



Clase 7

Exploración y modelo lineal

Miriam Lerma

Marzo 2021

Intro

- Explorar datos.
- Modelo lineal

Ustedes









- Conocimientos de R (saben abrirlo, cargar paquetes y datos, saben hacer operaciones y gráficos).
- Quieren conocer explorar datos y conocer la sintaxis para hacer modelos lineales en R.

Notas

Ya vieron teoría, hoy es solo para que practiquen en R.

Recuerden que los modelos dependen de sus preguntas y experimentos o muestreos.

Créditos & materiales:

-  Ejemplos de regresiones lineales simples
Sthda por Alboukadel Kassambara
-  Ejercicios de estadística con R
Matias Andina
-  Libro
Handbook of Regression Models in People Analytics
-  Tutoriales diversos
STAT 545
-  Ejercicios practicos
ourcodingclub
-  Outliers
Rocio Joo
- Imágenes adicionales
 Unsplash
 Portada por Kristine Wook

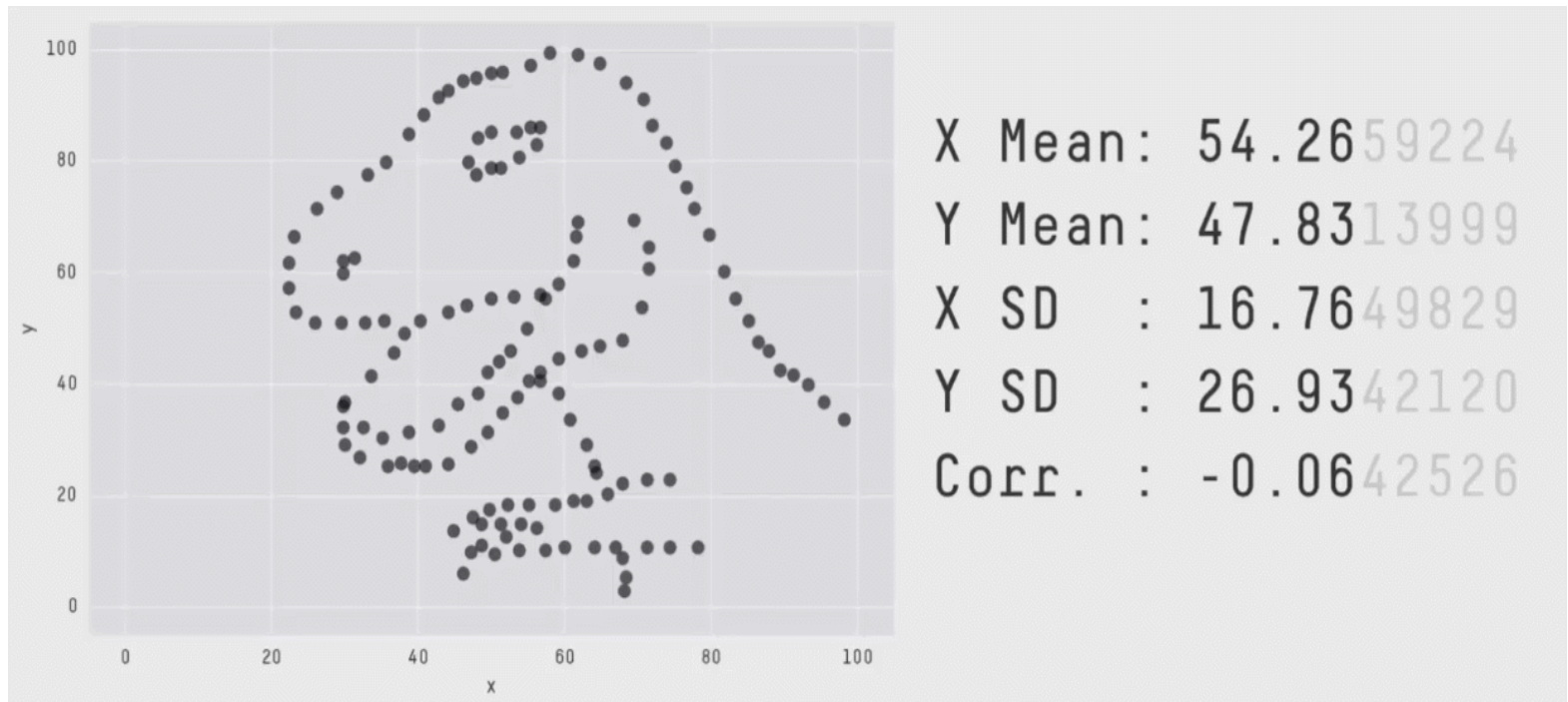


Exploración

1.1. Inspecciona

Siempre inspecciona tus datos!

Todos estos gráficos tienen medias, desviaciones estándar y una correlación entre puntos similares.



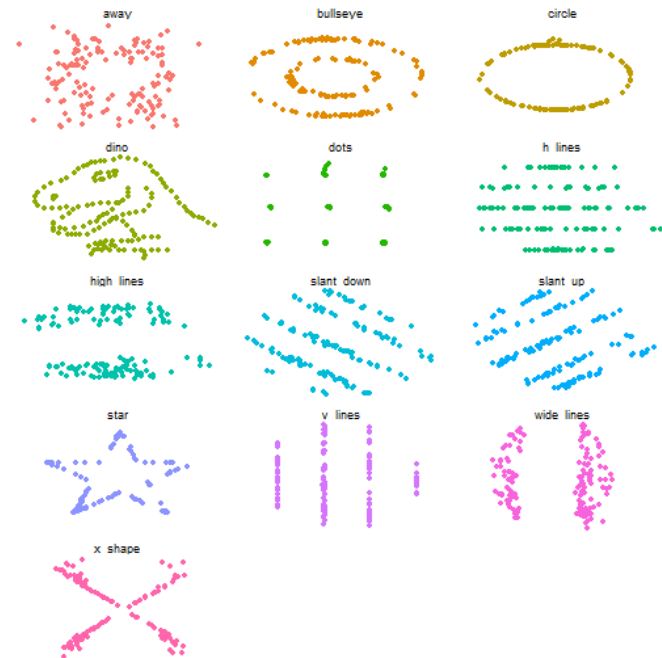
1.1. Inspecciona

Alberto Cairo creo este paquete (datasauRus) para ilustrarlo.

Si quieren replicar algunas graficas:

```
#install.packages('datasauRus')  
library(datasauRus)
```

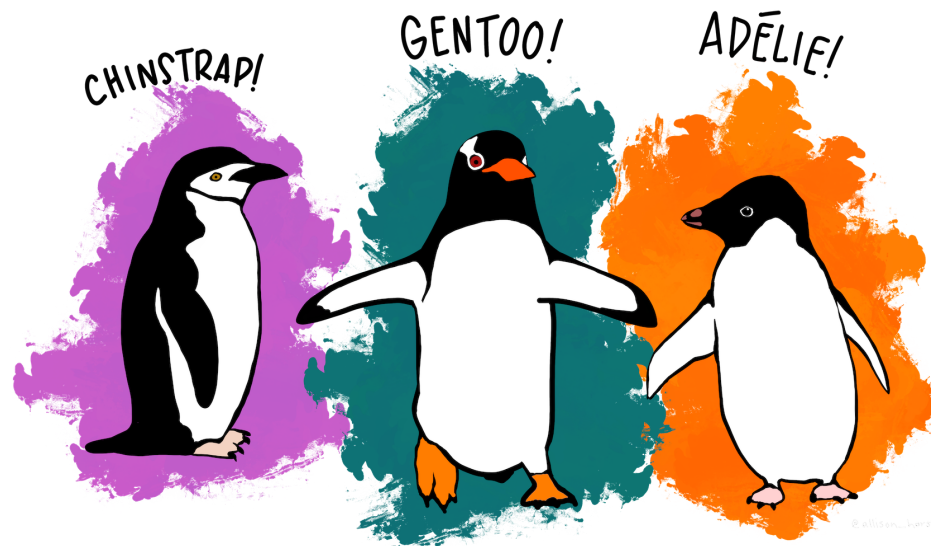
```
ggplot(datasaurus_dozen,  
       aes(x=x, y=y,  
           colour=dataset))+  
  geom_point()+  
  theme_void()+  
  theme(legend.position = "none")  
  facet_wrap(~dataset, ncol=3)
```



1.2. Pinguinos

Exploremos los datos de pinguinos.

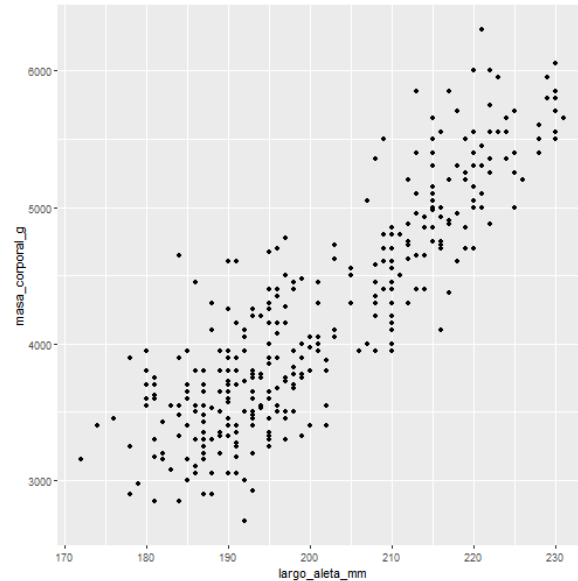
```
library(ggplot2)  
library(datos)  
Pinguinos<-datos::pinguinos
```



1.2. Pingüinos

Recordemos como se realizan los gráficos de puntos.

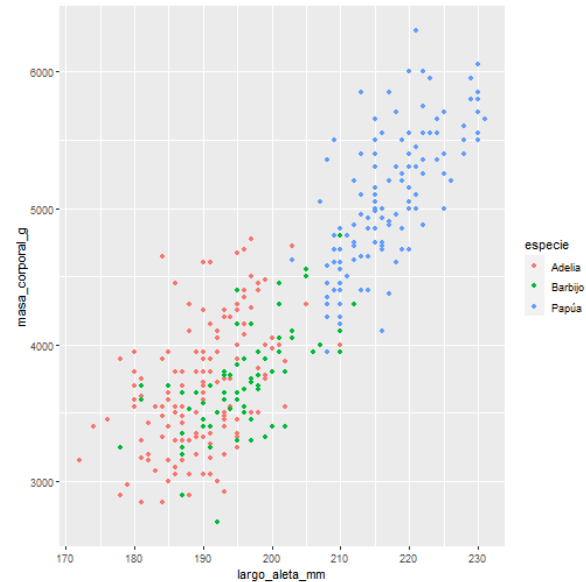
```
ggplot(Pingus) +  
  aes(x = largo_aleta_mm,  
      y = masa_corporal_g)+  
  geom_point()
```



1.2. Pinguinos

Sabemos que hay tres especies, separemos las especies por colores.

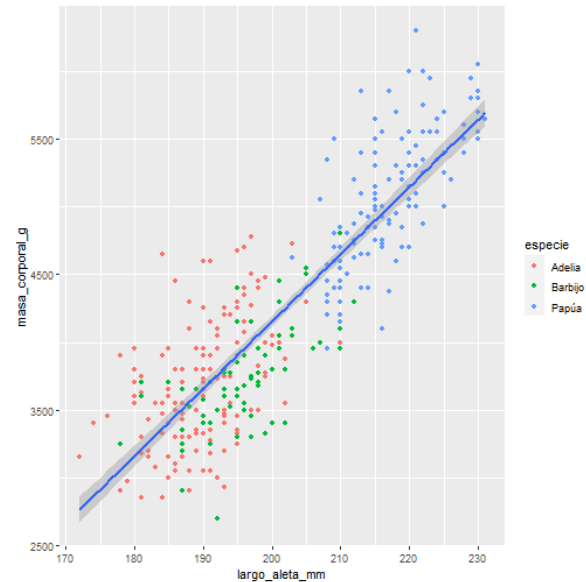
```
ggplot(Pingus,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g,  
      color=especie))+  
geom_point()
```



1.2. Pinguinos

Si agregamos una nueva capa con la línea de tendencia, especificamos un ajuste lineal ("lm") podemos ver como se relacionan estos datos. No obstante! tenemos datos de tres especies diferentes!

```
ggplot(Pingus,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g)) +  
  geom_point(aes(color = especie)) +  
  geom_smooth(method="lm")
```

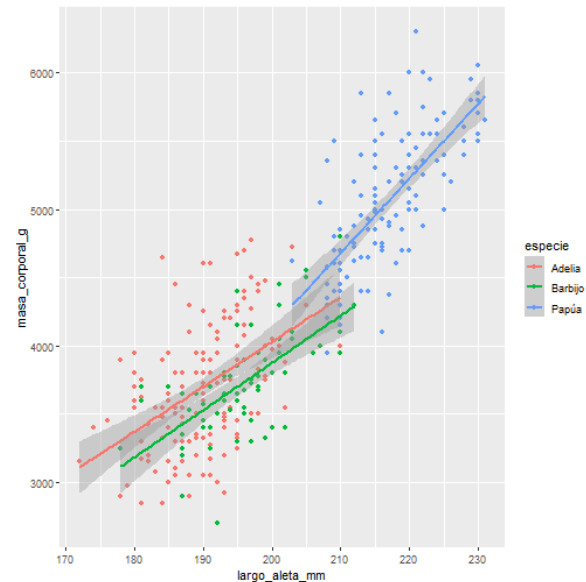


1.2. Pinguinos

Cambiando algunos argumentos nos permite explorar y obtener diferentes resultados gráficos usando los mismos datos.

Por ejemplo, si cambiamos la ubicación del color, le decimos que me haga líneas por especies.

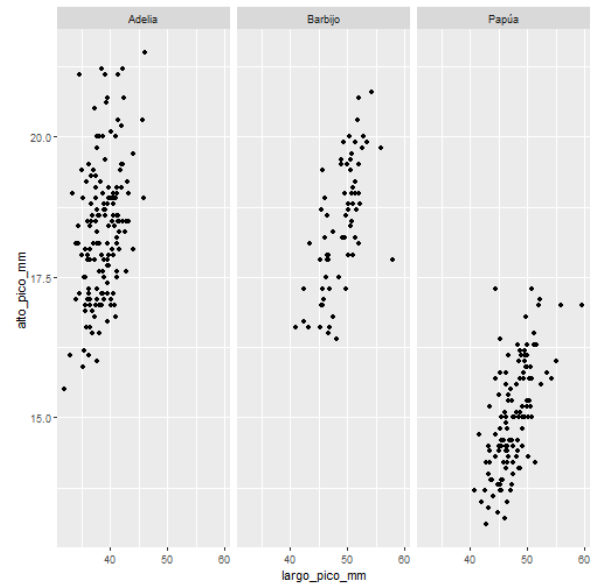
```
ggplot(Pingus,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g,  
      color = especie)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```



1.3. facet_wrap

`facet_wrap` es un argumento que nos permite ver variables categoricas separadas por panel.

```
ggplot(Pingus,  
       aes(x=largo_pico_mm,  
           y=alto_pico_mm)) +  
  geom_point()+  
  facet_wrap(~especie)
```



1.4. cowplot

Noten que al usar `facet_wrap` los paneles se acomodan de cierta manera que no es fácil de cambiar, para cambiar como están acomodados podemos usar `cowplot` o `patchwork`.

Deben instalarlo antes.

```
#install.packages("cowplot")  
library(cowplot)
```

Guardar plots con nombres e incluirlos en una sola figura.

```
cowplot::plot_grid(p1, p2, p3, p4,  
                  labels = "AUTO") #<< Agrega letras
```

1.5. plotly

Cargar libreria.

```
#install.packages(plotly)  
library(plotly)
```

Crear ggplot.

```
Pingus_puntos<-ggplot(Pingus,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g,  
      color = especie)) +  
  geom_point()
```

La funcion **ggplotly** te permite inspeccionar tu grafico de manera interactiva.

```
ggplotly(Pingus_puntos)
```

Vamos a ver un ejemplo en los ejercicios.

Ejercicios

- Cargar datos de pingüinos
- Crear dos gráficos de puntos con líneas de regresión
- Usar `facet_wrap`
- Crear un gráfico de puntos interactivo usando `plotly`

1. 6. Ejercicios

Cargar datos desde el paquete, usar `read_csv`, o `import dataset`.

```
library(datos)
library(tidyverse)
Pingus<-datos::pinguinos
```

Crear un gráfico.

```
ggplot(Pingus,
       aes(x=largo_aleta_mm,
           y=masa_corporal_g,
           color = especie)) +
  geom_point() +
  geom_smooth(method = "lm")
```

Cambiamos el orden de los argumentos.

```
ggplot(Pingus,
       aes(x=largo_aleta_mm,
           y=masa_corporal_g)) +
  geom_point(aes(color =especie))+
  geom_smooth(method="lm")
```


1. 6. Ejercicios

Ver variables categoricas separadas por panel.

```
ggplot(Pingus, aes(largo_pico_mm, alto_pico_mm)) +  
  geom_point()+  
  facet_wrap(~especie)
```

Exploremos los datos usando solo los datos de Pinguinos de Adelia.

```
Adelia<-Pingus%>%  
  filter(especie=='Adelia')
```

```
ggplot(Adelia,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g)) +  
  geom_point()+  
  geom_smooth(method="lm")
```

1. 6. Ejercicios

Cargar libreria.

```
#install.packages(plotly)  
library(plotly)
```

Crear ggplot.

```
Pingus_puntos<-ggplot(Pingus,  
  aes(x=largo_aleta_mm,  
      y=masa_corporal_g,  
      color = especie)) +  
  geom_point()
```

La funcion **ggplotly** te permite inspeccionar tu grafico de manera interactiva.

```
ggplotly(Pingus_puntos)
```



Modelos lineales

2. Modelos lineales

Recordatorio:

- La realidad es multidimensional, compleja e incierta.
- Un modelo es una representación formal de un fenómeno, una reducción de dimensionalidad que posee utilidad práctica.
- Dicha representación normalmente puede ser condensada en una expresión matemática, una fórmula, que indica cómo una variable se relaciona con otra(s).

2.1. Generar datos

Cuando busquen ejemplos en Internet, en algún momento van a toparse con:

```
set.seed(123)
ejemplo <- rnorm(n = 10000, mean = 0, sd = 1)
```

Que es **set.seed**?

set.seed genera secuencias de numeros "random" pero al poner una "semilla" nos aseguramos de que nos genere la misma secuencia en todas las computadoras.

Que es **rnorm**?

rnorm sirve para generar muestras aleatorias a partir de una población teórica con distribución normal, dándole media y desviación estándar.

Cuando hagan preguntas en internet, es muy útil usarlo!

2.2. Chocolate y felicidad

- Supongamos que podemos medir felicidad de manera cuantitativa, como una variable continua.
- Supongamos, además, que nuestro laboratorio quiere investigar cómo impactan distintas dosis de **chocolate** a la **felicidad** de los humanos.



2.2. Chocolate y felicidad

Para esto, tomamos una muestra de **100 voluntarios** y los asignamos de manera aleatoria a **5 dosis de chocolate (20, 40, 60, 80, y 100 gramos)**. Los individuos consumen la dosis asignada, el chocolate aumenta su felicidad (según la fórmula $\text{felicidad} = \text{dosis} * 2.5 + 10$), que medimos y graficamos.

Generar participantes

```
id <- 1:100
```

Generar dosis

```
dosis <- sort(rep(seq(20,100,20), 20))
```

Generar respuesta "ideal"

```
respuesta <- dosis * 2.5 + 10
```

Construir data.frame

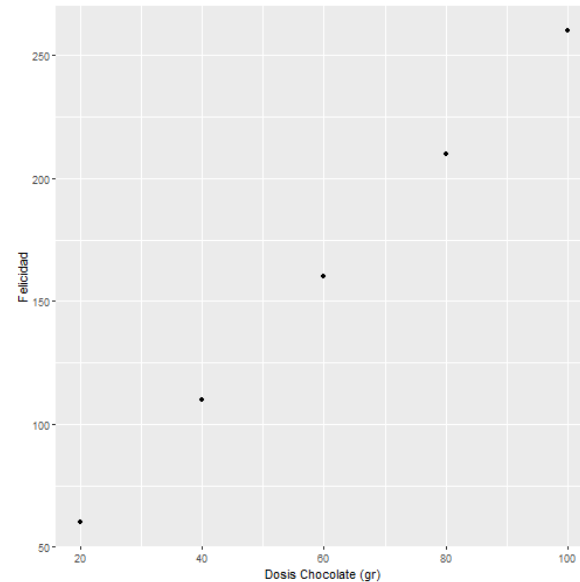
```
datos <- data.frame(id=id,dosis=dosis,respuesta=respuesta)
```

2.3. Chocolate y felicidad

Así se vería nuestro modelo **ideal**

```
p <- ggplot(datos,  
            aes(x=dosis,  
                y=respuesta))+  
  geom_point()+  
  xlab("Dosis Chocolate (gr)")  
  ylab("Felicidad")
```

p



2.4. Chocolate y felicidad

- Pero, en la realidad, esperamos variabilidad en la respuesta al chocolate entre individuos.
- Si queremos trabajar con un modelo **más realista** deberíamos tener mas variacion en la respuesta:

Semilla para muestras aleatorias.

```
set.seed(444)
```

Agregar ruido con distribución normal (media 0, sd = 5)

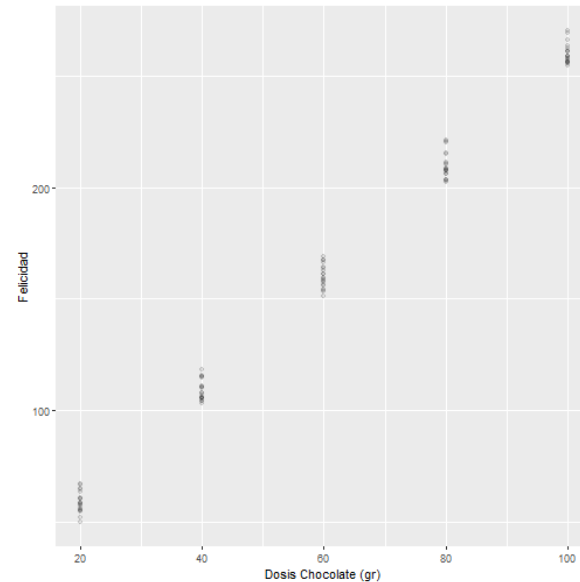
```
datos$respuesta <- datos$respuesta + rnorm(n = 100, mean = 0, sd = 5)
```

2.4. Chocolate y felicidad

- Modelo un poco **más** realista, la respuesta muestra variaciones.

```
p <- ggplot(datos,  
            aes(x=dosis,  
                y=respuesta))+  
  geom_point(alpha = 0.1)+  
  xlab("Dosis Chocolate (gr)  
  ylab("Felicidad")
```

p



nuevo concepto: alpha en el grafico crea puntos con '*transparencia*'.

2.5. Chocolate y felicidad

- ¿Cuál es el valor esperado de felicidad para una dada dosis de chocolate?
- ¿Cómo podemos estimarlo?

¿ 100 g  = 😊 ?

¿ 200 g  = 😊 ?

¿ 300 g  = 😊 ?

2.6. Modelo lineal

Para construir modelos en R, es importante el simbolo *virgulilla*

```
~
```

En nuestro caso, queremos estudiar la relación entre la felicidad (respuesta) y la dosis de chocolate (dosis). Entonces el modelo se construiría de la siguiente manera.

```
modelo_chocolate <- lm(data=datos,  
                        respuesta ~ dosis)
```

2.6. summary

Ver resultados del modelo.

```
summary(modelo_chocolate)
```

```
##  
## Call:  
## lm(formula = respuesta ~ dosis, data = datos)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -9.204  -3.696  -1.330   3.091  11.497   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  8.59279    1.15698   7.427 4.15e-11 ***   
## dosis        2.51659    0.01744 144.283 < 2e-16 ***   
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 4.933 on 98 degrees of freedom  
## Multiple R-squared:  0.9953,    Adjusted R-squared:  0.9953   
## F-statistic: 2.082e+04 on 1 and 98 DF,  p-value: < 2.2e-16
```

2.7. Broom

El paquete broom (de la paquetería tidyverse), nos permite extraer información estadística de los modelos.

Aquí está la tabla con los estimadores:

```
broom::tidy(modelo_chocolate)
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>         <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    8.59      1.16      7.43 4.15e- 11
## 2 dosis         2.52      0.0174   144. 5.93e-116
```

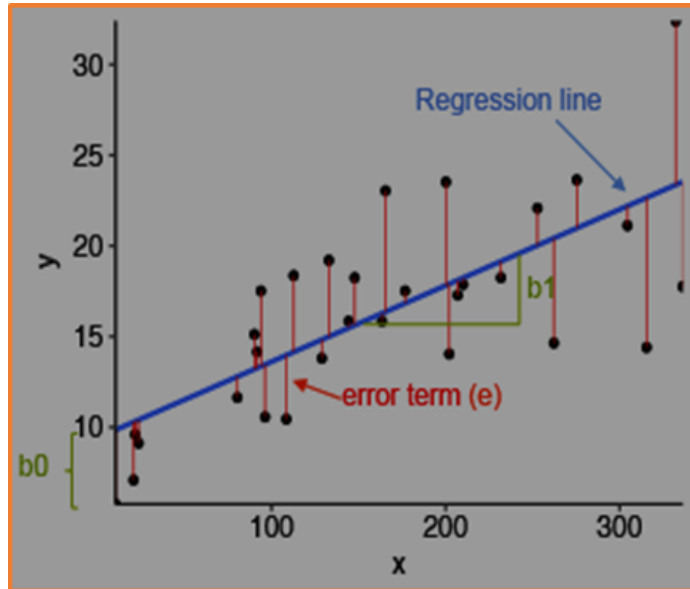
A partir de la columna de estimadores (estimate), vemos que el consumo de chocolate incrementa la felicidad (esperamos mayor un incremento en ~2.52 unidades de felicidad por cada gramo de chocolate).

Nuestro modelo puede escribirse como:

felicidad=2.52*dosis de chocolate+8.59

2.8. Recordatorio

$$\text{felicidad} = 2.52 \cdot \text{dosis de chocolate} + 8.59$$



Estimated (or predicted) y value

Estimate of the regression intercept

Estimate of the regression slope

Independent variable

Error term

$$y_i = b_0 + b_1 x + e$$

2.9. Coeficientes

También podemos acceder a porciones del modelo por separado.

Coeficientes.

```
modelo_chocolate$coefficients
```

```
## (Intercept)      dosis  
##      8.592791      2.516593
```


2.9. Intervalos

Intervalos.

```
round(confint(modelo_chocolate), 3)
```

```
##           2.5 % 97.5 %  
## (Intercept) 6.297 10.889  
## dosis       2.482  2.551
```

Valores predichos.

```
head(modelo_chocolate$fitted.values, 5)
```

```
##           1           2           3           4           5  
## 58.92464 58.92464 58.92464 58.92464 58.92464
```

Residuales.

```
head(modelo_chocolate$residuals, 5)
```

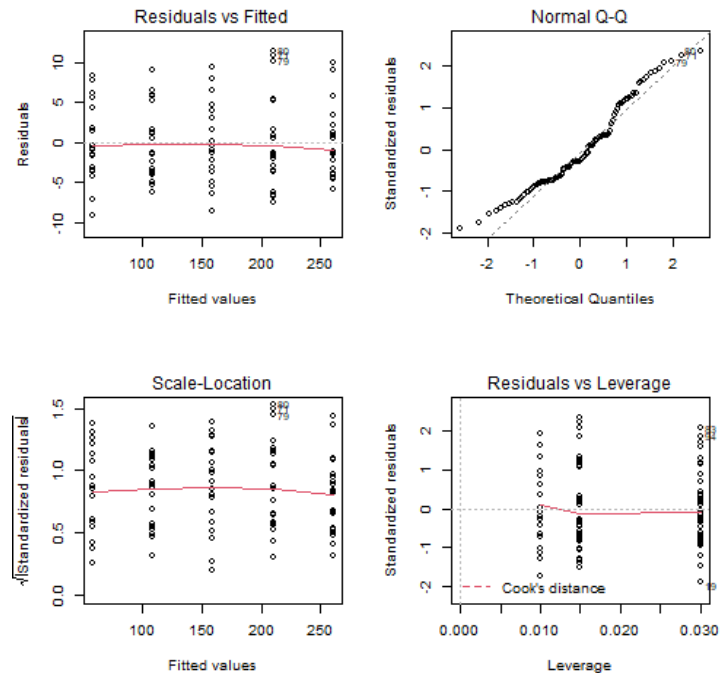
```
##           1           2           3           4           5  
## -3.7340820 -0.3253313 -7.2040228  1.8230764  6.2374206
```

2.10. Supuestos

Podemos explorar el ajuste y analizar el cumplimiento de supuestos en R utilizando la función `plot` que maneja bien objetos `lm`.

```
par(mfrow = c(2, 2))  
plot(modelo_chocolate)
```

nuevo concepto `par(mfrow)`, es que nos muestre los graficos en dos columnas y dos filas. *par* por grafical parameters, *mf* de Multiple Figures/Frames y *row* de ordenado por fila.



2.10. Supuestos

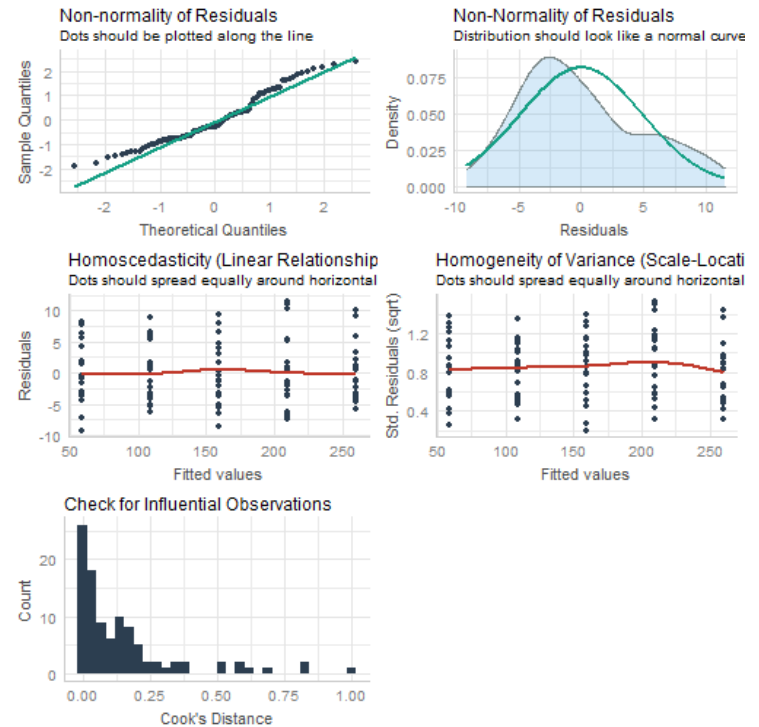
Cargar e instalar paquetes.

```
#install.packages("performance")  
#install.packages('see')  
library(performance)
```

Algunas funciones del paquete:

- `check_collinearity()`
- `check_normality()`
- `check_heteroscedasticity()`
- `check_model()`

```
check_model(modelo_chocolate)
```



2.11. Nuevo modelo

Cambiamos nuestros datos para un peor ajuste.

```
datos$nueva_dosis <- datos$dosis + rnorm(100,10,10)
```

Creemos un nuevo modelo.

```
nuevo_modelo <- lm(data = datos,  
                   respuesta~nueva_dosis)
```

Agreguemos los valores predichos y los residuales a nuestro data frame.

```
datos$nuevo_pred <- nuevo_modelo$fitted.values  
datos$residuos <- nuevo_modelo$residuals
```

2.11. Nuevo modelo

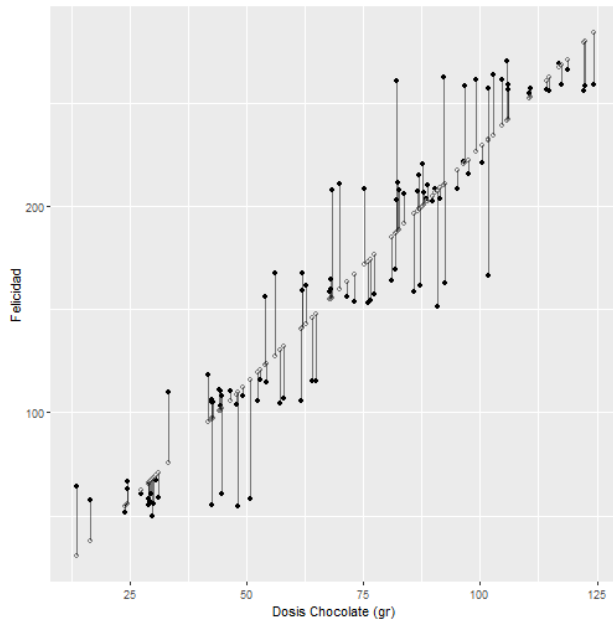
Estos son nuestros nuevos datos, y la linea de regression.

```
Plot_nueva_dosis<- ggplot(datos, aes(nueva_dosis, respuesta))+  
  geom_point()+  
  geom_point(aes(nueva_dosis, nuevo_pred), color="gray50", pch=1) +  
  theme(plot.background = element_rect(colour = NA))+  
  xlab("Dosis Chocolate (gr)") +  
  ylab("Felicidad")  
Plot_nueva_dosis
```

2.11. Residuales

Agregar los residuales.

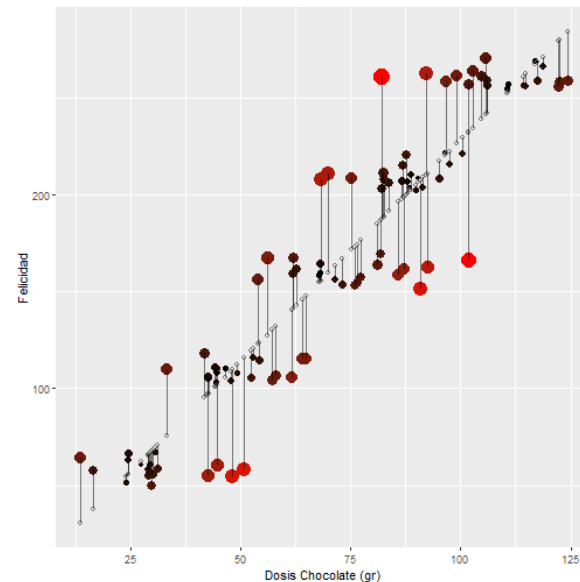
```
Plot_nueva_dosis +  
  geom_segment(aes(xend = nueva_dosis,  
                  yend = nuevo_pred),  
              alpha=0.5)
```



2.11. Residuales

Una herramienta para visualizar mejor los puntos con residuos grandes es graficarlos utilizando una escala de color y tamaño.

```
ggplot(datos, aes(nueva_dosis, re
  geom_point(aes(color = residuos
  geom_point(aes(nueva_dosis, nue
  geom_segment(aes(xend = nueva_d
    alpha=0.5))+
  xlab("Dosis Chocolate (gr)") +
  ylab("Felicidad") +
  scale_color_gradientn(colours =
  guides(color = FALSE,
    size = FALSE)
```



2.12. ¿Por qué hacer una regresión?

Los objetivos de realizar un análisis de regresión pueden resumirse en:

- Describir la relación funcional entre X e Y
- Determinar cuánta de la variación en Y puede ser explicada por la variación de X y cuánto permanece sin explicar.
- Estimar los parámetros del modelo.
- Hacer inferencia sobre los parámetros del modelo (mediante pruebas de hipótesis y cálculo de intervalos de confianza).

Ejercicios

- Generar datos, usando `set.seed` y `rnorm`
- Crear una figura con estos datos
- Crear un modelo lineal
- Extraer estimadores del modelo
- Crear un segundo modelo
- Graficar los residuales

2.13. Ejercicios

Generar nuestros datos.

```
id <- 1:100
dosis <- sort(rep(seq(20,100,20), 20))
respuesta <- dosis * 2.5 + 10
datos <- data.frame(id=id,
                    dosis=dosis,
                    respuesta=respuesta)

set.seed(444)
datos$respuesta <- datos$respuesta + rnorm(n = 100, mean = 0, sd = 5)
```

Crear figura.

```
p <- ggplot(datos,
            aes(x=dosis, y=respuesta))+
  geom_point(alpha = 0.1)+
  geom_smooth(method="lm")+
  xlab("Dosis Chocolate (gr)")+
  ylab("Felicidad")

p
```

2.14. Ejercicios

Sintaxis de modelos lineares.

```
modelo_chocolate <- lm(data=datos,  
                        respuesta ~ dosis)
```

Obtener estimadores.

```
summary(modelo_chocolate)  
broom::tidy(modelo_chocolate)
```

Ver coeficientes.

```
modelo_chocolate$coefficients
```

Checar supuestos.

```
#install.packages("performance")  
#install.packages('see')  
library(performance)
```

```
check_model(modelo_chocolate)
```

2.14. Ejercicios

Ver los residuales. Cambiemos nuestros datos para un peor ajuste.

```
datos$nueva_dosis <- datos$dosis + rnorm(100,10,10)
```

```
nuevo_modelo <- lm(data = datos,  
                  respuesta~nueva_dosis)
```

```
datos$nuevo_pred <- nuevo_modelo$fitted.values  
datos$residuos <- nuevo_modelo$residuals
```

```
ggplot(datos, aes(nueva_dosis, respuesta))+  
  geom_point(aes(color = residuos, size=abs(residuos)))+  
  geom_point(aes(nueva_dosis, nuevo_pred), color="gray50", pch=1) +  
  geom_segment(aes(xend = nueva_dosis, yend = nuevo_pred),  
              alpha=0.5)+  
  xlab("Dosis Chocolate (gr)")+  
  ylab("Felicidad")+  
  scale_color_gradientn(colours = c("red", "black", "red"))+  
  guides(color = FALSE,  
         size = FALSE)
```

Resapitulando

Esta clase:

- Explorar datos.
- Modelo lineal

Siguiente clase:

- Objetos clase factor.
- Analisis de varianza.

Contacto

Para dudas, comentarios y sugerencias:
Escríbeme a miriamjlerma@gmail.com

Este material esta accesible y se encuentra en
mi [github](#) y mi [página](#)

