.
13. Ahora incluyen a la especie de la flor como una variable descriptora adicional. Esto se hace **exactamente de la misma manera** en la sintaxis de `lm`: `Petal.Width ~ Petal.Length + Species`.
14. Obtengan el `summary` del modelo. ¿Cuántos parámetros tiene este modelo? ¿Cómo se compara con el modelo de dos variables contínuas?
15. Hagan los gráficos correspondientes:

a. Residuos en función de las predicciones.
b. Predicciones del modelo en un gráfico de `Petal.Width` vs. `Petal.Length`, usando colores para cada especie.

Para trabajar en casa

Vamos a trabajar con una versión filtrada del conjunto de datos de `insurance`. En esta versión (disponible en el campus), dejamos solo a los no fumadores que siguen la tendencia global.

El **objetivo** de este ejercicio es que usen las herramientas de modelado descubrir nuevos patrones en este conjunto de datos y para cuantificar relaciones.

1. Para empezar, realicen una gráfico de `charges` en función de `age`, y evalúen el aspecto de la relación que ya conocemos.

2. Ajusten un modelo lineal a estos datos, usando solo la edad como variable explicativa. Usen la fórmula `charges ~ age`, y evalúen el modelo y sus residuos.

Ahora, vamos a crear nuevas variables a partir de `age`, que tal vez resulten más útiles para describir la relación. Por ejemplo, para crear variables que sean potencias de `age`, pueden usar `I(age**2)` (la `I` es importante porque de otro modo la minisintaxis de `lm` considera que estamos hablando de variables de interacción, tema para dentro de un par de semanas). También se puede usar `poly(x, 2)` en la fórmula; por ejemplo `charges ~ poly(x, 2)` para ajustar con un polinomio de grado dos.

4. Usen alguna fórmula de las de arriba para ajustar los datos. Comparen con el modelo del punto 2. Evalúen los valores de RMSE y de R^2.

5. Ahora agreguen alguna otra variable, categórica o contínua y vean cómo cambia el ajuste. Busquen la mejor descripción de los datos posible, en base a los parámetros RMSE y R^2.

   Nota: pueden usar `facet_grid()` o `facet_wrap()` para graficar el modelo separadamente para cada valor de las variable categóricas. Vean el libro de Wickham.

6. Mirando los coeficientes del modelo definitivo y su significancia, interpreten qué patrón están viendo en el modelo.

Entrega número 6.

- Escriban un párrafo de no más de 100 palabras explicando qué variables y/o transformación de variables usaron para el mejor modelo definido arriba.
- Preparen y entreguen un gráfico en el que se vean los datos y las predicciones del mejor modelo.
- Presenten una tabla con los valores de los parámetros ajustados y las métricas que consideren útiles.
- Agreguen un párrafo de no más de 250 palabras interpretando el modelo y explicando el significado de los valores obtenidos para los parámetros.