Curso “Modelos de regresión y análisis bayesiano” – Mayo 2025

Guía de actividades

Unidad 1: Generalidades de R y modelo lineal

Actividad 1: Rudimentos de R y visualización de datos

Vamos a ejecutar un script sencillo de R para abrir y visualizar datos y aprovechar para repasar algunos elementos que usaremos mucho en R. Los datos son de patrones de fuego en Patagonia (ver descripción “datos\barbera_metadata.txt”). Brevemente, muestran para cada año el área (en hectáreas) quemada junto con algunas variables climáticas.

1.1 Abrir RStudio. Conviene abrir el proyecto ya creado para el curso (según lo explicado en el documento “Instalación de programas y paquetes”). Se puede abrir ejecutando el archivo .Rproj que se encuentra en la carpeta del proyecto (por ej., “R_curso_modelos”).

1.2 En RStudio, abrir el script “TP1A_Visualizacion datos.R”.

1.3 Ir leyendo y ejecutando línea por línea el script. Se puede ejecutar cada línea posicionando el cursor sobre la línea a ejecutar y presionando CTRL + Enter en el teclado.

1.4 Se puede verificar qué hizo cada línea llamando en la consola el objeto involucrado en la línea o consultando la ayuda de la función utilizada.

Actividad 2: Modelo lineal simple

Vamos a especificar un modelo lineal, simular datos según ese modelo, graficarlos y compararlos con datos reales (patrones de fuego en Patagonia).

2.1 Abrir el script “TP1B_Modelo lineal.R”.

2.2 Ejecutarlo línea por línea. El script termina con un gráfico con los valores simulados y los datos de área quemada por año en función de la temperatura.

2.3 Modificar el script para que el modelo lineal se acomode lo mejor posible a los datos.

> **Ayuda:** editar el valor de la ordenada al origen ('a', en línea 18) y la pendiente ('b', en línea 19) y volver a ejecutar las líneas 23 a 29 y 44 a 48.

Unidad 2: Determinismo y aleatoriedad

Actividad 1: Explorar distribuciones de probabilidad

Existen muchas distribuciones de probabilidad. Para familiarizarnos con algunas de las más usadas, vamos a explorarlas en una página web, jugando con sus parámetros y evaluándolas gráficamente.

1.1 Abrir esta página web:

https://www.acsu.buffalo.edu/~adamcunn/probability/normal.html

Ahí encontramos tres partes. A la izquierda, podemos elegir la distribución y vemos su descripción y fórmula. En el centro, podemos elegir visualizar la función de densidad de probabilidad (pdf), la función de probabilidad acumulada (cdf) y simular datos (botón "Simulate"). A la derecha, podemos elegir visualizar alguna de las distribuciones emparentadas con la elegida.

1.2 Empezar con la distribución normal, jugar con diferentes valores de media y varianza y visualizar la pdf y la cdf. Luego se puede ver la simulación.

1.3 Repetir esta exploración con otras distribuciones: binomial, Bernoulli, Poisson, beta, gama, uniforme y cualquier otra que llame la atención.

Actividad 2: Distribución binomial

Para ganar un poco de intuición sobre cómo surge una distribución de probabilidad, vamos a simular datos que nos den una distribución binomial.

2.1 Abrir el script "TP2A_Distribucion binomial.R".

2.2 Ejecutarlo línea por línea, verificando qué hace el script en cada paso, de forma de seguir cómo se generan los datos.

2.3 Comparar los resultados (el histograma) con lo que muestra la página usada en la Actividad 1 (https://www.acsu.buffalo.edu/~adamcunn/probability/normal.html).

2.4 Modificar el código para obtener otro valor medio para el total de la distribución.

> **Ayuda:** editar la cantidad de ceros y unos en el vector 'elementos2' definido en la línea 24 y volver a ejecutar desde la línea 24 hasta el final.

Actividad 3: Distribución normal, pdf y cdf

La función de densidad de probabilidad (pdf) y la función de distribución acumulada (cdf) son dos formas de describir cómo se comporta una distribución de probabilidad. Para entender qué representan estas funciones, vamos a usar de ejemplo la distribución normal, que nos es tan familiar. Pero es importante saber que la lógica de la pdf y la cdf se aplica a cualquier distribución de probabilidad, no solo a la normal.

3.1 Abrir el script "TP2B_Distribucion normal.R".

3.2 Ejecutarlo línea por línea la primera parte del script (hasta la línea 53). Prestar especial atención al uso de las funciones de R 'rnorm' (línea 28), 'dnorm' (línea 35), 'pnorm' (línea 36) y 'qnorm' (línea 37) y al gráfico que muestra sus resultados (líneas 48-52).

3.3 Leer la ayuda de R para estas funciones y fijarse qué argumentos tiene cada una. Como estas cuatro funciones son "parientes" entre sí, aparecen todas en la misma documentación de la ayuda. Ejecutar en la consulta '?dnorm'.

3.4 Continuar ejecutando el resto del script (de la línea 54 en adelante). Acá se utiliza el dato de precipitación por año presente en los datos de distribución de fuegos en la Patagonia. Se compara la distribución de valores de la precipitación con una distribución normal con igual media y varianza que los datos.

Actividad 4: Modelo lineal con error

Vamos a agregar estocasticidad (llamada "error" muchas veces) al resultado de un modelo lineal simple.

4.1 Abrir el script "TP2C_Modelo lineal con error.R".

4.2 Ejecutarlo línea por línea. Debería obtenerse un gráfico al finalizar el script.

4.3 Volver a ejecutar todo el script. Comparar el nuevo gráfico con el anterior. ¿Se ve alguna variabilidad?

4.4 Abrir el script "TP2D_Modelo lineal con error parte 2.R". Este script simula 'y' muchas veces para los mismos valores de 'x'. Esto permite calcular una distribución de valores de 'y' para cada valor de 'x'. Para eso repite el cálculo de 'y' hecho en el script previo ("TP2C_Modelo lineal con error.R").

4.5 Ejecutar el script línea por línea. Observar con atención cómo se genera el intervalo de incertidumbre, calculando los cuantiles (líneas 47-53).

4.6 Volver a ejecutar este script ("TP2D") pero definiendo 'x' según la línea 17 (está comentada, así que hay que sacarle el '#' para que se ejecute.

4.7 Observar la distribución univariada de 'y_obs' (últimas líneas del script).

Actividad 5: Modelo lineal con parámetros variables

Además de la estocasticidad en el resultado del modelo lineal simple, vamos a agregar estocasticidad en el valor de cada parámetro del modelo (ordenada al origen y pendiente).

5.1 Abrir el script "TP2E_Parametros variables.R".

5.2 Ejecutarlo línea por línea. Identificar dónde se introduce estocasticidad (hay más de un lugar).

5.3 Abrir el script "TP2F_Parametros variables parte 2.R". Este script repite el cálculo de 'y' para así obtener una distribución de valores y representa esta distribución graficando el rango intercuantil del 90%.

5.4 Ejecutarlo línea por línea. La lógica es similar a la del script "TP2D_Modelo lineal con error parte 2.R". Prestar atención a cómo se repite varias veces la simulación usando un loop 'for' (líneas 33-48).

5.5 ¿A qué corresponden el color naranja y el color azul en los gráficos resultantes?

5.6 Modificar los valores de los parámetros utilizados y volver a simular y graficar. ¿Qué efecto tiene en el resultado cada parámetro?

Unidad 3: Modelos no lineales

Actividad 1: Formular modelo no lineal

Cada grupo recibirá un gráfico con datos reales.

1.1 Formular en papel un modelo (de y en función de x) para los datos en el gráfico.

1.2 Especificar los componentes estocástico y determinístico. Se puede usar cualquier función matemática y distribución de probabilidades que parezca razonable.

1.3 Especificar para cada parámetro del modelo una distribución de probabilidad verosímil (al menos, verosímil en cuanto a rango y forma).

Actividad 2: Modelo de Poisson

Para ganar un poco de intuición sobre cómo surge una distribución de probabilidad, vamos a simular datos que nos den una distribución binomial.

2.1 Abrir el script "TP3A_Modelo Poisson.R".

2.2 Ejecutar línea por línea. Prestar especial atención a cómo está implementado el modelo de Poisson.

2.3 Modificar los valores de los parámetros utilizados y volver a simular y graficar. ¿Qué efecto tiene en el resultado cada parámetro?

Actividad 3: Modelo logístico

El modelo logístico es una forma habitual de representar el comportamiento de una variable de respuesta dicotómica. Vamos a ver cómo implementarlo y simular datos.

3.1 Abrir el script "TP3B_Modelo logistico.R".

3.2 Ejecutar línea por línea. Prestar especial atención a cómo está implementado el modelo logístico.

3.3 Modificar los valores de los parámetros utilizados y volver a simular y graficar. ¿Qué efecto tiene en el resultado cada parámetro?

3.4 El modelo logístico se puede reescribir de forma que los parámetros estén más intuitivamente relacionados con el resultado. Probar reescribir el modelo (y el código) reemplazando la parte lineal $a + bx$ por su equivalente $b(c + x)$. Simular y graficar.

3.5 Probar nuevamente modificando los valores de estos parámetros. ¿Qué efecto tienen en la forma y ubicación de la curva resultante?

Actividad 4: Modelo logístico para variable dependiente continua

El modelo logístico puede usarse no solo para una variable de respuesta dicotómica, sino también para una variable continua con vaya entre 0 y 1.

4.1 Formular en papel un modelo logístico para una variable continua que pueda tomar valores solamente entre 0 y 1.

Respuesta: el modelo luce así:

$$y_i \sim \text{Beta}(\alpha_i, \beta_i)$$

$$\alpha_i = \mu_i \kappa$$

$$\beta_i = (1 - \mu_i)\kappa$$

$$\mu_i = \frac{1}{1 + e^{-(a + bx_i)}}$$

4.2 Implementar ese modelo en R. Se puede partir del script "TP3B_Modelo logistico.R" y modificar lo que haga falta. Alternativamente, en el script "TP3C_Modelo logístico Beta.R" ya está implementado.

Unidad 4: Modelos jerárquicos

Actividad 1: Formular modelo jerárquico

Cada grupo recibirá unos datos graficados. También se incluye una descripción de los datos y una posible pregunta para abordar con un modelo.

1.1 Teniendo en cuenta la descripción de los datos y la pregunta, formular en papel un modelo de y en función de x para los datos en el gráfico.

1.2 Especificar los componentes estocástico y determinístico. Se puede usar cualquier función matemática y distribución de probabilidades que parezca razonable.

1.3 Especificar para cada parámetro del modelo una distribución de probabilidad verosímil (al menos, verosímil en cuanto a rango y forma).

Actividad 2: Modelo para varios grupos (no pooling)

Los modelos que contemplan para un parámetro un valor diferente según el grupo (individuo, parcela, etc.) pueden escribirse de manera que el valor de un grupo sea independiente de los demás grupos. Esto se lo suele llamar "no pooling" (es decir, sin agrupamiento). Vamos a implementar este tipo de modelo para luego (actividad 3) compararlo con un jerárquico (partial pooling) para ese mismo parámetro.

2.1 Abrir el script "TP4A_Modelo no pooling.R".

2.2 Ejecutar línea por línea. Prestar atención a cómo se simulan los datos (hay un loop 'for' anidado en otro loop 'for') y cómo/dónde se sortean los valores para los parámetros.

Actividad 3: Modelo jerárquico (partial pooling)

La implementación de un modelo jerárquico tiene algunas particularidades que vamos a abordar simulando datos según un modelo jerárquico.

3.1 Abrir el script "TP4B_Modelo jerarquico.R".

3.2 Ejecutar línea por línea. Prestar atención a cómo se simulan los datos (hay un loop 'for' anidado en otro loop 'for').

3.3 ¿A qué corresponde cada panel del gráfico resultante?

3.4 Implementar el modelo especificado en la Actividad 1 de esta unidad, simular datos y graficarlos. Se puede partir del script "TP4B_Modelo jerarquico.R" y modificar la necesario.

Actividad 4: Modelo con detección imperfecta

Vamos a implementar modelos con detección imperfecta y simular datos.

4.1 Abrir el script "TP4C_Modelo detecc imperfecta.R".

4.2 Ejecutar línea por línea hasta el final (genera un gráfico).

4.3 ¿Qué representa cada color en el gráfico?

4.4 Escribir en papel (con notación matemática) el modelo implementado en el script.

4.5 Implementar el siguiente modelo jerárquico que representa la detección de caracoles:

$$y_i \sim \text{Bernoulli}(p_i z_i)$$

$$\text{logit}(p_i) = a + bT_i$$

$$z_i \sim \text{Bernoulli}(q_i)$$

$$\text{logit}(q_i) = c + dx_i$$

donde y_i es la detección o no de caracoles (variable binaria) en el sitio i, z_i es la presencia o no de caracoles (variable binaria, latente) en el sitio i, p_i es la probabilidad de detectar caracoles en el sitio i dado que estén presentes, q_i es la probabilidad de presencia de caracoles en el sitio i, T_i es la temperatura en el sitio i, x_i es alguna covariable (profundidad o pH del agua) que afecta la presencia de caracoles en el sitio i, mientras que a, b, c y d son los parámetros de las rectas.
Considerar especificar las distribuciones necesarias (y compatibles) para simular datos.

4.6 Simular datos y graficar de forma análoga a lo graficado en el script "TP4B_Modelo detecc imperfecta.R".

Unidad 5: Estimación de parámetros e inferencia bayesiana

El objetivo de estas actividades es generar una intuición gráfica sobre cómo se estiman los parámetros de los modelos y cómo influyen la cantidad de datos y las distribuciones previas (en el caso bayesiano).
Se pueden seguir los scripts "TP5A_Estimacion y bayes en 1D.R" (en una dimensión) y "TP5B_Estimacion y bayes en 2D.R" (en dos dimensiones), que contienen el ejemplo mostrado en clase. En una dimensión el código es más sencillo y más fácil de seguir. Alternativamente se puede escribir otro script reciclando ese y utilizando otros datos. En caso de ponerse creativo, conviene limitarse a modelos que tengan uno o dos parámetros. Si tienen más, habrá que fingir que los conocemos, fijándolos en algún valor

verosímil. Esto es necesario para poder generar una intuición gráfica (muy difícil si tenemos más de dos parámetros).

Actividad 1: Funciones de costo y verosimilitud

1.1 Elegir un set de datos y definir un modelo sencillo (uno (1D) o dos (2D) parámetros). Puede ser un modelo estadístico o no; es decir, la función de costo no necesariamente tiene que ser una -log verosimilitud.

1.2 Evaluar qué restricciones deberían imponerse sobre los parámetros para que el modelo sea sensato para los datos y las idea en cuestión.

1.3 Graficar la función de verosimilitud del modelo (o la función de costo) y encontrar el valor máximo utilizando el método de la grilla. Si el modelo tiene D > 2 parámetros, se deberán fijar D - 2 parámetros para poder graficar. Graficar la predicción en función de alguna variable predictora (si es que la hay) utilizando el valor de la grilla con mayor verosimilitud o menor costo.

1.4 Comparar la estimación basada en la grilla con la obtenida usando optim.

1.5 ¿Existen otros valores de los parámetros que también presentan un ajuste razonable a los datos, aunque sea subóptimo? ¿Cómo se podría escoger un set de valores razonables?

1.6 Repetir los puntos 1.3 y 1.5 pero utilizando el 20% y el 50% de los datos disponibles y comparar los resultados.

Actividad 2: Verosimilitud, distribución previa y distribución posterior

2.1 Utilizando el mismo set de datos y modelo escogido previamente (punto 1.1), definir distribuciones previas para los parámetros, convirtiendo al modelo en uno bayesiano.

2.2 Graficar la función de verosimilitud, la densidad previa y la densidad posterior.

2.3 Explorar distintas previas, ya sea variando los parámetros de una misma distribución o utilizando otra distribución. Intentar crear previas que influyan en el resultado y otras que no.

2.4 Repetir la comparación entre posteriores, pero utilizando únicamente dos o tres observaciones. ¿Siguen sin influir las previas que no influían en el resultado usando todos los datos?

2.5 ¿Cómo se podría caracterizar la distribución posterior de una forma más compacta que un gráfico? ¿De qué manera se podrían describir resultados sobre un parámetro en particular si el modelo tiene más de un parámetro?

Unidad 6: Inferencia bayesiana en la práctica: MCMC

Actividad 1: Ajustar un modelo con Stan

Ajustaremos modelos bajo un enfoque bayesiano muestreando la distribución posterior con Stan. Igual que en las actividades de la unidad 6, se pueden seguir scripts ya elaborados o reciclarlos para estimar modelos de confección propia.
En la carpeta "modelos" hay scripts de R y Stan implementando distintos modelos. En la misma carpeta, el archivo "modelos_descripcion.txt" tiene una descripción de los modelos disponibles. En algunos casos, un script de R implementa varios modelos, por lo que hay más de un archivo .stan para su correspondiente archivo .R.
Estos scripts de modelos ofician de ejemplo del contenido de las unidades 6 y 7, por lo que en estas actividades no es necesario ejecutarlos completamente, sino que ahora ejecutaremos el código previo a la verificación predictiva posterior. Esta sección y las siguientes corresponde a la unidad 7.
Se recomienda elegir modelos de complejidad acorde a la habilidad con R/Stan, además del interés personal (ver archivo "modelos_descripcion.txt").

1.1 En caso de optar por construir un modelo nuevo, escribirlo en papel/txt y luego en Stan (guardarlo en un archivo .stan). En caso de seguir uno ya desarrollado, esforzarse por entender el código de Stan, identificar qué hace cada parte. Acá, como en todo momento, vale preguntar a los docentes, compañeros, y/o a cualquier inteligencia artificial.

1.2 Compilar el modelo y muestrearlo (desde R, usando cmdstanr). Procurar que el Rhat sea < 1.01 y el Número efectivo de muestras (*bulk* y *tail)* sea >= 1000 para todos los parámetros. Si los diagnósticos sugieren algún otro problema, leer sobre ello o preguntar (ver https://mc-stan.org/learn-stan/diagnostics-warnings.html).

1.3 Una vez obtenida una muestra confiable de la posterior (diagnósticos OK), obtener un resumen de las distribuciones marginales de los parámetros y también graficarlas. Realizar enunciados probabilísticos de interés sobre los parámetros.

1.4 Comparar las distribuciones posteriores marginales de los parámetros con sus previas. ¿Aprendimos algo al ver los datos?

1.5 Repetir (3) pero sobre cantidades derivadas de los parámetros (ejemplo: lambda en una regresión poisson, o la distribución de la respuesta si fijamos la predictora en algún valor).

1.6 Si lo anterior fluyó como un río, explorar el efecto de modificar las previas. Puede probar previas muy amplias y muy estrechas. ¿Cambia mucho el resultado?

Unidad 7: Verificación e interpretación de modelos

Practicaremos cómo llevar a cabo la verificación de un modelo (es decir, evaluar si representa los datos de forma adecuada) y su interpretación. Usaremos de guía/molde los modelos que se encuentran en la carpeta "modelos", pero nuevamente, también se pueden escribir nuevos. Si verificar e interpretar modelos resulta sencillo, se recomienda explorar la sección de simulaciones previas de la clase 7 e implementarlas en algún modelo de interés.

Actividad 1: Verificación

1.1 Simular datos de la distribución predictiva posterior utilizando los valores de las predictoras observados. Utilizar esta distribución para evaluar el ajuste del modelo a los datos, basándonos en el paquete DHARMa pero también explorando gráficos del paquete bayesplot. Graficar también los residuos en función de las predictoras.

1.2 ¿Cuán mal ajusta el modelo? ¿De qué manera ajusta mal? ¿Cómo podría mejorarse?

Actividad 2: Interpretación

2.1 Realizar gráficos sencillos para entender las implicancias del modelo. Por ejemplo, predicciones de la media en función de una predictora.

2.2 Pensar en métricas o gráficos de interés menos ortodoxos y elaborarlos o calcularlas (por ejemplo, la derivada parcial de la media con respecto a una predictora). Que una métrica o gráfico resulte de interés depende fuertemente del contexto de aplicación, así que pensar en un problema propio será de ayuda. De ser posible, aplicar esas ideas en el modelo elegido, aunque en el problema en cuestión no resulte de interés. El objetivo es reconocer que podemos obtener una distribución posterior de cualquier métrica que se pueda calcular a partir del modelo (cantidades derivadas) y practicar cómo hacerlo.

Actividad 3: Simulaciones previas (opcional)

3.1 Utilizar simulaciones para definir previas de un modelo de interés o evaluar gráficamente las previas utilizadas en los modelos (antes de ajustar el modelo). Nótese que algunos scripts tienen algunas simulaciones previas.