Regresión lineal por mínimos cuadrados

Utilizaremos la librerir [letsPlot] para visualizar y la libreria [commons-math3] para obtener la pendiente y el término independiente de la recta

In [1]:
```
// libreria de commons-math3
@file:Repository("https://repo1.maven.org/maven2/")
@file:DependsOn("org.apache.commons:commons-math3:3.6.1")
```

In [2]:
```
// libreria de lets-plot
%use lets-plot
```

In [3]:
```
// Crear un objeto de regresión simple
import org.apache.commons.math3.stat.regression.SimpleRegression

val regression = SimpleRegression()
```

Preparamos datos

In [4]:
```
// Crear un conjunto de datos con los valores de x e y para graficar con lets-plot
val d = mapOf(
    "x" to listOf(1.0, 2.0, 3.0, 4.0, 5.0),
    "y" to listOf(2.0, 3.0, 5.0, 4.0, 6.0)
)
```

In [5]:
```
// este bucle nos permite recorrer el mapa para alimentar el objeto de regresión de math3
for (i in d["x"]?.indices ?: emptyList<Int>().indices) {
    val xValue = d["x"]?.get(i)
    val yValue = d["y"]?.get(i)

    if (xValue != null && yValue != null) {
        println("x = $xValue, y = $yValue")
        regression.addData(xValue,yValue)
    }
}
```

```
x = 1.0, y = 2.0
x = 2.0, y = 3.0
x = 3.0, y = 5.0
x = 4.0, y = 4.0
x = 5.0, y = 6.0
```

Dibujamos la regresión lineal

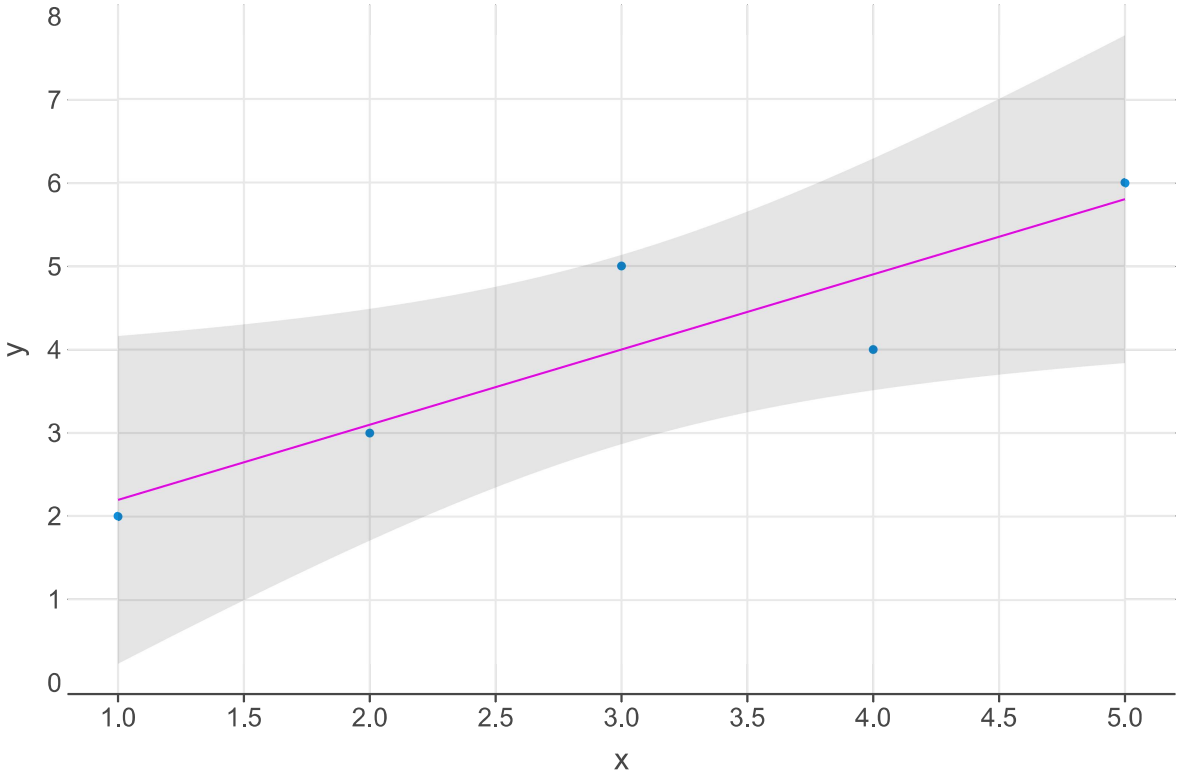
In [6]:

```
// Creamos un objeto plot, que es una representación gráfica de los datos:
//   la función recibe dos parámetros: el primero es el mapa de datos `d`,
//                                     y el segundo es una expresión lambda que indica qué variables usar
//                                     para el eje x y el eje y del gráfico.
val p = letsPlot(d) {x="x"; y="y"}

// Se usa el operador `+` para añadir dos capas geométricas al objeto plot:
//      la función `geomPoint()` dibuja un punto por cada fila de los datos,
//       y la función `geomSmooth()` dibuja una línea suavizada que muestra la tendencia de los datos.
//mpg_plot + geomPoint() + geomSmooth()

//En este caso forzamos que suavice con -> statSmooth(method = "lm",  level = 0.95)
//        El método “lm” significa regresión lineal
//        Establecemos un nivel de confianza (entre 0 y 1) del 95%
//               nota:  nivel de confianza establece un rango de valores que es probable que contenga el verdadero valor
//                      del parámetro que estamos estimando
p + geomPoint() + statSmooth(method = "lm", level = 0.95)
```

Out[6]:



Modelo matemático de la regresión lineal

In [7]:
```
// Obtener los parámetros del modelo
val slope = regression.slope
val intercept = regression.intercept

// Imprimir los resultados
println("La ecuación de la línea de regresión es: y = $slope x + $intercept")
```

```
La ecuación de la línea de regresión es: y = 0.9 x + 1.3
```

In [8]:
```
val r = regression.getR()
println("El coeficiente de correlación de Pearson es: $r")
```

```
El coeficiente de correlación de Pearson es: 0.9
```

In [ ]: