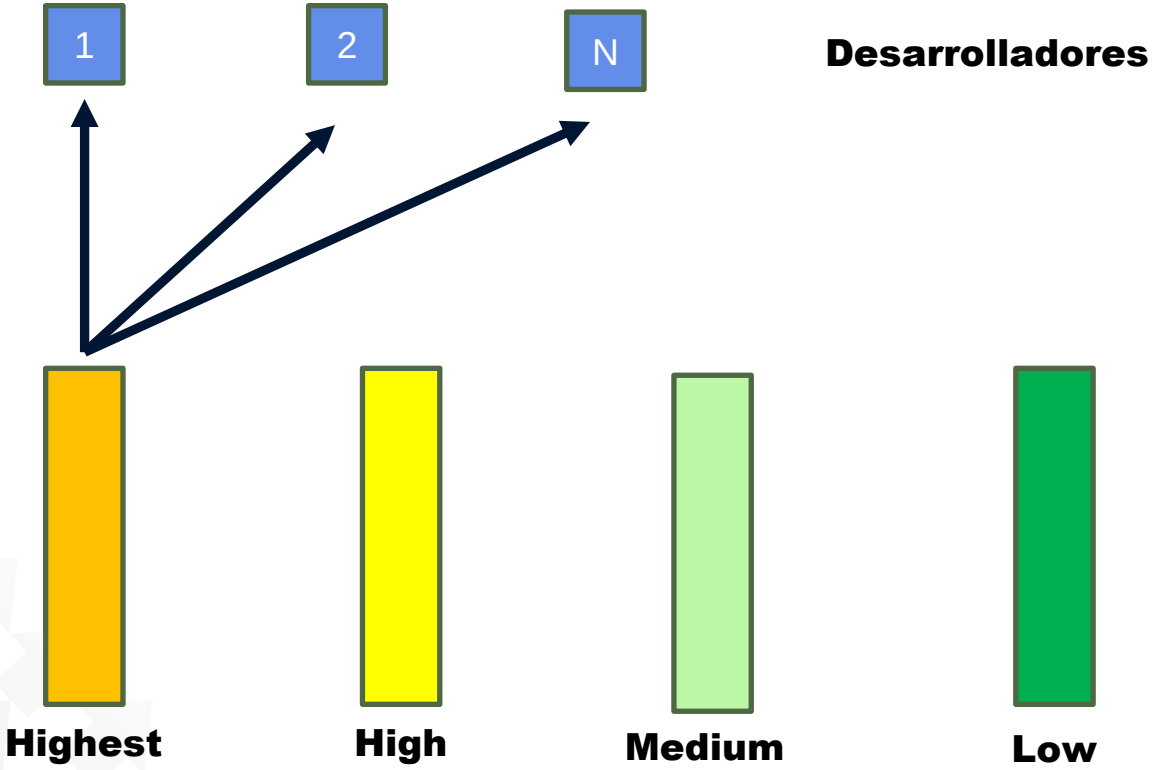
Estudio para averiguar la cantidad de **desarrolladores** y la **frecuencia** óptima de actualización de prioridades, para un proyecto con tareas de **prioridad cambiante**

Grupo 2:

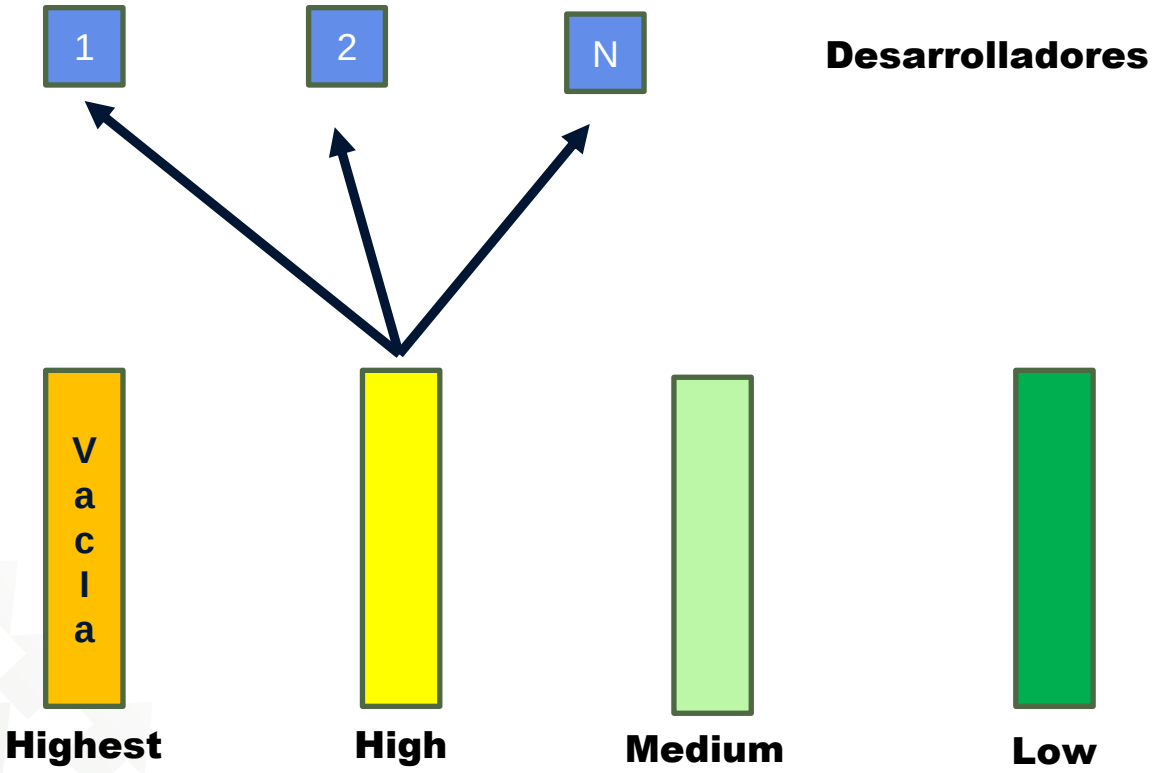
- Ornella Fasciolo
- Gabriel Spisso
- Alejandro Deheza

Índice:

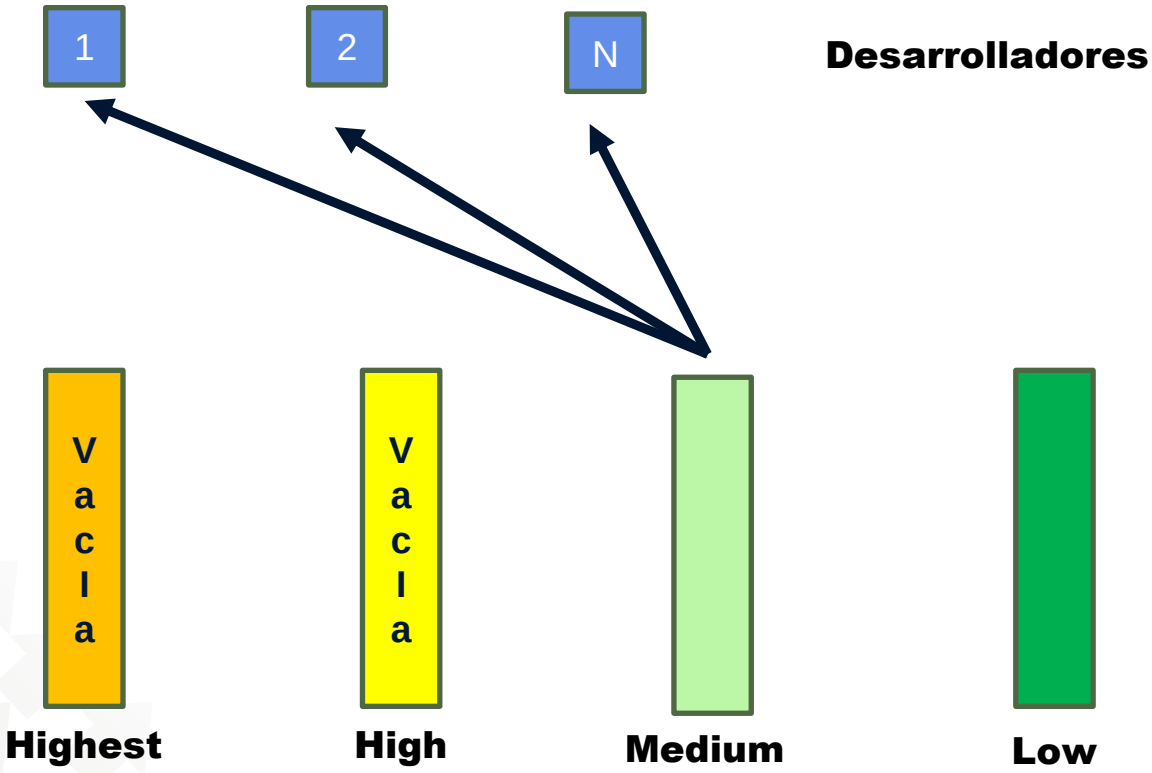
1. Problema:



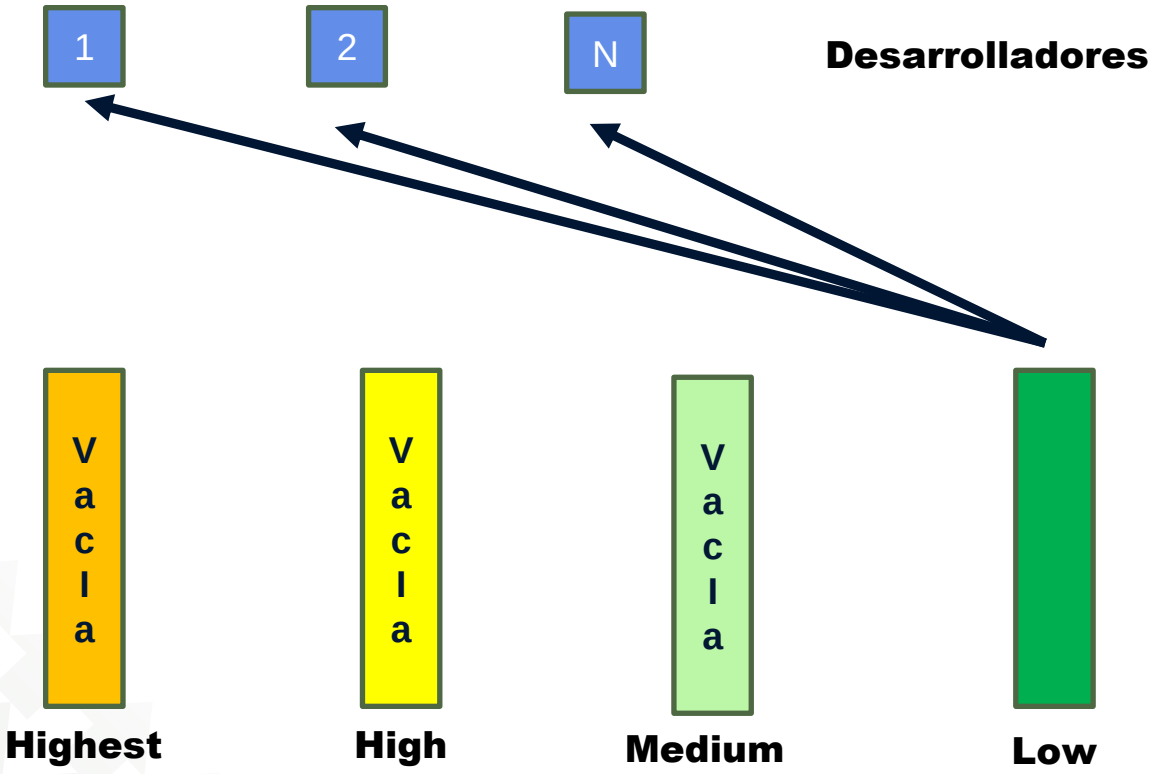
1. Problema:



1. Problema:

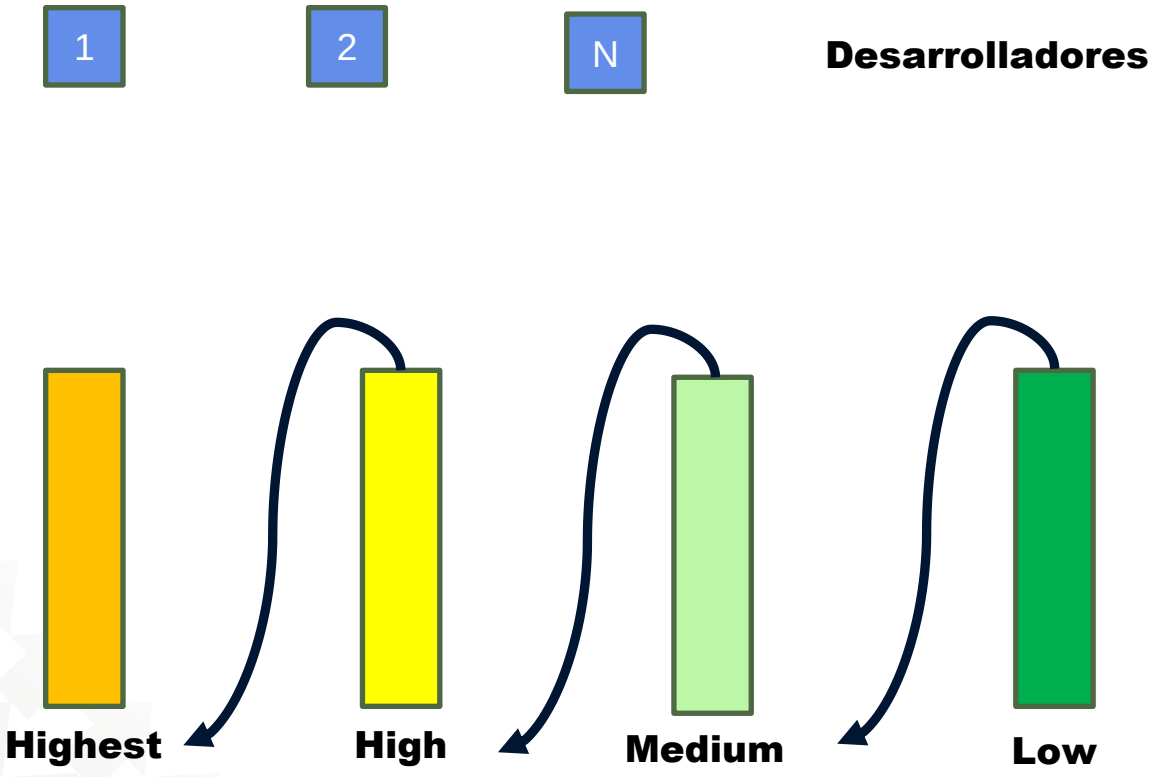


1. Problema:


1
2
N
Desarrolladores
Vacía
Vacía
Vacía
Highest
High
Medium
Low

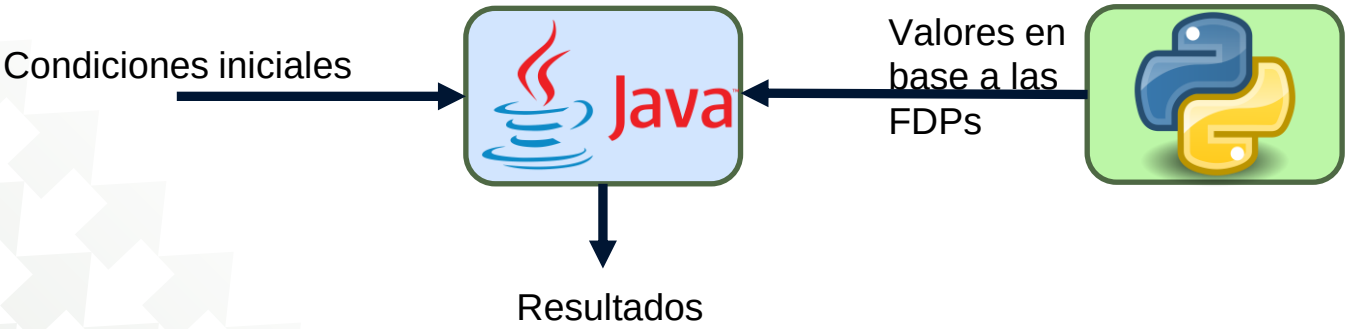
1. Problema:

ACTUALIZACION DE PRIORIDADES



2. Herramientas utilizadas:

- Kaggle
- Colab (Pandas, Numpy, SciPy/Stats, Fitter)
- Codigo en Java y Python

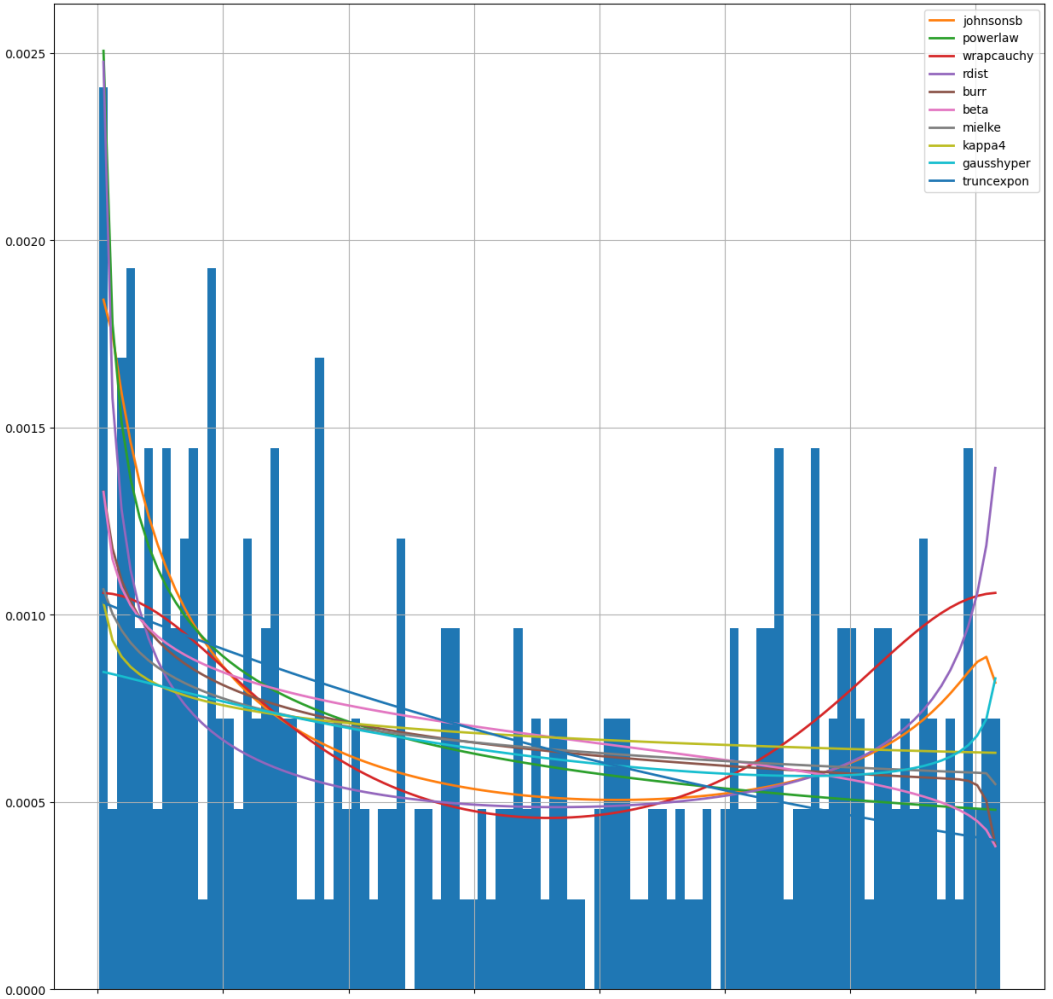


3. Análisis previo:

Var. Datos:

⇨ **Intervalo entre arribos de tickets:**

johnsonsb

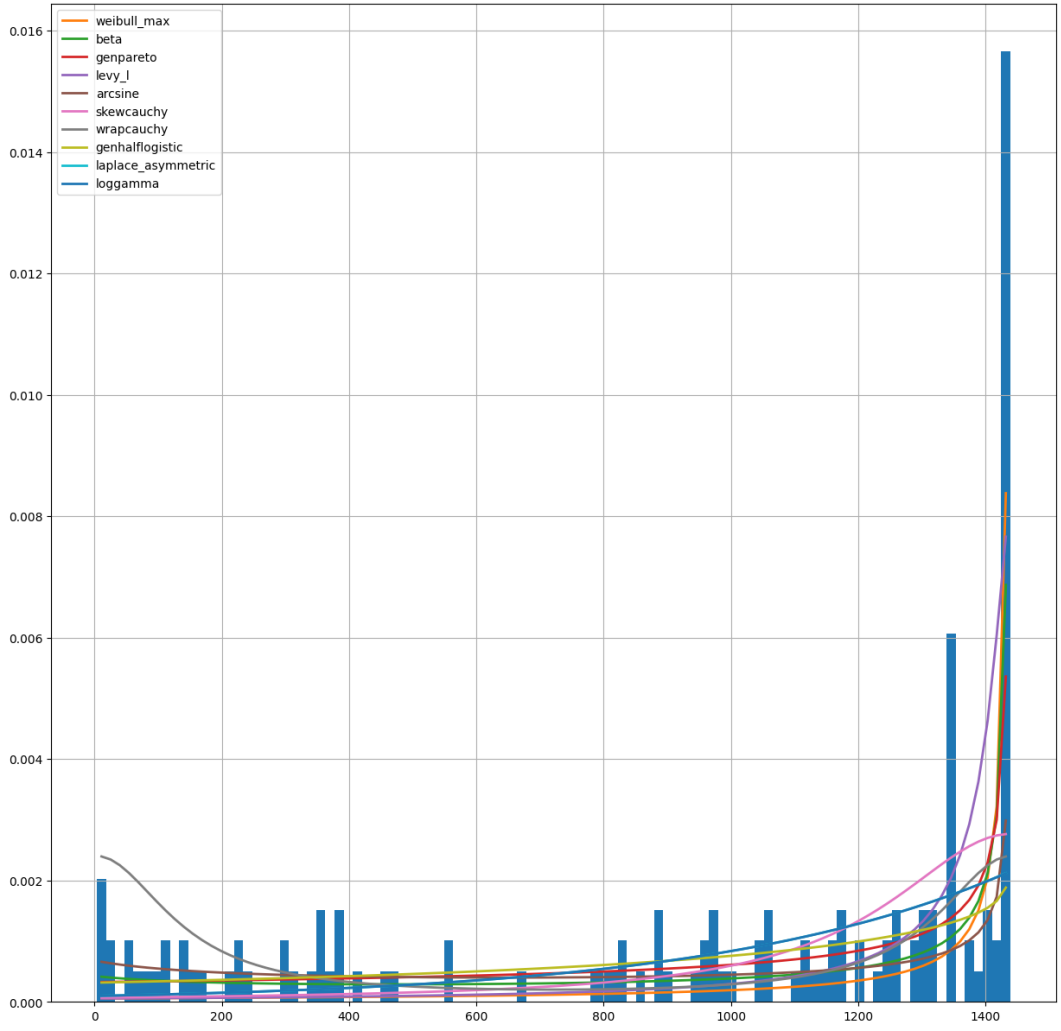


3. Análisis previo:

Var. Datos:

⇨ **Tiempo de resolución de un ticket:**

weibull_max

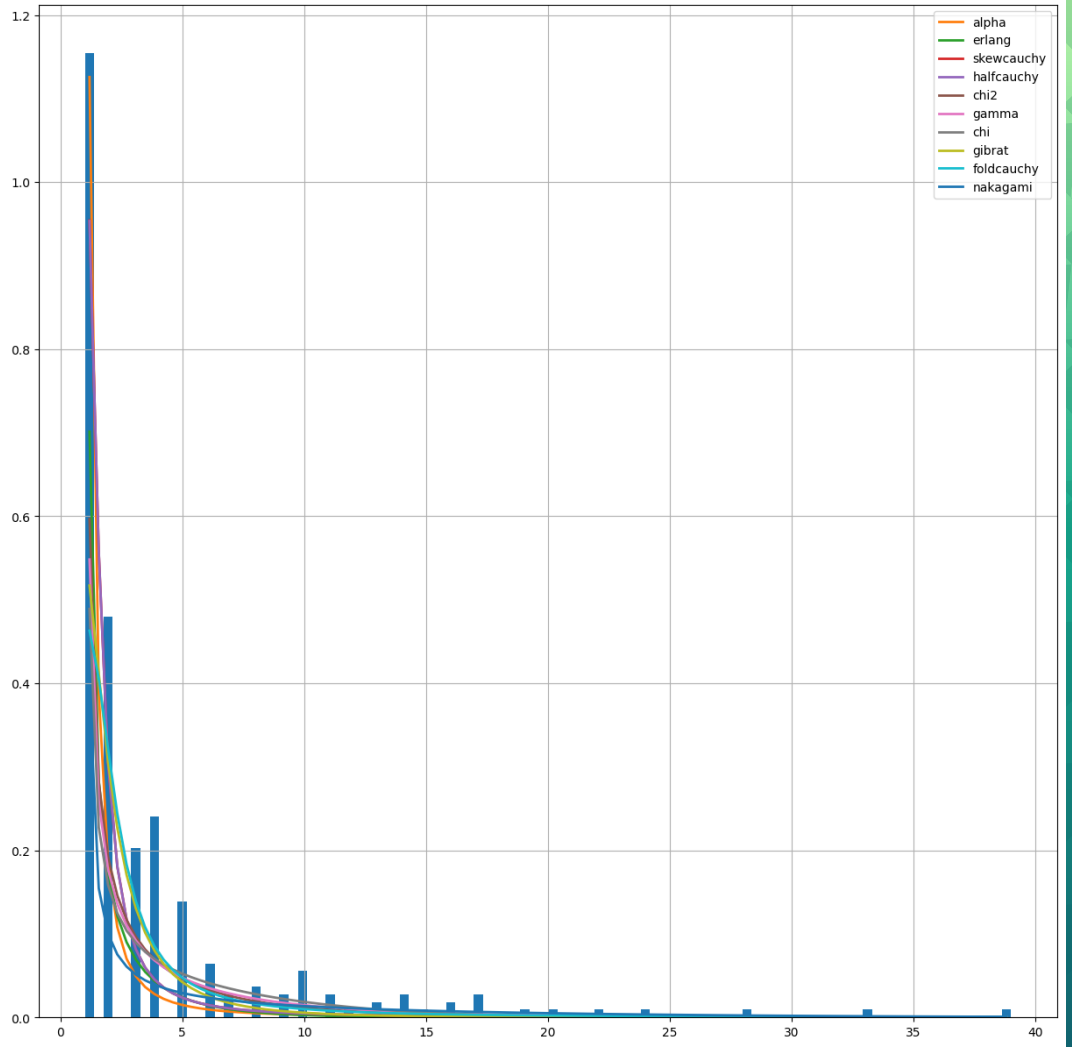


3. Análisis previo:

Var. Datos:

⇨ **Puntaje del ticket:**

alpha



3. Análisis previo:

Var. Control:

- Cantidad de desarrolladores
- Tiempo de actualización de prioridades de tickets
- Valor de cada punto de estimación de tickets

3. Análisis previo:

Var. Estado:

- Cantidad de tickets pendientes en prioridad HIGHEST
- Cantidad de tickets pendientes en prioridad HIGH
- Cantidad de tickets pendientes en prioridad MEDIUM
- Cantidad de tickets pendientes en prioridad LOW

3. Análisis previo:

Var. Resultado:

- Tiempo promedio de permanencia en el sistema de los tickets
- Promedio de tickets resueltos por semana
- Promedio de desfase de ticket

3. Análisis previo:

Tabla de eventos independientes:

Eventos	EFnoC	EFC	Condición
Llegada	Llegada	Salida(i)	tickets en HIGHEST + tickets en HIGH + tickets en MEDIUM + tickets en LOW <= Cantidad de desarrolladores
Salida(i)	-	Salida(i)	tickets en HIGHEST + tickets en HIGH + tickets en MEDIUM + tickets en LOW >= Cantidad de desarrolladores
Actualización	Actualización	-	-

1 <= i <= Cantidad de desarrolladores

Tabla de eventos futuros:

- Tiempo de próxima llegada
- Tiempo de próxima salida(i)
- Tiempo de próxima actualización

4. Escenarios planteados:

Tiempo final = 20 años

Escenarios:	N°1	N°2	N°3	N°4	N°5	N°6
Cantidad de programadores	1	3	5	10	5	5
Actualización de prioridades de tickets	Cada 2 dias	Cada 2 dias	Cada 2 dias	Cada 2 dias	Cada 7 dias	Cada 2 dias
Valor de cada punto de estimación de tickets	12 horas	12 horas	12 horas	12 horas	12 horas	8 horas

5. Resultados:

Tiempo final = 20 años

Escenarios:	N°1	N°2	N°3	N°4	N°5	N°6
Cantidad de programadores	1	3	5	10	5	5
Actualización de prioridades de tickets	Cada 2 días	Cada 2 días	Cada 2 días	Cada 2 días	Cada 7 días	Cada 2 días
Valor de cada punto de estimación de tickets	12 horas	12 horas	12 horas	12 horas	12 horas	8 horas
Resultados						
Tiempo promedio de permanencia en el sistema de los tickets	40.881 días	1.798 días	2.676 días	4.903 días	2.677 días	2.672 días
Promedio de tickets resueltos por semana	15.849	15.645	15.699	15.701	15.692	15.707
Promedio de desfase de ticket	-32.6 horas	-24.6 horas	10.8 horas	59.1 horas	4.817 horas	3.831 horas

6. Conclusión:

Lo más optimo:

- ⇨ **5 Desarrolladores**
- ⇨ **Actualización de prioridades cada 2 días**
- ⇨ **1 Punto de estimación = 12 horas**

" ¡Gracias por su atención!

Grupo 2:

- **Ornella Fasciolo**
- **Gabriel Spisso**
- **Alejandro Deheza**