PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

INTRODUCCIÓN A LA PROGRAMACIÓN

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Minutos 25-26 (?)

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Tragación

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Temario de hoy

- Próximas clases
- Control 2
- Solemne 2
- Detección de casos sospechosos
- Repaso (recuerdo) de la materia

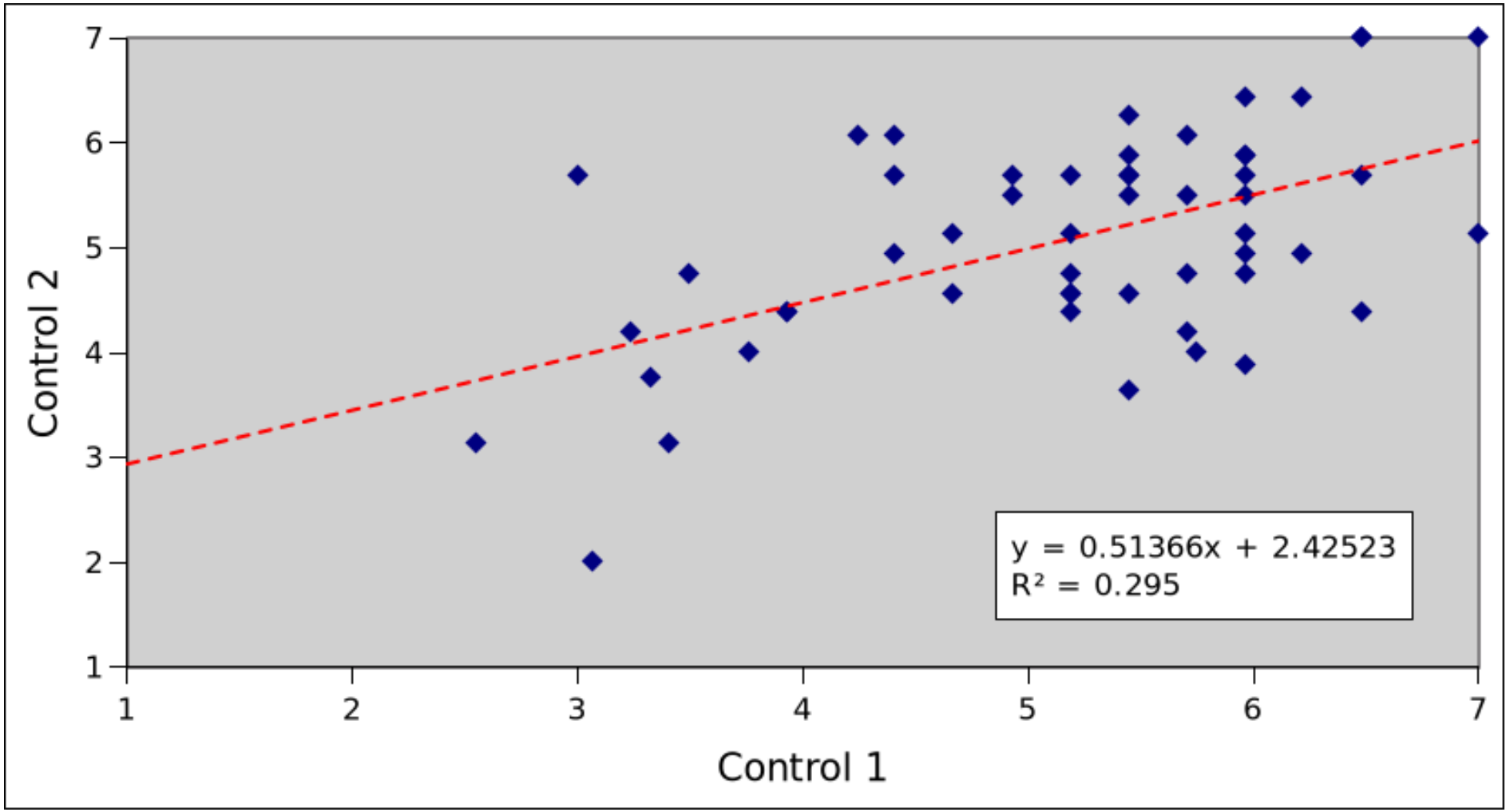
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Últimas semanas

- Semana 22-sep al 25-sep
  - Repaso (hoy), **listas** y **listas anidadas**
- Semana 29-sep al 2-oct
  - Listas anidadas
- Semana 6-oct al 9-oct
  - **SOLEMNE 2** (hasta listas anidades)
- Semana 13-oct al 16-oct
  - Repaso y cierre notas pre-examen
- Semana 20-oct al 23-oct
  - **EXAMEN**

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Control 1 vs Control 2



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Estrategia Solemne 2

- Próximas evaluaciones:
  - Solemne 2
  - Examen
    - Eximición con promedio ≥ 5.5, sólo notas azules
- Practicar programando
- Practicar escribiendo programas en papel
- Estudiar estructuras de solución típica, imitar

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Casos sospechosos

- Guías
- Desarrollo de soluciones sospechosas
- Qué podemos ver
- Relación con bajo rendimiento en Solemne 1
- Pronóstico para Solemne 2
- Última oportunidad para enmendar

Repaso/Recuerdo

- If-elif-else
- For-While
  - Bucles simples
  - Bucles anidados
- Strings
- Funciones
  - Funciones propias
  - Importar funciones

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Ejemplos prácticos

- Los siguientes ejemplos fueron implementados durante la clase
- Videos disponibles
- Agregados a las láminas *ex-post*

Ejemplo: String de Tokens

- **Problema**: implemente la función **ntoken** que, dado un string de tokens separados por punto y coma (ej. `azul;verde;rojo`) y un número entero **N**, retorne el **N-ésimo token** (contado desde cero)
  - **Problema anexo**: implemente la función **sin_token** que, dado un string de tokens separados por punto y coma y un número entero **N**, retorne el string de tokens, pero **borrando** el N-ésimo token
- Solución en video: https://youtu.be/GRmDxiYHOP4

Solución (1/3)

```
def ntoken( cadena, N ):
    # cadena: un string con tokens seps por ;
    # N: un int que indica el token a retornar
    resp = ""
    n = 0
    for caracter in cadena:
        if caracter == ";":
            n += 1
        elif n == N:
            resp += caracter
    return resp
```

Solución (2/3)

```
def sin_token( cadena, N ):
    # cadena: un string con tokens seps por ;
    # N: un int que indica el token a remover
    resp = ""
    n = 0
    for caracter in cadena:
        if caracter == ";":
            n += 1
            resp += ";"
        elif n != N:
            resp += caracter
    return resp
```

Solución (3/3)

```
# flujo principal (el "main")
print("Ingrese una cadena de tokens:")
tkn = input()
print("Ingrese un entero (0-...):")
nro = int(input())

r = ntoken(tkn,nro)
print("El token es:", r)

r = sin_token(tkn,nro)
print("Sin token es:", r)
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Ejemplo: Similitud de Parlamentarios

- **Problema**: consideremos que la conducta de votos de los parlamentarios se escribe en cadenas como `++---A+A--`, donde el signo `+` representa votar a favor, el signo `-` representa votar en contra y la letra `A` representa una ausencia o inasistencia. Implemente la función **simpoli** que reciba dos cadenas de votos de dos parlamentarios y que retorne la tasa de coincidencia entre ellos (número de veces que actuaron igual dividido por el total de votaciones).
- Solución en video: https://youtu.be/4NAGGdJ8bTo

Solución

```
def polisimil( p1, p2 ):
    identicos = 0
    largo = len(p1)
    for i in range(largo):
        if p1[i] == p2[i]:
            identicos += 1
    cuociente = identicos / largo
    return cuociente

# main (flujo principal)
print("Ingrese el actuar de votaciones del parlamentario 1:")
par1 = input()
print("Ingrese el actuar de votaciones del parlamentario 2:")
par2 = input()
s = polisimil( par1, par2 )
print("Similitud:", s)
```