COMPUTACIONAL ESTADÍSTICA

LABORATORIO 5

Parte 1

Se dispone de diferentes conjuntos de datos reales y simulados (como *iris*, *singer*, *barley*, *mtcars*, *volcano*, entre otros), que contienen variables numéricas, categóricas y geoespaciales. Utilizando el sistema gráfico **Lattice** en R, se requiere elaborar visualizaciones **univariadas, bivariadas, trivariadas y de múltiples variables** que permitan explorar distribuciones, relaciones y patrones.

Para cada ejercicio propuesto, implemente el código en R empleando la función gráfica correspondiente, ajuste parámetros estéticos y de presentación, y finalmente interprete los resultados obtenidos en el contexto de los datos utilizados.

Univariados (barchart, bwplot, densityplot, dotplot, histogram, qqmath, stripplot)

1. Usa histogram(~ Petal.Length, data=iris) con distintos breaks (Sturges, 10, 20) y comenta cómo cambia la percepción de la distribución.
2. Compara densidades con densityplot(~ Petal.Length | Species, data=iris); añade una densidad normal por panel con panel.mathdensity y discute el ajuste.
3. Con singer, genera bwplot(height ~ voice.part, data=singer) y reporta mediana, IQR y outliers por registro vocal.
4. Con singer, replica el 3) pero con stripplot(voice.part ~ jitter(height), jitter=TRUE); explica cuándo prefieres stripplot vs. boxplot.
5. Construye un dotplot(mpg ~ disp, groups=factor(cyl), data=mtcars, auto.key=TRUE) y analiza si hay patrón de consumo por cilindros.
6. Arma barchart(yield ~ variety | site, groups=year, data=barley, layout=c(1,6)) y explica el efecto de usar stack=TRUE frente a barras agrupadas.
7. Con histogram(~ height | voice.part, data=singer, nint=17, layout=c(2,4)), interpreta asimetrías y curtosis por panel.
8. Genera 10 muestras exponenciales y aplica qqmath(~ ejexp[,1] | factor(ejexp[,2]), distribution=qexp); identifica paneles con peor ajuste a exponencial.
9. En iris, compara densityplot(~ Sepal.Width, groups=Species, auto.key=TRUE) vs. bwplot(Sepal.Width ~ Species); discute qué revela cada uno.
10. Con barley, usa barchart(yield ~ variety | site, groups=year, scales=list(x=list(abbreviate=TRUE,minlength=5))) y justifica el abreviado de etiquetas.

Bivariados (xyplot, qq)

1. Con iris, xyplot(Petal.Width ~ Petal.Length, groups=Species, auto.key=TRUE); añade type=c("p","r") y compara pendientes por especie.
2. Repite 1) pero condiciona: xyplot(Petal.Width ~ Petal.Length | Species, groups=Species); contrasta leer por panel vs. por color.
3. Usa xyplot(mpg ~ wt, data=mtcars, groups=factor(cyl), type=c("p","smooth")) y comenta la relación consumo–peso por cilindrada.
4. Implementa panel.superpose en 3) y ajusta span=0.75; discute sobreajuste/infraajuste del suavizado.
5. Construye qq(y ~ x) para comparar cuantiles de dos mediciones simuladas (p.ej., alturas vs. pesos normalizados); evalúa linealidad y colas.
6. Con barley, grafica xyplot(yield ~ as.numeric(variety) | site, groups=year) y discute si hay "ranking" estable de variedades entre sitios.
7. Divide por meses un conjunto simulado (ventas) y usa xyplot(ventas ~ día | mes) para detectar estacionalidad o picos atípicos.
8. Con mtcars, xyplot(qsec ~ hp, groups=factor(am)); ¿la transmisión (am) cambia la relación potencia–cuarto de milla?
9. Añade bandas de confianza a una recta: panel.abline(lm(...)) + panel=... (usa panel.lmline + bandas simuladas) y comenta.
10. Genera un QQ-plot entre dos grupos de mtcars (p.ej., 4 vs. 8 cilindros) para mpg; interpreta desplazamientos y cambios de forma.

Trivariados (levelplot, contourplot, cloud, wireframe)

1. Reproduce el ejemplo de superficie: define x,y y z <- cos(r^2)*exp(-r/(pi^3)); usa levelplot(z ~ x*y, grid, cuts=50, region=TRUE) y explica patrones de nivel.
2. Cambia cuts (20, 50, 100) en levelplot y discute el compromiso entre detalle y ruido visual.
3. Traza contourplot(volcano, at=seq(..., by=10), region=TRUE) y comenta cómo la resolución de curvas afecta la lectura de pendientes.
4. Con cloud(Sepal.Length ~ Petal.Length * Petal.Width, groups=Species, data=iris), experimenta con screen=list(x=-90,y=70); explica la mejor vista para separar especies.
5. Usa wireframe(volcano, shade=TRUE, aspect=c(61/87,0.4)) y comenta el efecto de light.source en la percepción de relieves.
6. Crea una malla con expand.grid y define z <- log((x^gr + y^2)*gr); usa wireframe(z ~ x*y, groups=gr, drape=TRUE, colorkey=TRUE) y explica la codificación por color.
7. Compara levelplot vs. contourplot para la función del punto 1); ¿cuándo preferirías contornos sin relleno?
8. Simula una gaussiana 2D con outer y grafícala con levelplot y contourplot; evalúa el efecto de correlación (ρ=0, 0.5, 0.9).
9. Con cloud, añade pch y cex basados en una cuarta variable (p.ej., Sepal.Width escalada) para mostrar "cuasi-4D"; interpreta.
10. Usa wireframe para dos superficies en subplots (par.settings + layout) y contrasta topologías (p.ej., cóncava vs. ondulada).

Varias variables (splom, parallel/parallelplot)

1. Con iris, splom(~ iris[1:4], groups=Species, auto.key=list(columns=3)); identifica pares de variables con mejor separación por especie.
2. Repite 1) y añade panel=panel.superpose; discute la utilidad de ver elipses de confianza (puedes agregarlas con panel personalizado).
3. Normaliza columnas de iris[1:4] y grafica parallelplot(~ iris[,1:4] | Species); compara perfiles medios por especie.
4. Modifica el tema con trellis.par.set(canonical.theme("col.whitebg")) y evalúa legibilidad en splom.
5. Construye un splom sólo para dos especies (filtra iris) y discute si la separación mejora.
6. En parallelplot, cambia el orden de ejes (permuta columnas) y explica cómo el orden altera la interpretabilidad de cruces.
7. Crea variables altamente correlacionadas y observa su “solapamiento” en parallelplot; discute multicolinealidad visual.
8. Con mtcars, selecciona 5 variables numéricas y usa splom(~ subset, groups=factor(cyl)); ¿qué combinaciones mejor separan 4/6/8 cilindros?
9. Personaliza panel para splom agregando panel.loess en los paneles inferiores y panel.cor (correlación) en los superiores.
10. Exporta a archivo (pdf(...)/png(...)) un parallelplot con leyenda (auto.key=TRUE) y documenta parámetros mínimos para reproducibilidad (tamaño, resolución).

Parte 2

Se dispone de diferentes conjuntos de datos que contienen variables numéricas, categóricas y geoespaciales.

ds_empleados – Datos de empleados (edad, salario, departamento, años de experiencia, desempeño, etc.)
ds_ventas_retail – Datos de ventas en retail (fecha, categoría, ventas, unidades vendidas, canal, descuento, satisfacción)
ds_alumnos – Datos académicos (nombre, edad, carrera, promedio, créditos, horas de estudio, participación)

Utilizando el sistema gráfico **Lattice** en R, se requiere elaborar visualizaciones **univariadas, bivariadas, trivariadas y de múltiples variables** que permitan explorar distribuciones, relaciones y patrones.

Para cada ejercicio propuesto, implemente el código en R empleando la función gráfica correspondiente, ajuste parámetros estéticos y de presentación, y finalmente interprete los resultados obtenidos en el contexto de los datos utilizados.

Univariados

1. histogram(~ Salario, data=ds_empleados) usando diferentes valores de breaks y comentando los cambios.
2. densityplot(~ Salario | Departamento, data=ds_empleados) y añadir densidad normal en cada panel.
3. bwplot(Edad ~ Departamento, data=ds_empleados) y reportar estadísticas de posición y dispersión.
4. stripplot(Departamento ~ Edad, jitter=TRUE, data=ds_empleados) y comparar con el boxplot del punto 3.
5. dotplot(Ventas ~ Categoria, groups=Canal, data=ds_ventas_retail, auto.key=TRUE) para analizar ventas por canal.
6. barchart(Unidades ~ Categoria | Canal, data=ds_ventas_retail, layout=c(1,3)) con barras agrupadas y apiladas.
7. histogram(~ Satisfaccion | Canal, data=ds_ventas_retail, nint=12) para ver sesgo y curtosis por canal.
8. Generar 10 muestras de Promedio en ds_alumnos y aplicar qqmath() contra distribución normal.
9. Comparar densityplot(~ HorasEstudio, groups=Carrera, data=ds_alumnos) vs. bwplot(HorasEstudio ~ Carrera); discutir hallazgos.
10. barchart(Creditos ~ Carrera, data=ds_alumnos) abreviando nombres de carrera en el eje X.

Bivariados

1. xyplot(Salario ~ AñosExperiencia, groups=Departamento, data=ds_empleados, auto.key=TRUE) con línea de regresión.
2. Repetir 1) pero condicionando por Departamento.
3. xyplot(Ventas ~ Descuento, groups=Categoria, data=ds_ventas_retail, type=c("p","smooth")) y discutir tendencias.
4. Aplicar panel.superpose al punto 3 con ajuste suave (span=0.75).
5. Crear qq(Ventas ~ Unidades, data=ds_ventas_retail) para comparar cuantiles de ventas y unidades vendidas.
6. xyplot(Ventas ~ as.numeric(Categoria) | Canal, groups=Descuento, data=ds_ventas_retail) y discutir rankings por canal.
7. Simular ventas diarias y xyplot(Ventas ~ Fecha | Categoria, data=ds_ventas_retail) para ver estacionalidad.
8. xyplot(Promedio ~ HorasEstudio, groups=Carrera, data=ds_alumnos) y comentar diferencias por carrera.
9. Añadir bandas de confianza a una recta de regresión en el punto 8.
10. qq(Promedio ~ Creditos, data=ds_alumnos) para comparar distribuciones académicas.

Trivariados

1. Definir x=Edad, y=AñosExperiencia, z=Salario y graficar levelplot(z ~ x*y, data=ds_empleados).
2. Cambiar cortes (cuts) en el levelplot anterior y comentar.
3. contourplot(Salario ~ Edad*AñosExperiencia, data=ds_empleados) con relleno de regiones.
4. cloud(Salario ~ Edad*AñosExperiencia, groups=Departamento, data=ds_empleados) variando screen.
5. wireframe(Salario ~ Edad*AñosExperiencia, data=ds_empleados, shade=TRUE) con cambios en light.source.
6. En ventas: wireframe(Ventas ~ Descuento*Unidades, groups=Categoria, data=ds_ventas_retail, drape=TRUE).
7. Comparar levelplot vs. contourplot para ventas en función de unidades y descuentos.
8. Simular función gaussiana sobre HorasEstudio y Creditos en alumnos y graficar.
9. cloud(Promedio ~ HorasEstudio*Creditos, data=ds_alumnos, pch=Carrera) con tamaño proporcional a Participacion.
10. wireframe(Promedio ~ HorasEstudio*Creditos, data=ds_alumnos) en dos vistas (layout).

Varias variables

1. splom(~ ds_empleados[,c("Edad","AñosExperiencia","Salario")], groups=Departamento); identificar separaciones.
2. Añadir panel.superpose y elipses de confianza al splom anterior.
3. parallelplot(~ ds_ventas_retail[,c("Ventas","Unidades","Descuento","Satisfaccion")], groups=Categoria) y comparar perfiles.
4. Cambiar tema visual en un splom de empleados.
5. Hacer splom solo con dos departamentos y discutir mejora en separación.
6. En parallelplot, reordenar ejes y explicar cambios.
7. Introducir alta correlación simulada en ventas y ver su reflejo en parallelplot.
8. splom(~ ds_alumnos[,c("Promedio","Creditos","HorasEstudio","Participacion")], groups=Carrera).
9. Personalizar panel del splom con panel.loess y panel.cor.
10. Exportar parallelplot de alumnos a PDF con parámetros de resolución y tamaño.