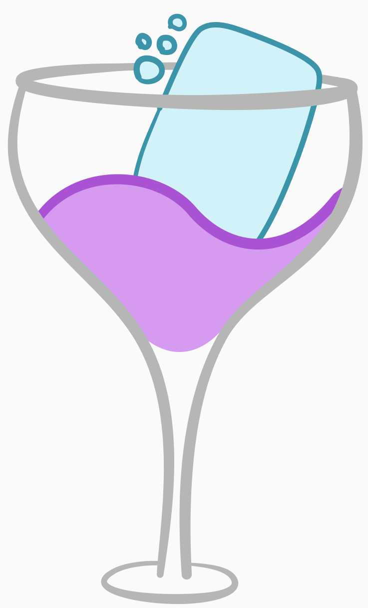
Uniform Bounds for **Scheduling** *with* **Job Size Estimates**

Ziv Scully
Cornell

Isaac Grosof
Carnegie Mellon → Northwestern

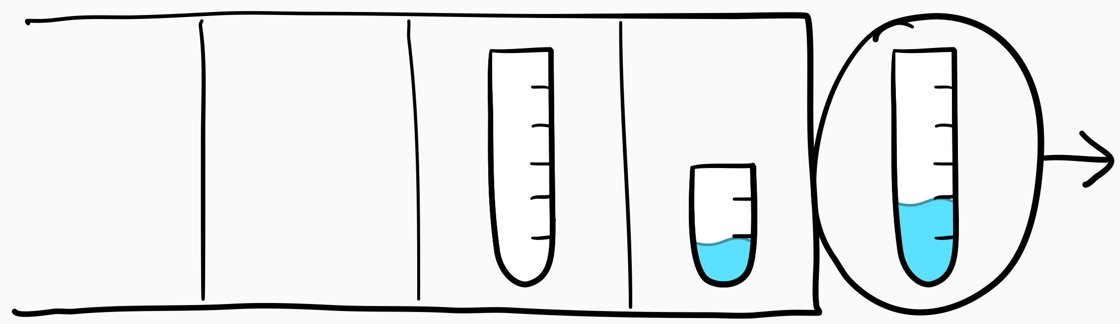
Michael Mitzenmacher
Harvard



algorithms

with

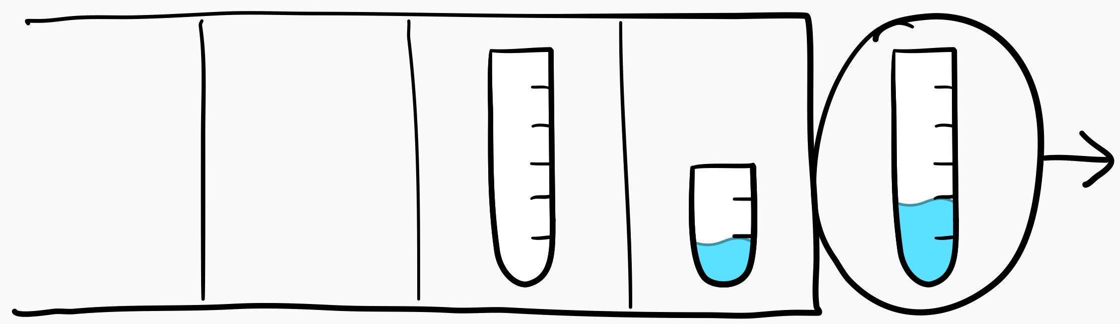
predictions



scheduling algorithms

with

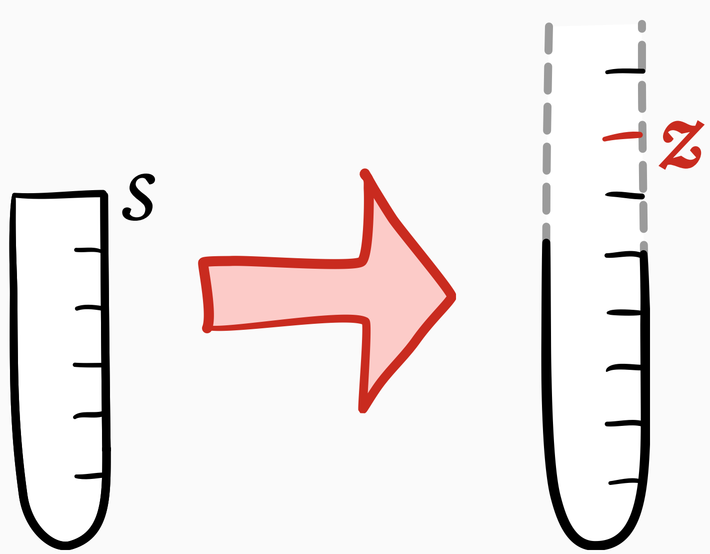
predictions

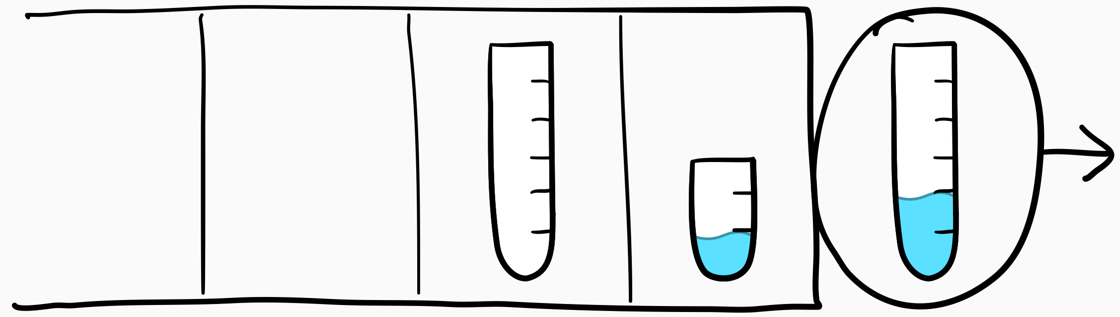


scheduling algorithms

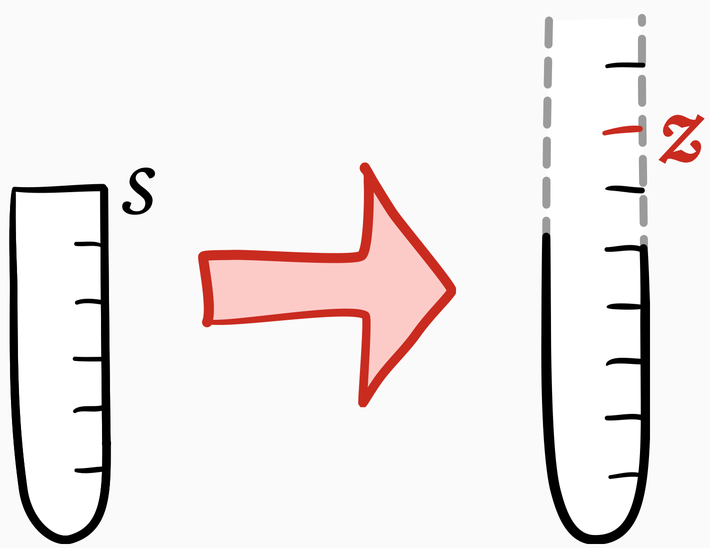
with

job size predictions





scheduling algorithms
with
job size predictions



Twist: *stochastic* setting

How do we schedule
to minimize delay
with noisy size estimates?

How do we schedule to minimize delay with noisy size estimates?

What are scheduling and delay?

How do we schedule
to minimize delay
with noisy size estimates?

What are scheduling and delay?

What job size noise model?

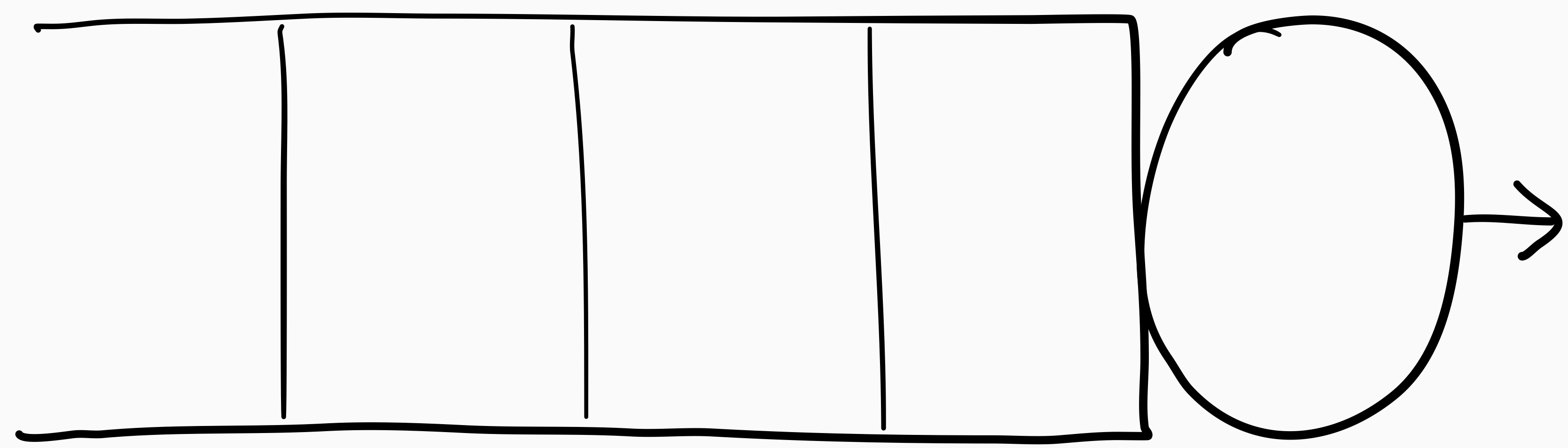
How do we schedule to minimize delay with noisy size estimates?

What are scheduling and delay?

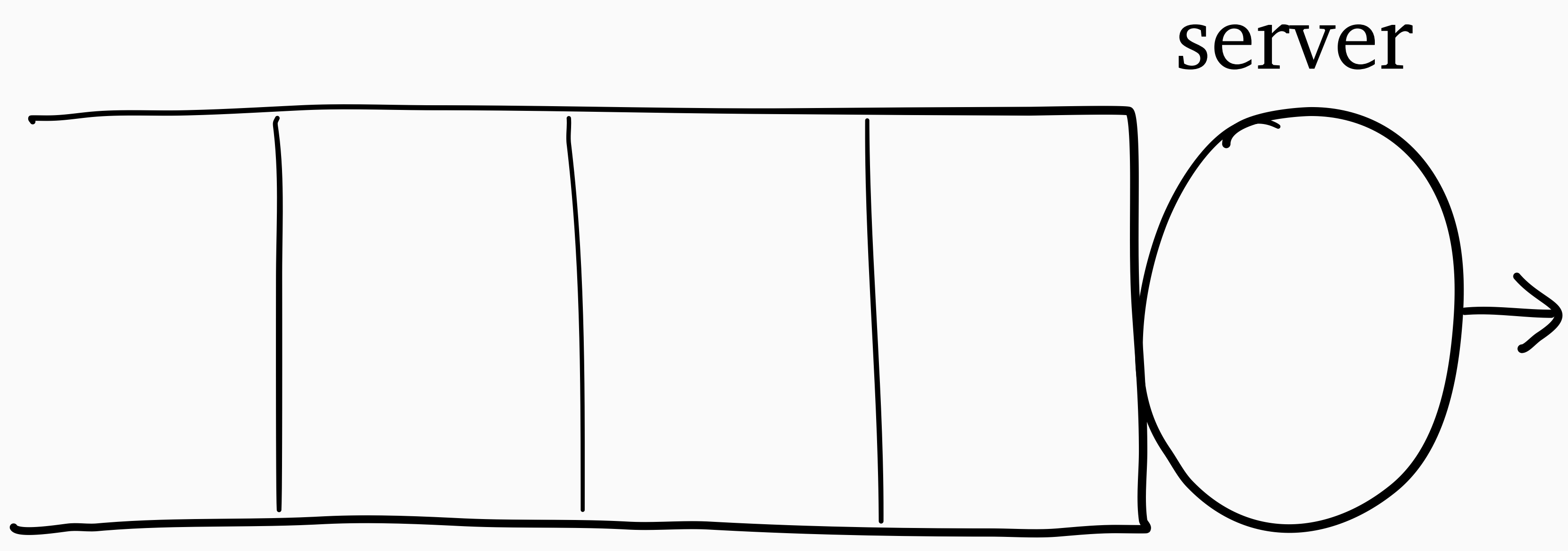
What job size noise model?

What can we hope to achieve?

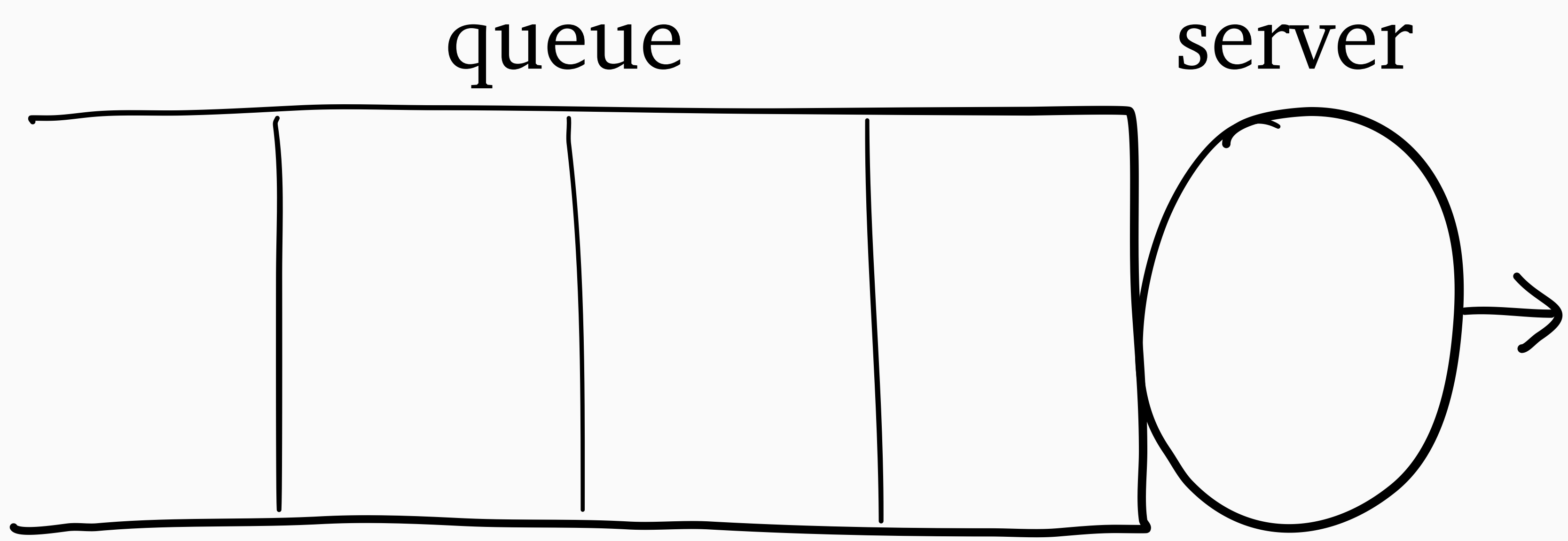
What are scheduling and delay?



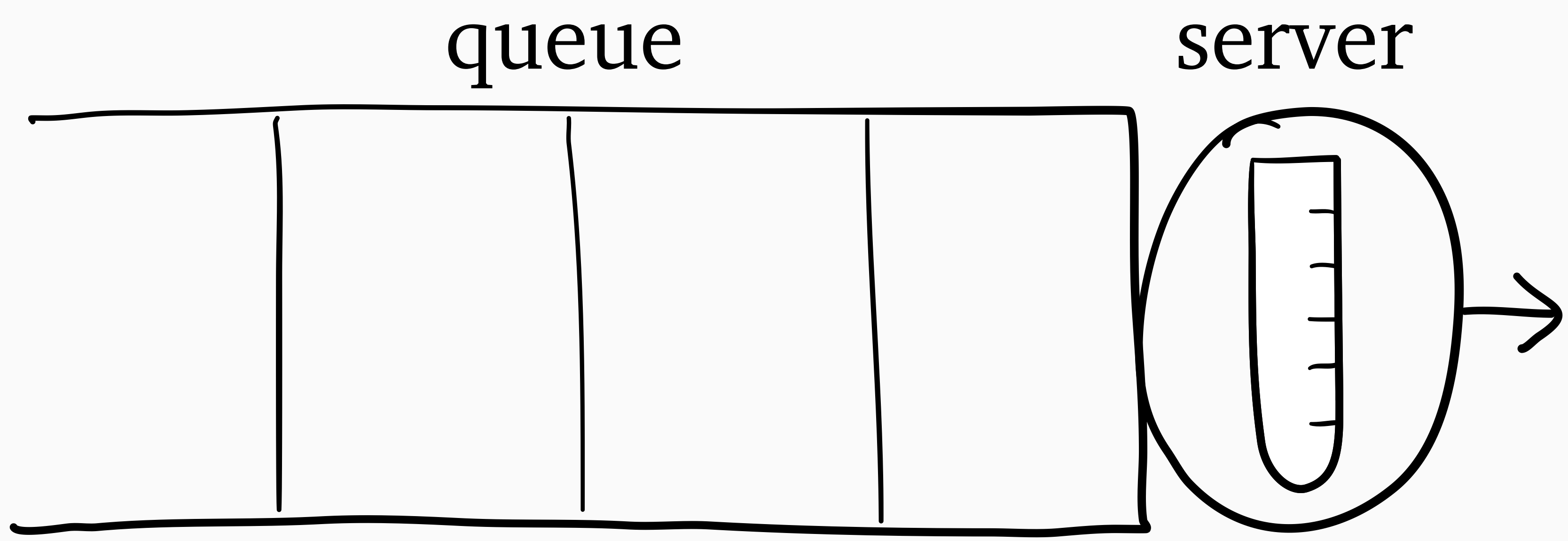
What are scheduling and delay?

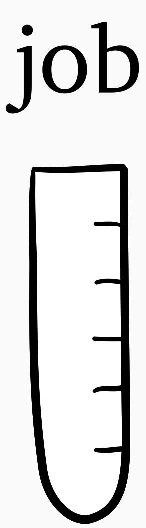


What are scheduling and delay?

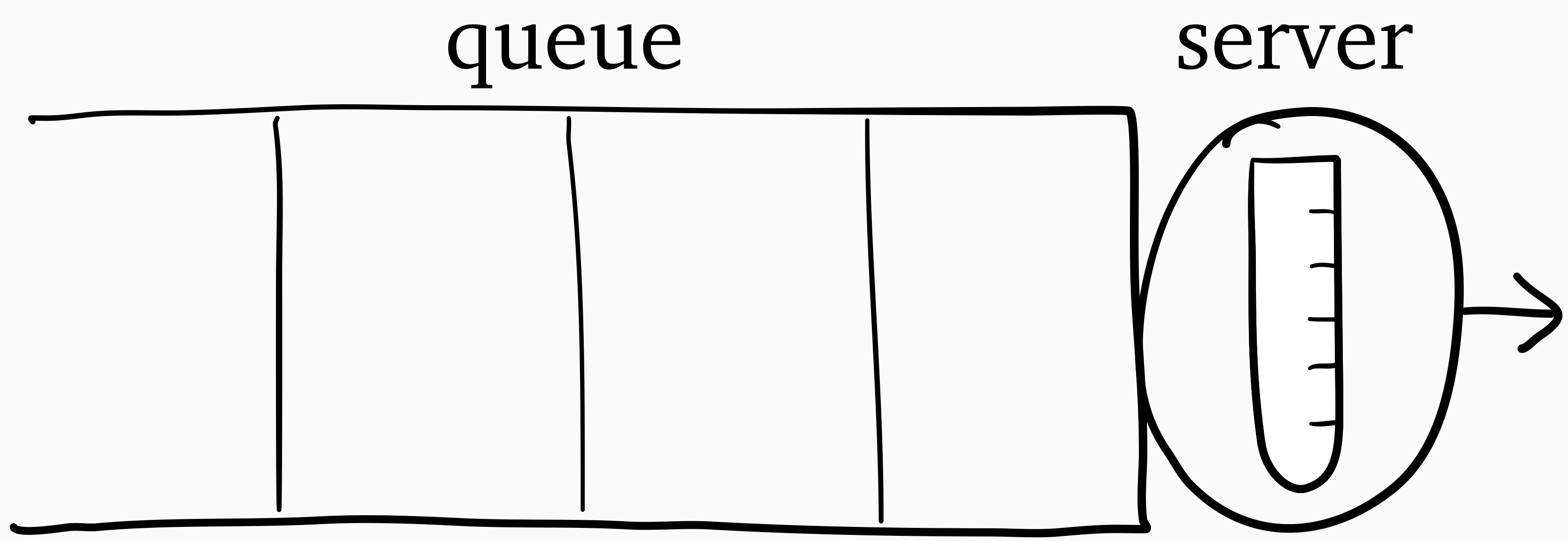


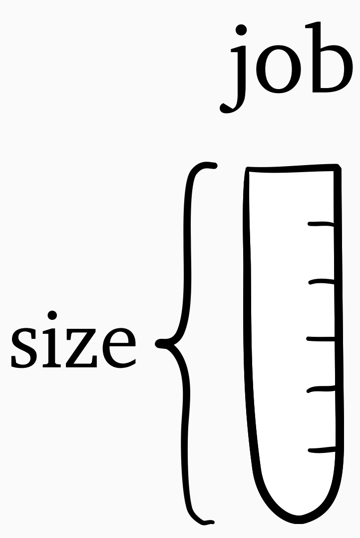
What are scheduling and delay?



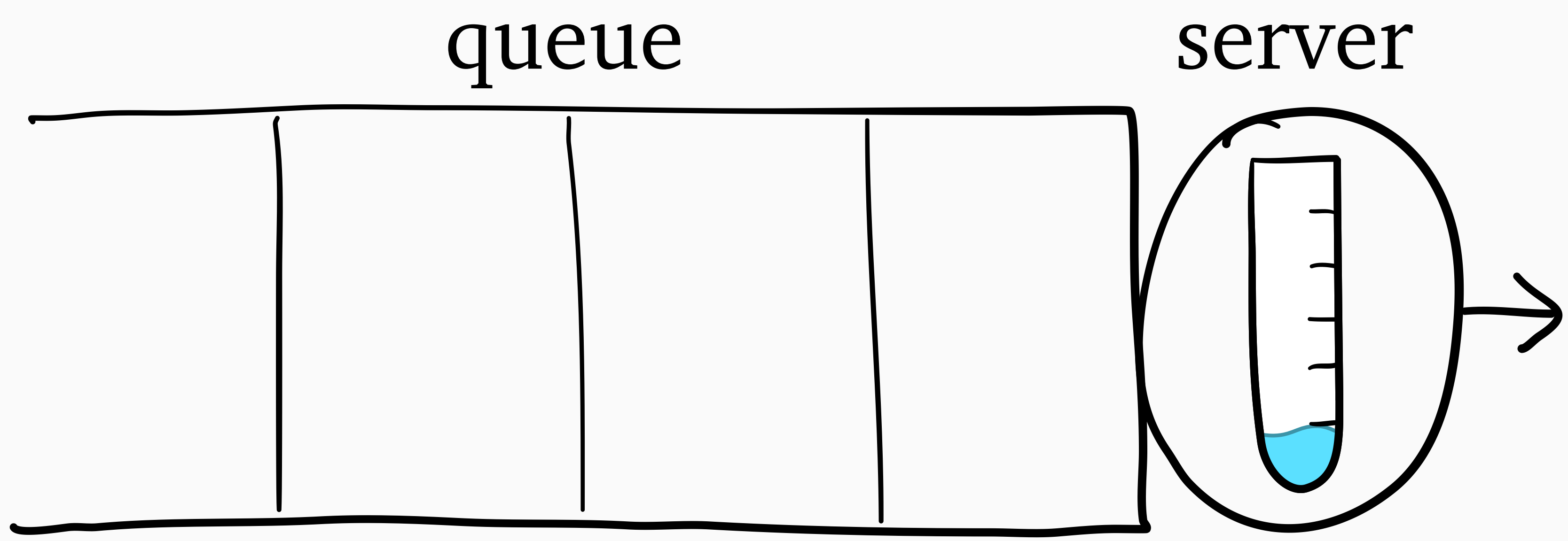


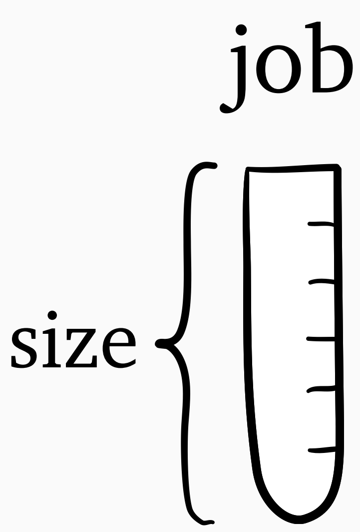
What are scheduling and delay?



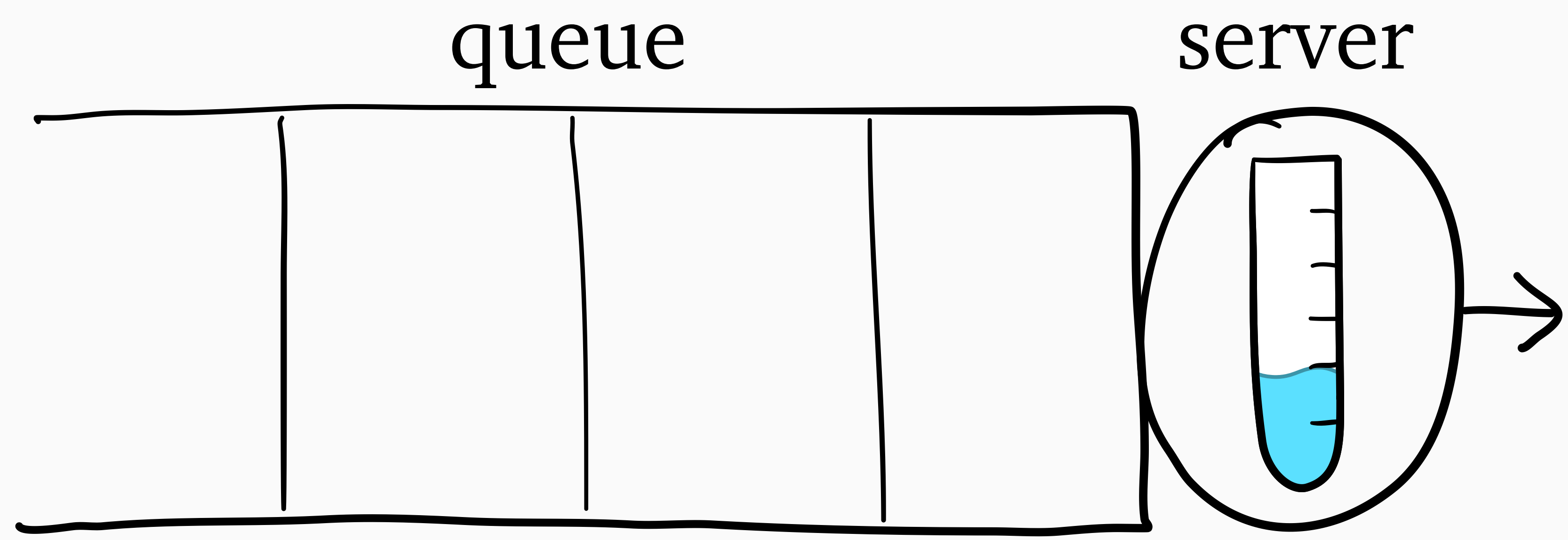


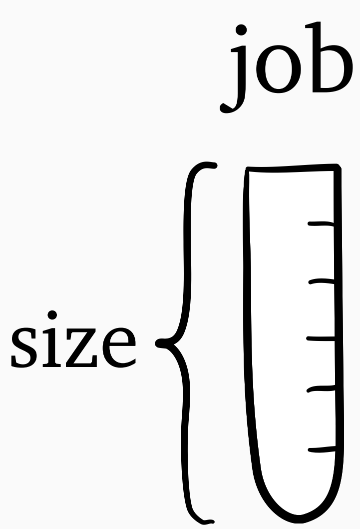
What are scheduling and delay?



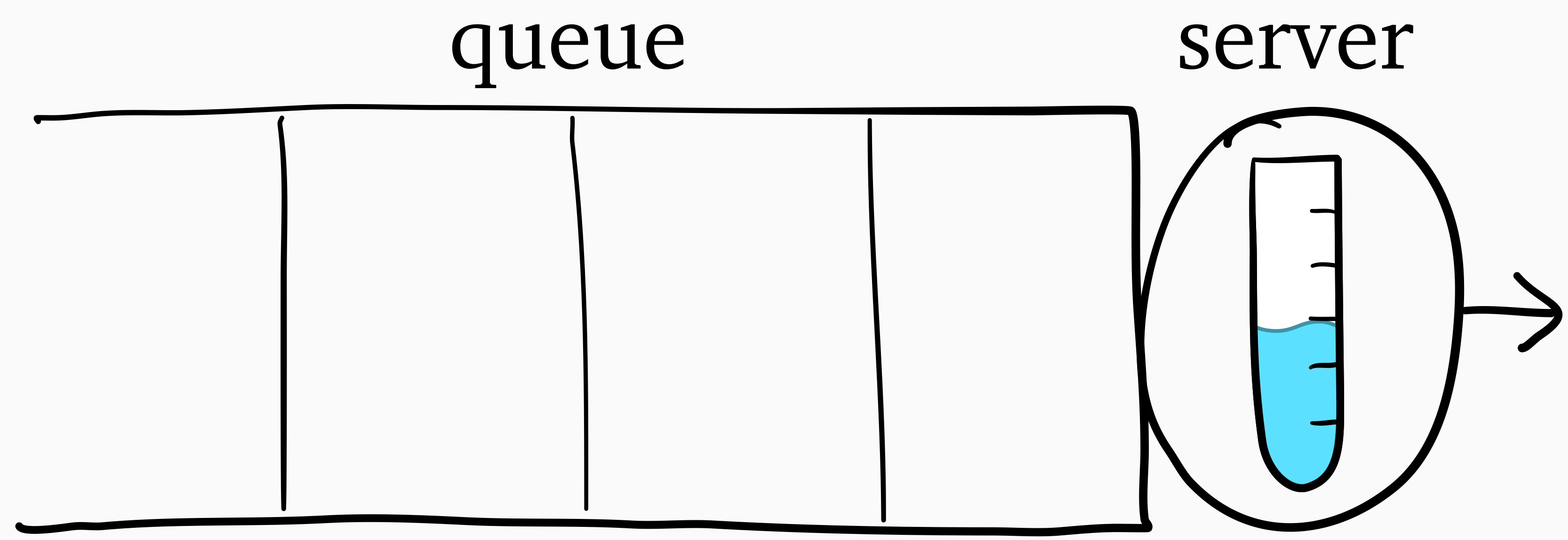


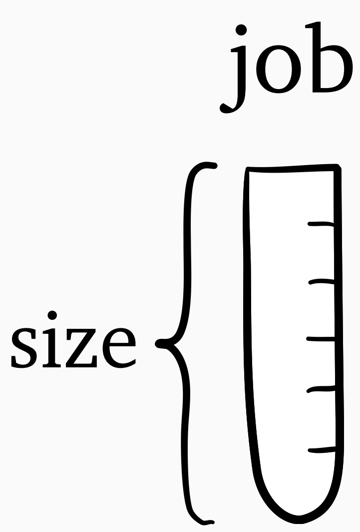
What are scheduling and delay?



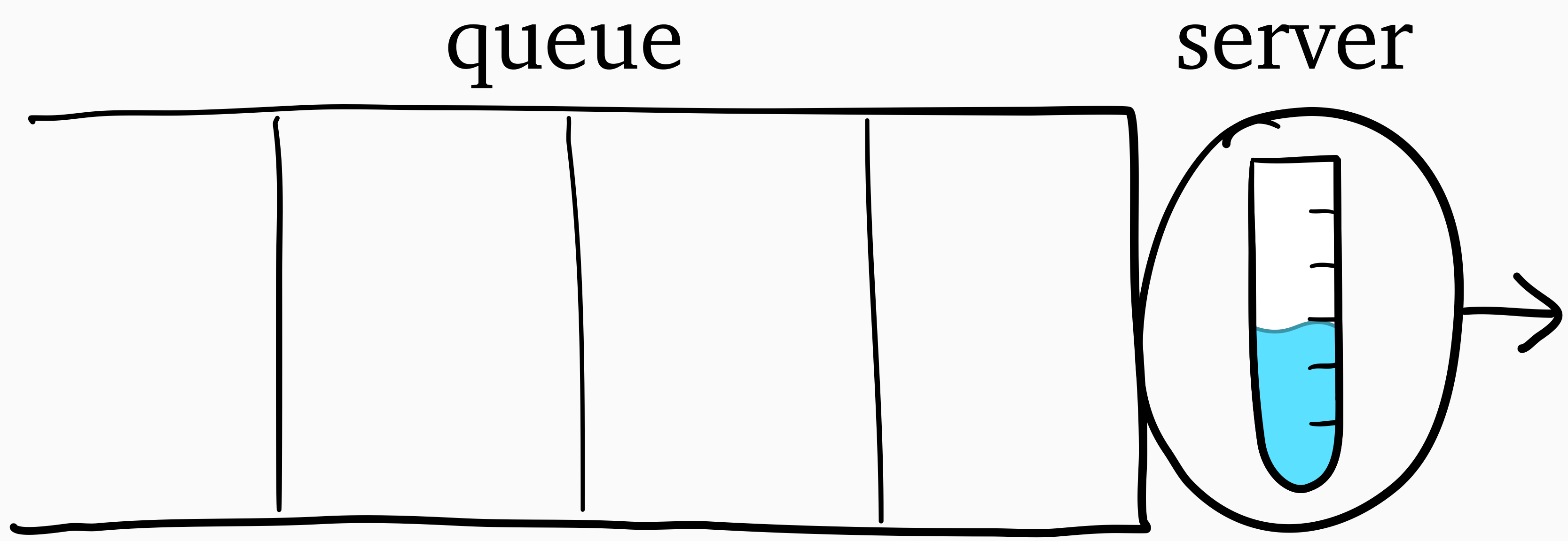


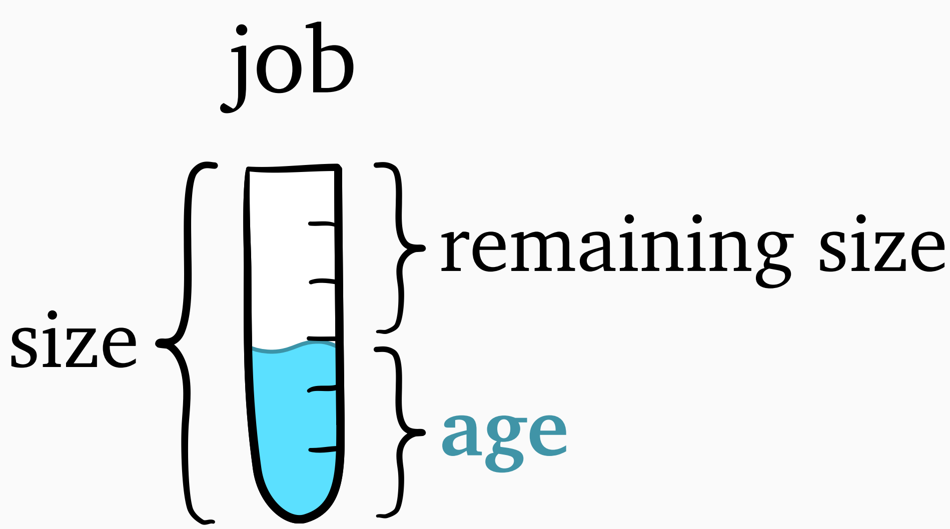
What are scheduling and delay?



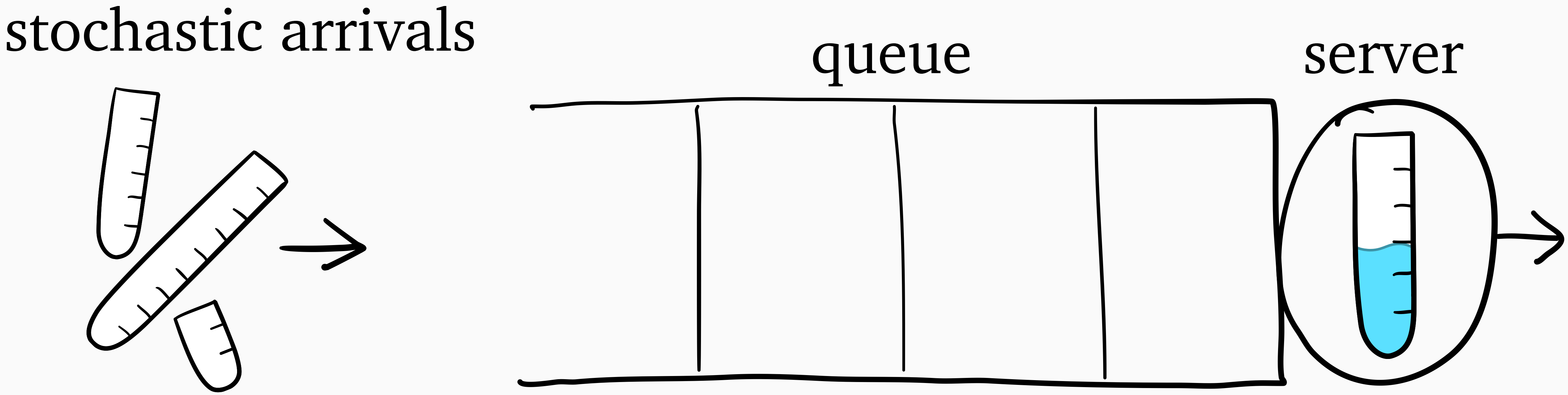


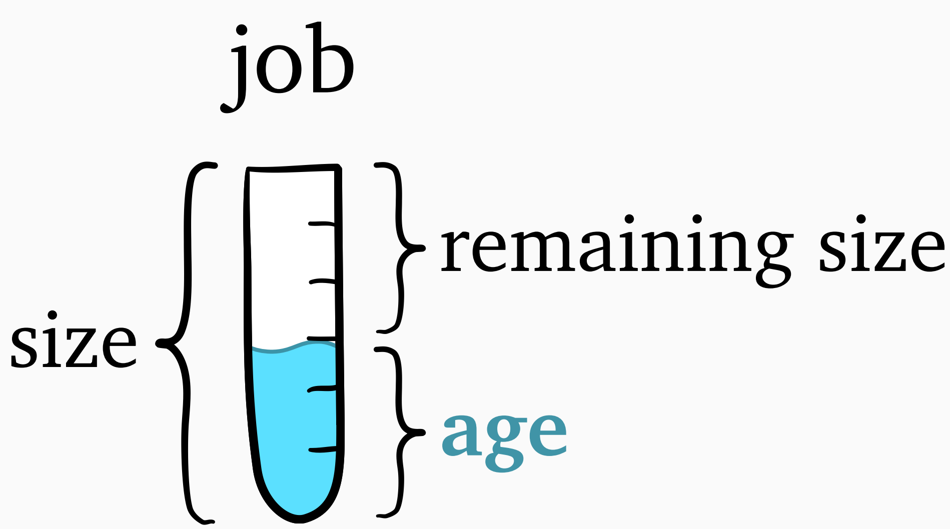
What are scheduling and delay?



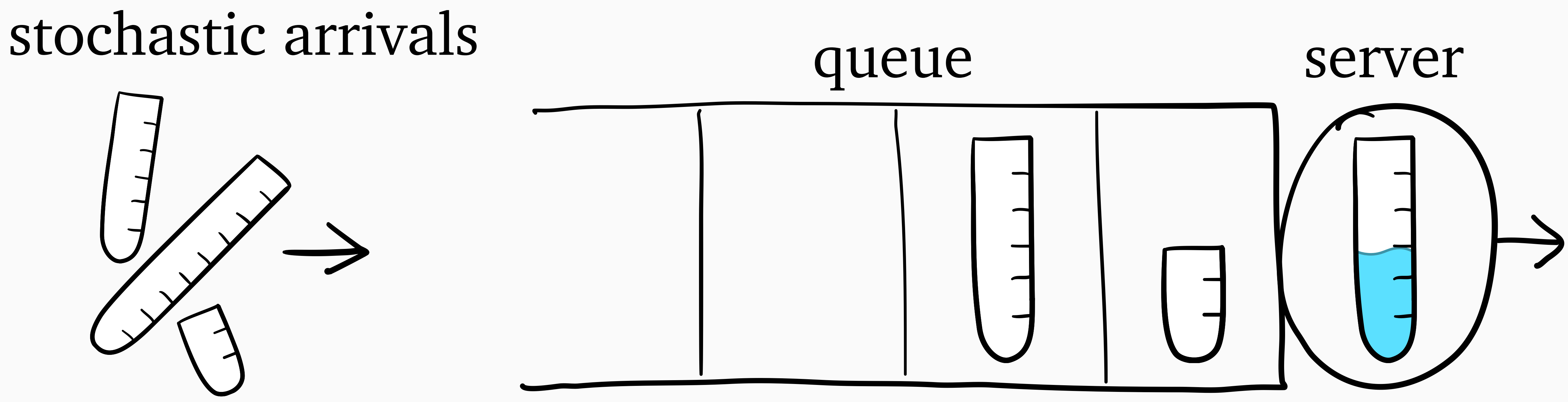


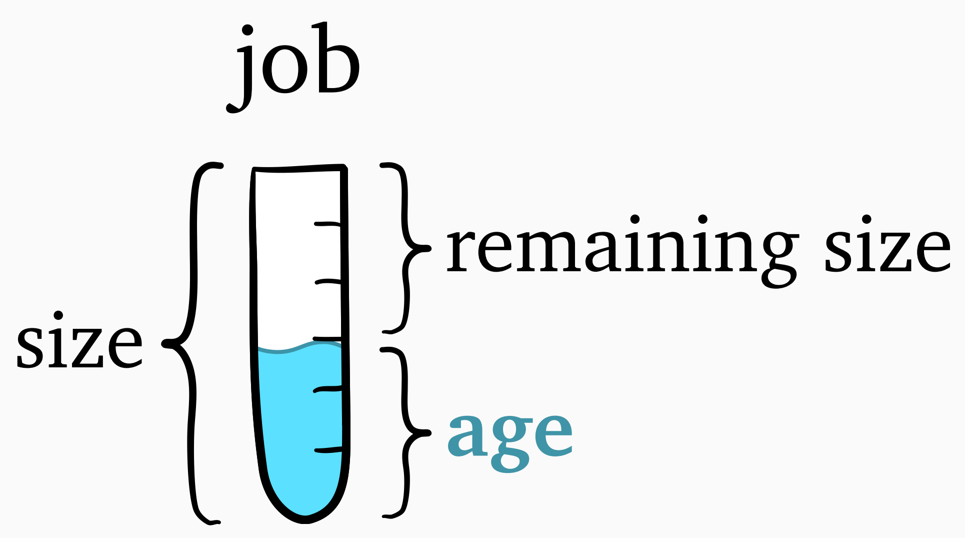
What are scheduling and delay?



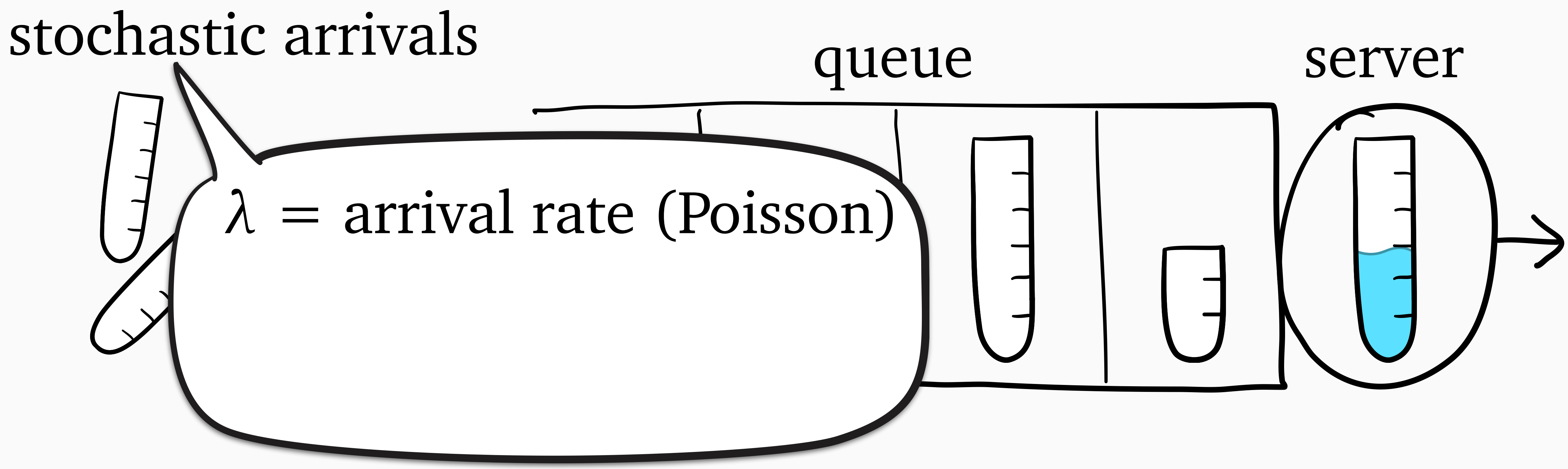


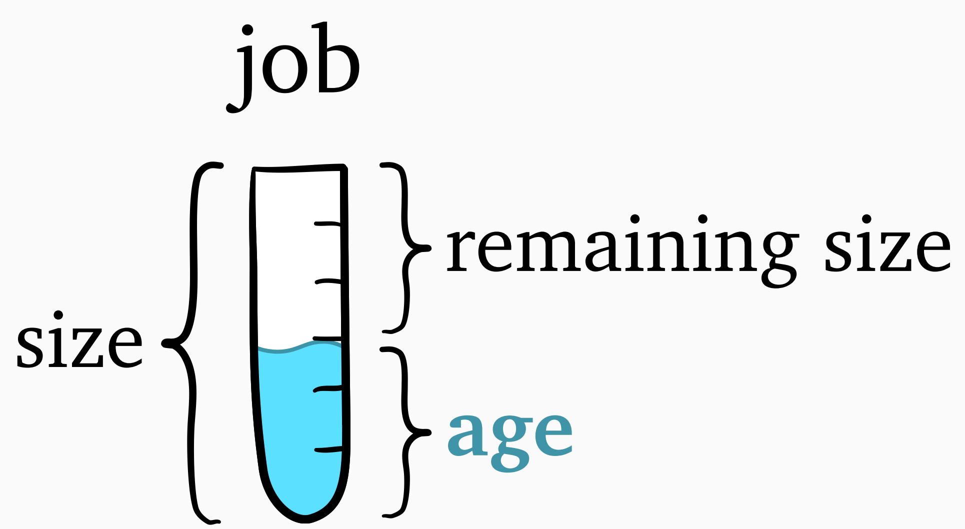
What are scheduling and delay?



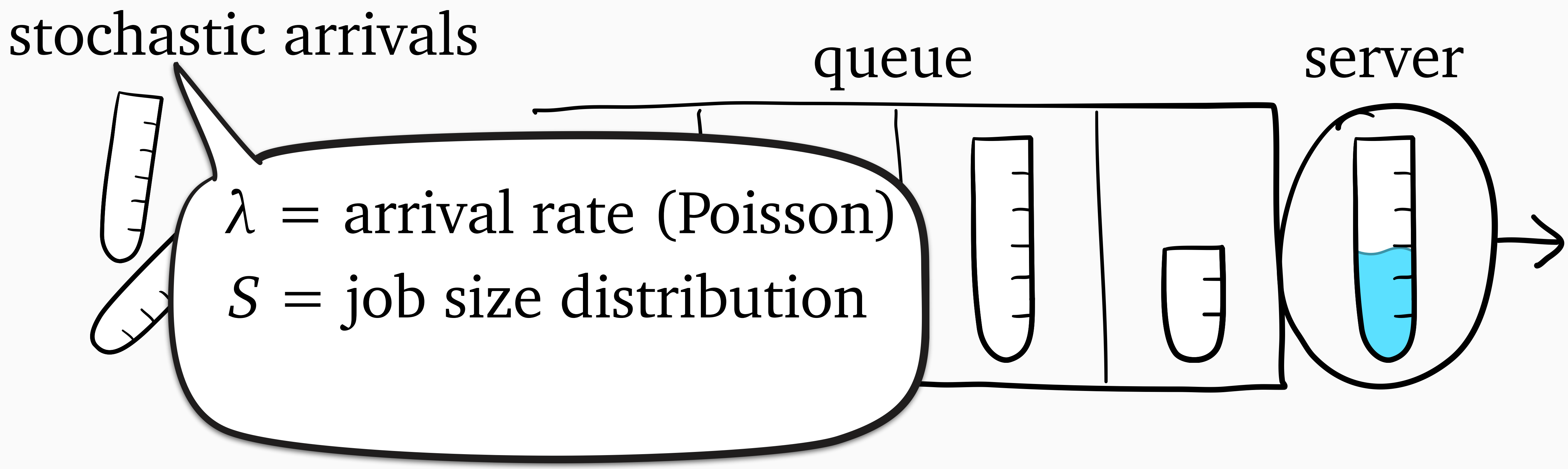


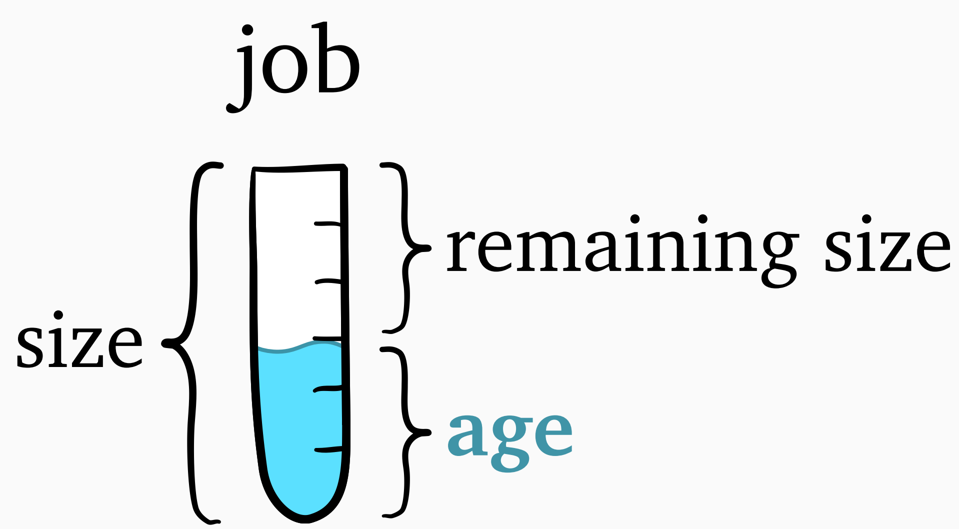
What are scheduling and delay?



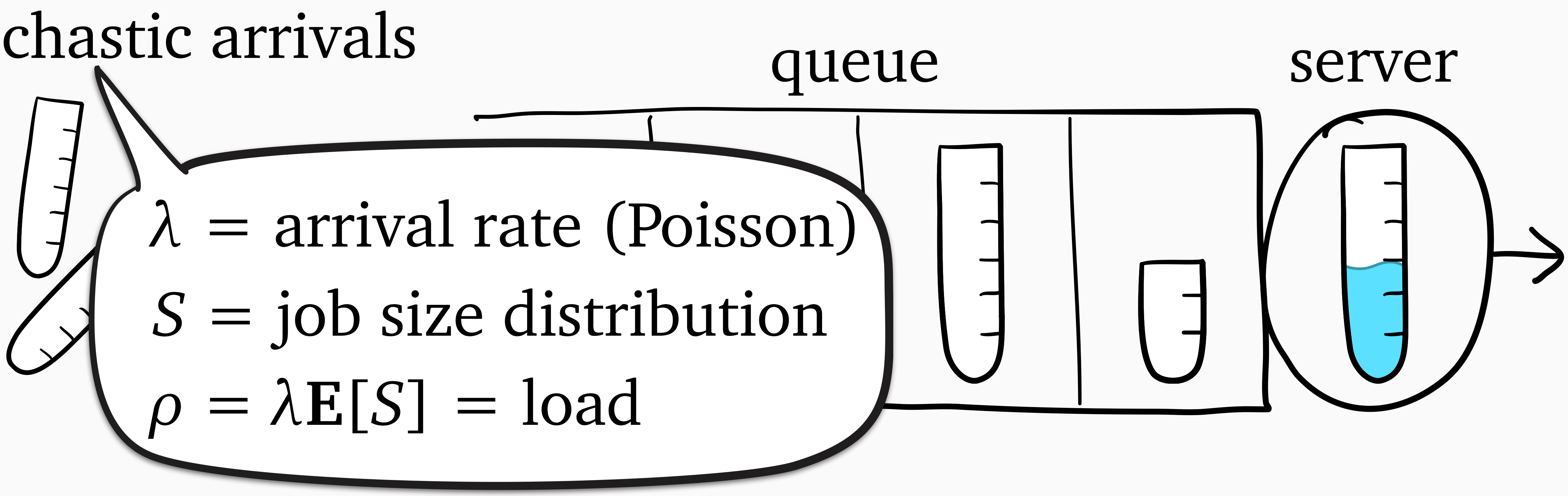


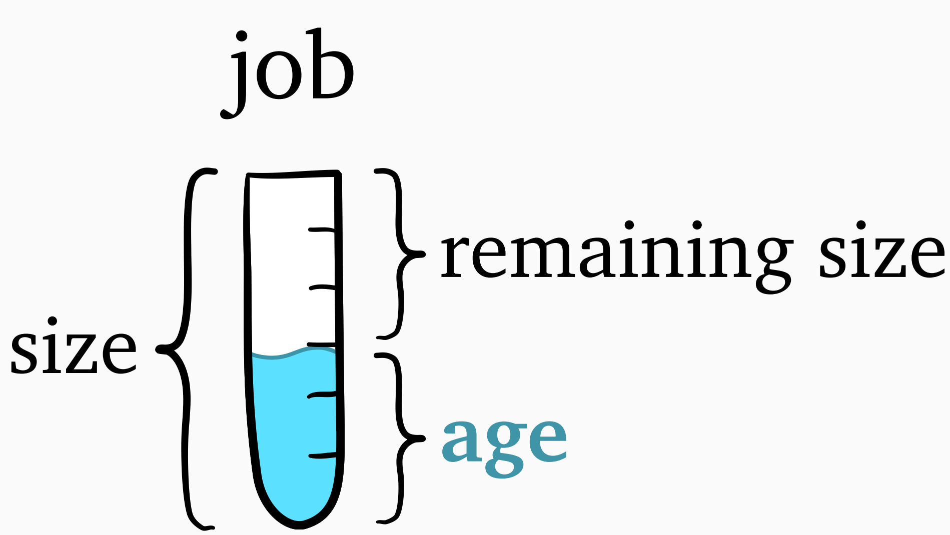
What are scheduling and delay?



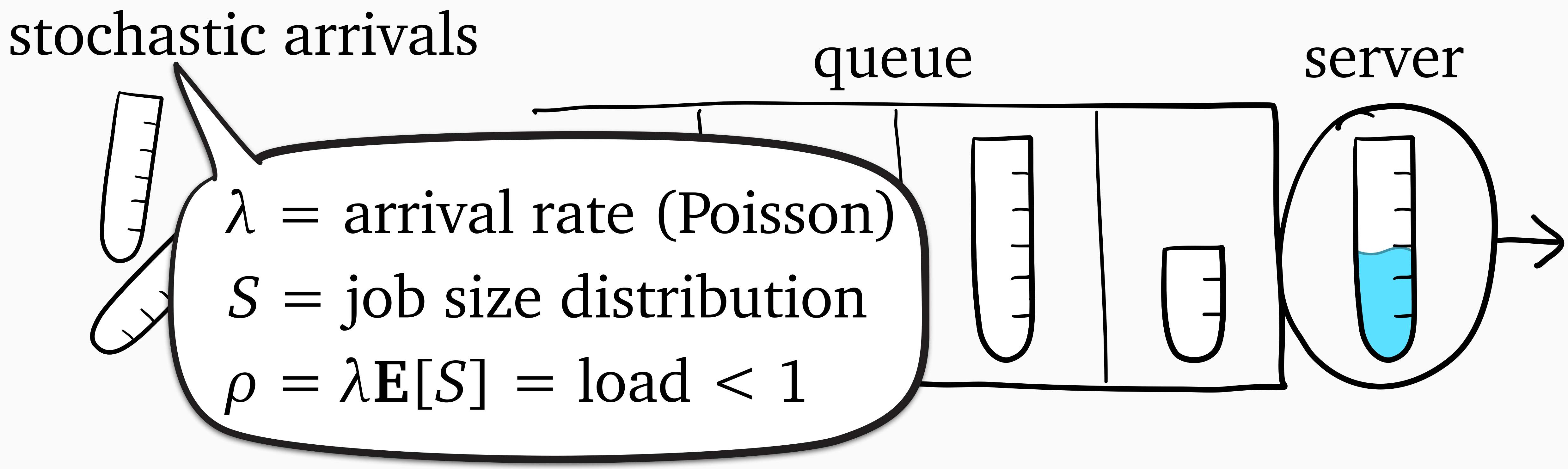


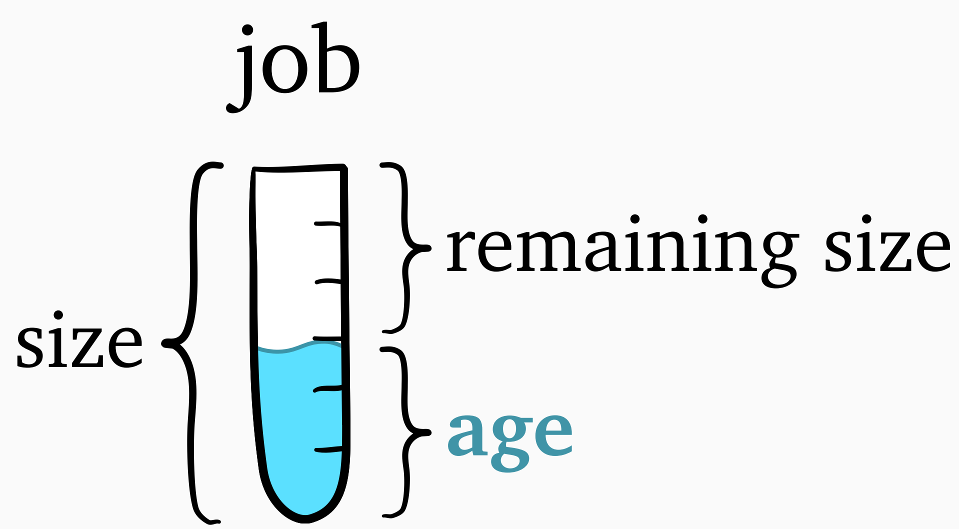
What are scheduling and delay?



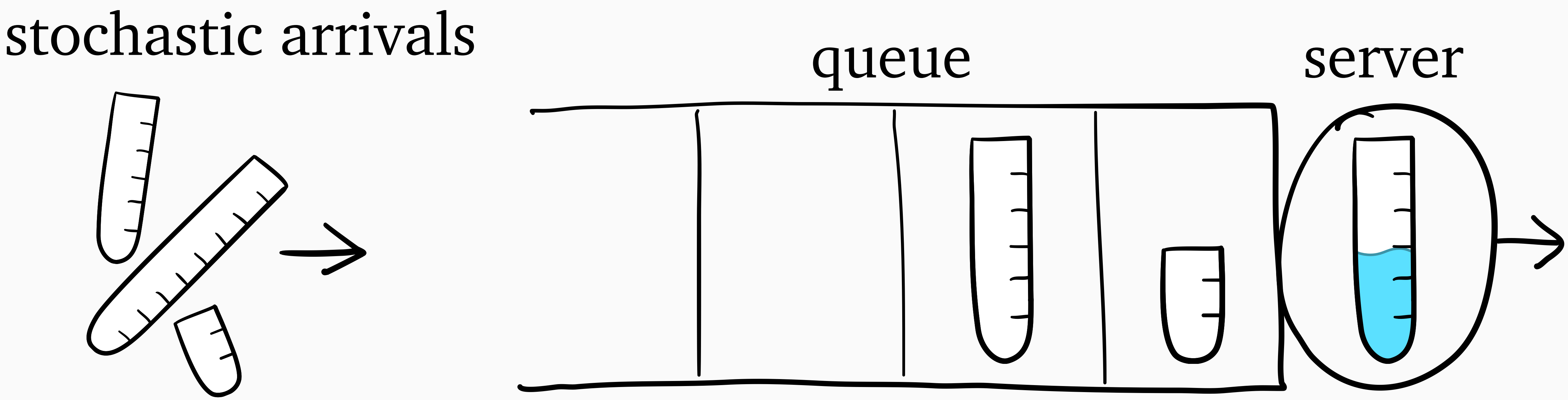


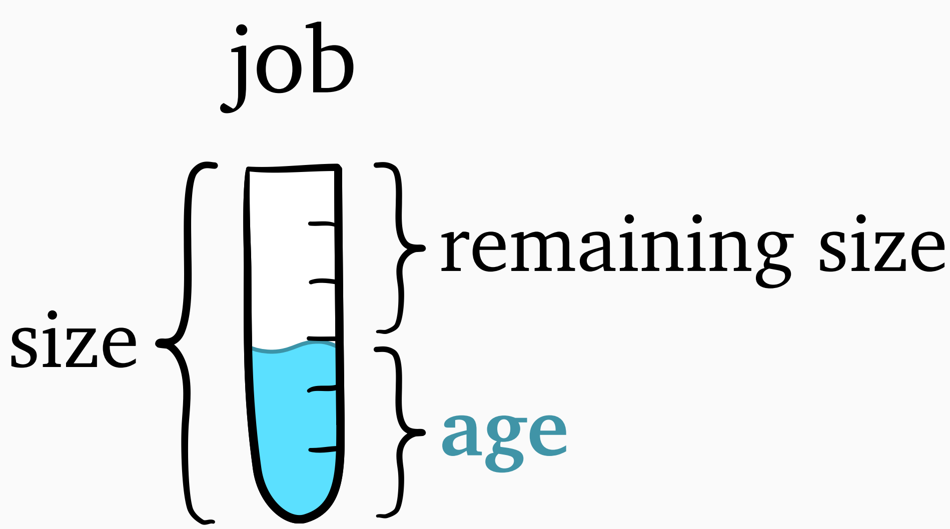
What are scheduling and delay?



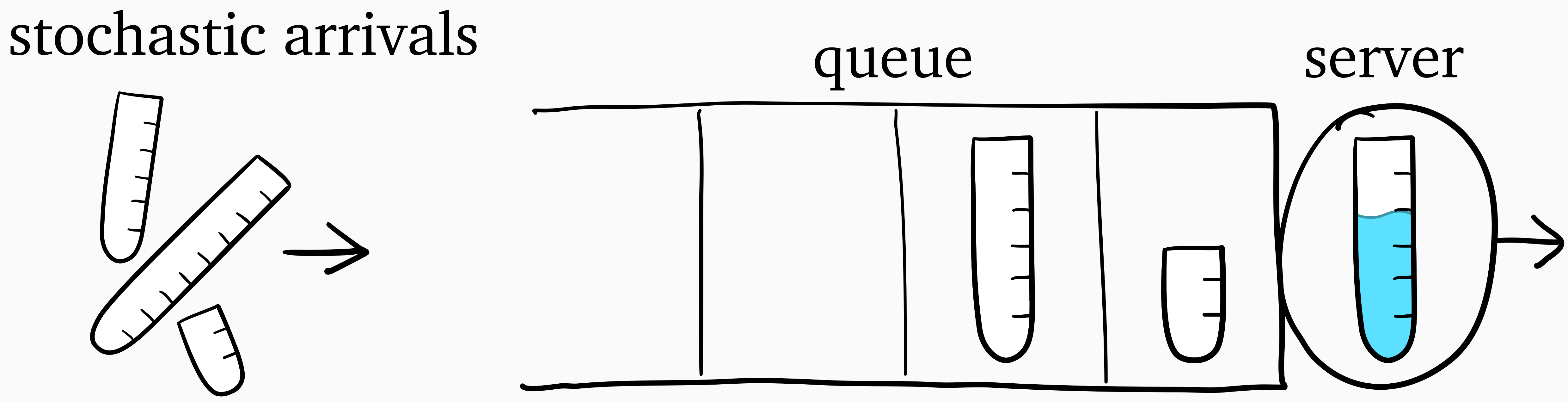


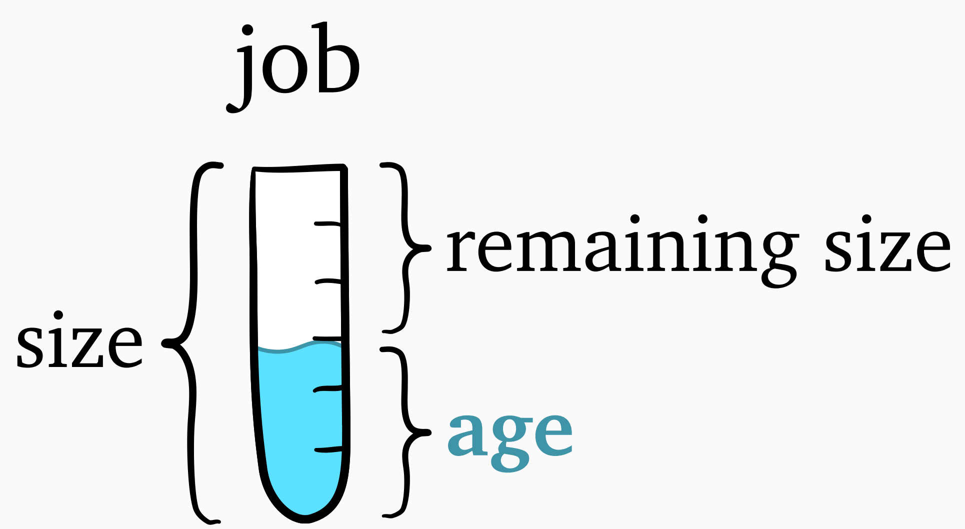
What are scheduling and delay?



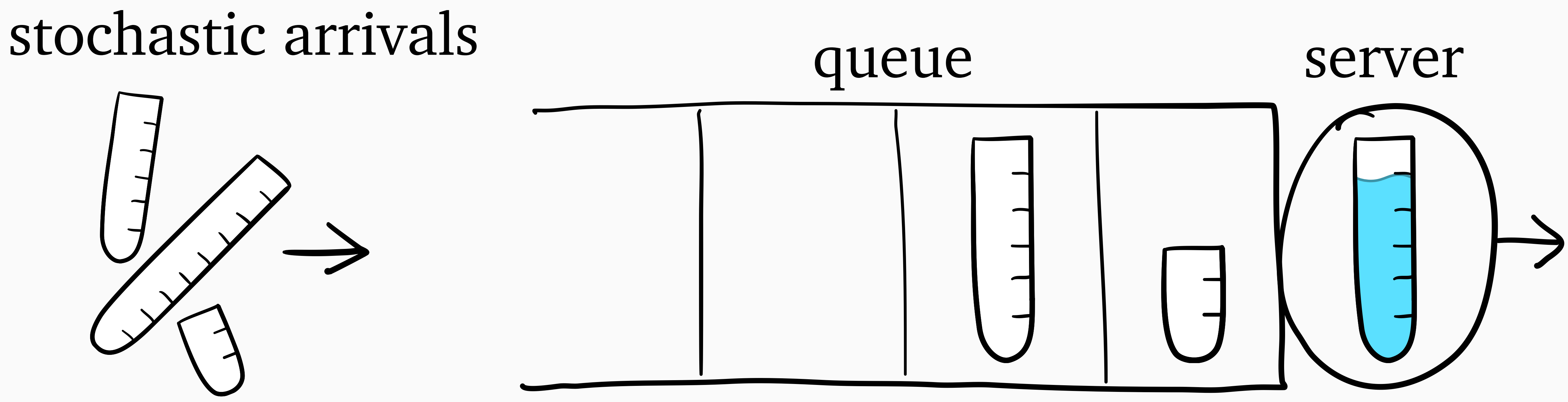


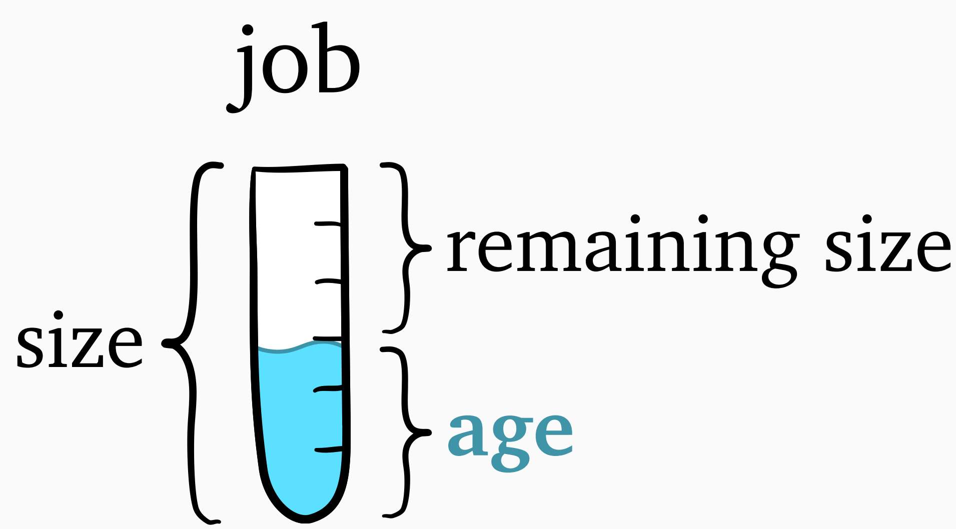
What are scheduling and delay?



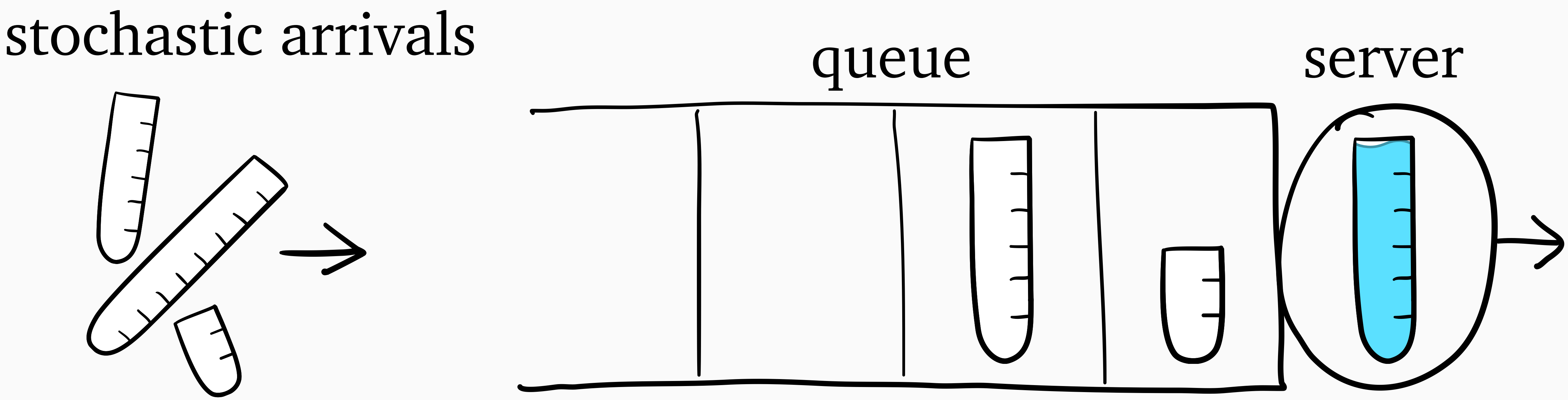


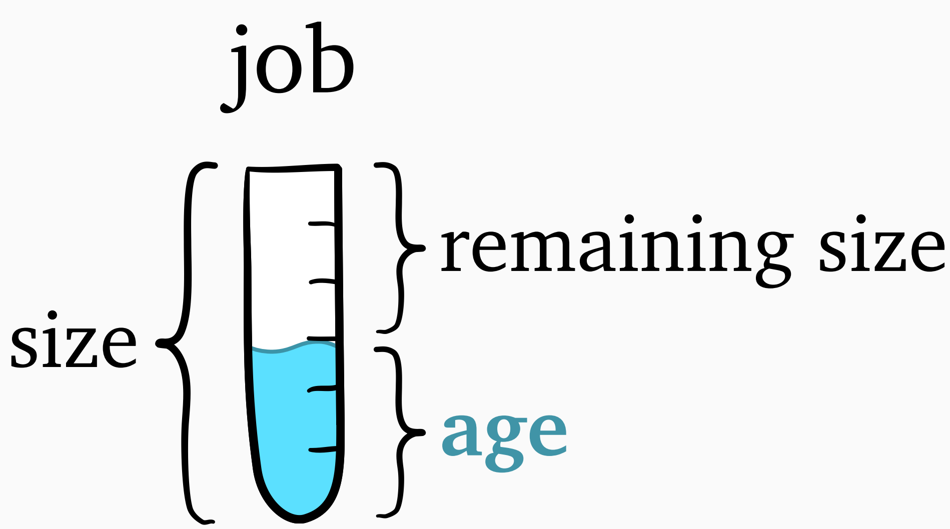
What are scheduling and delay?



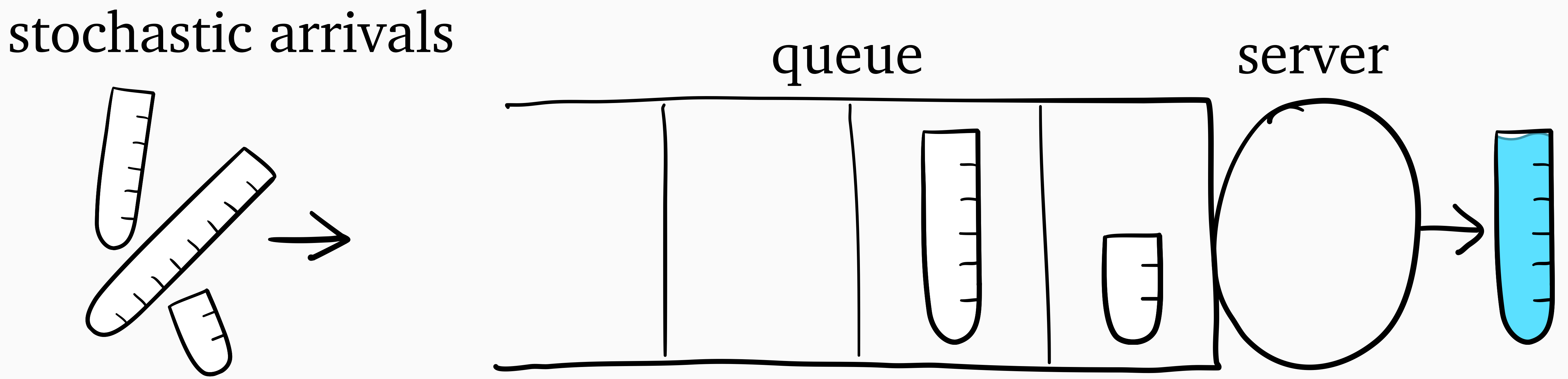


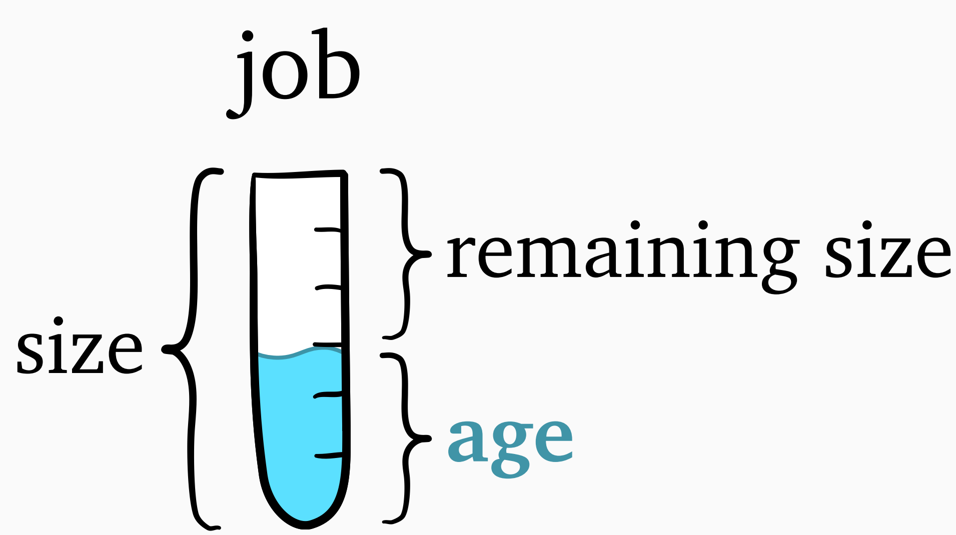
What are scheduling and delay?



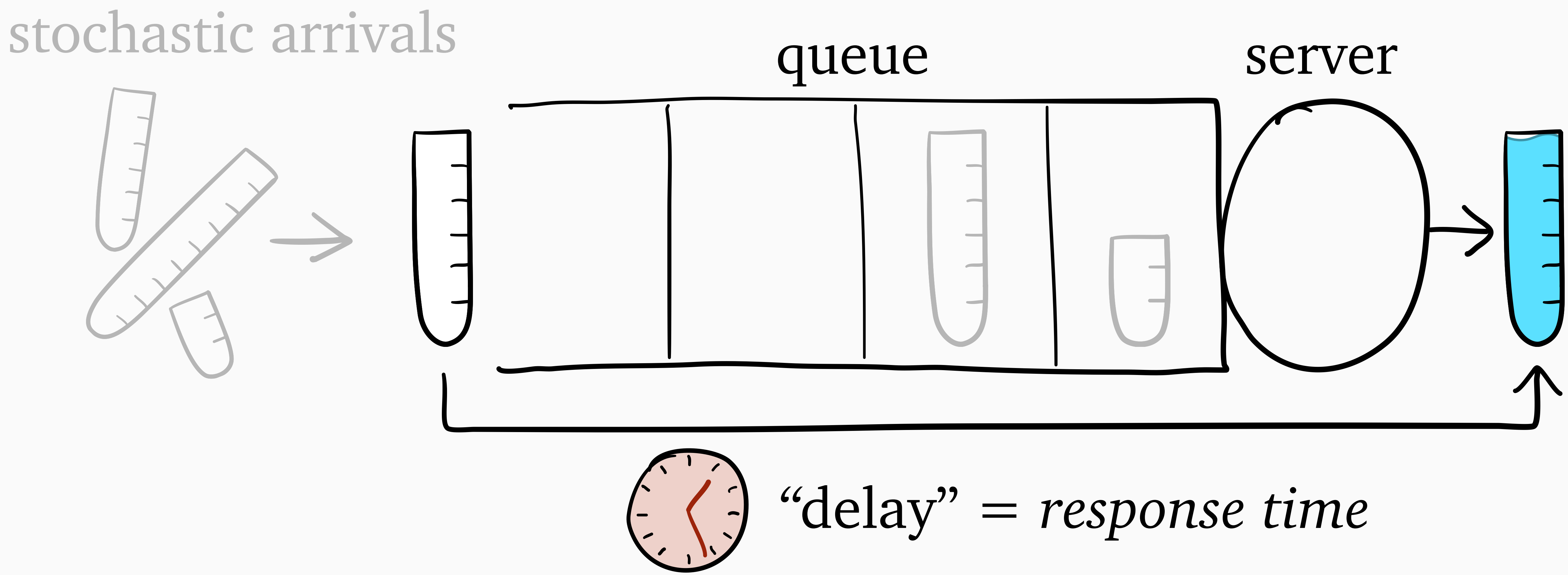


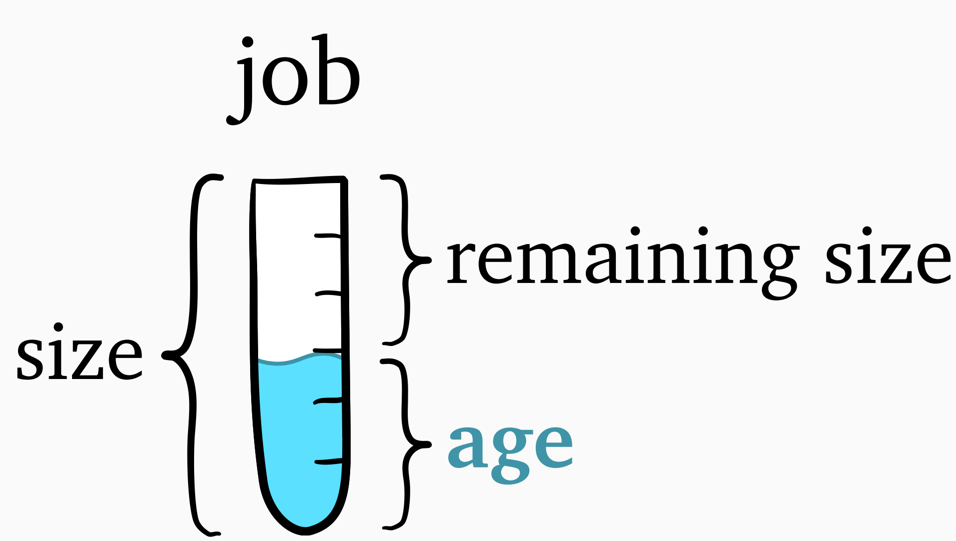
What are scheduling and delay?



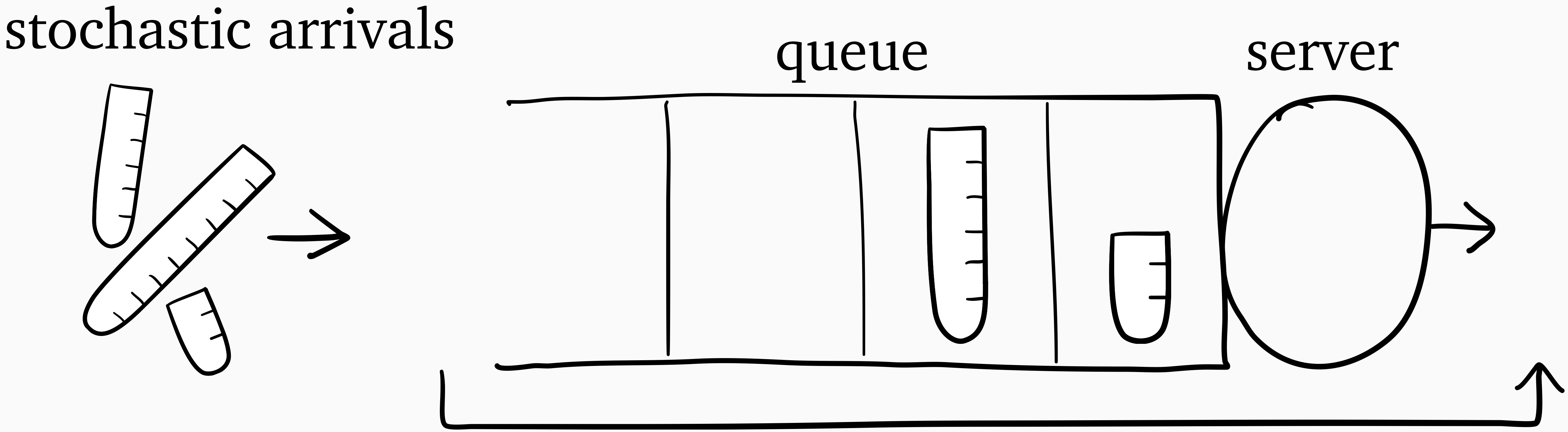


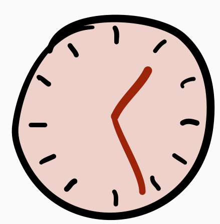
What are scheduling and delay?



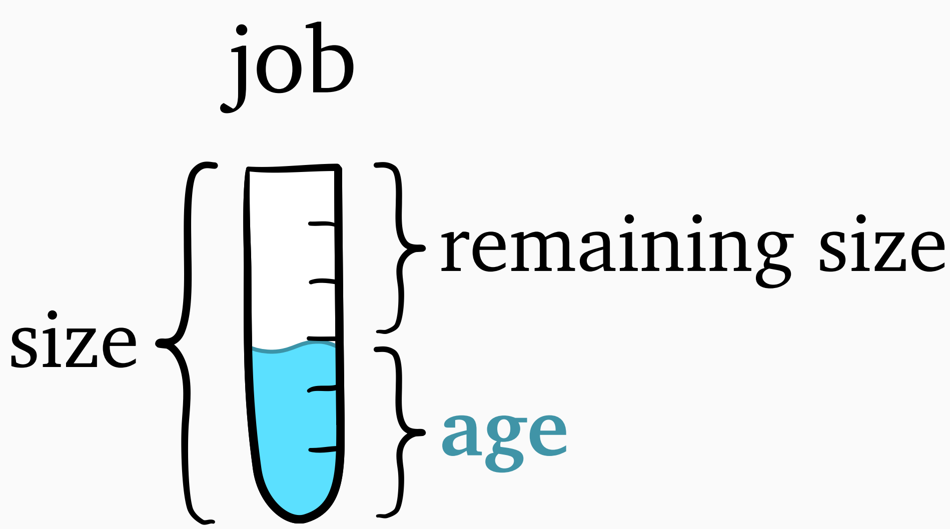


What are scheduling and delay?





“delay” = *response time*

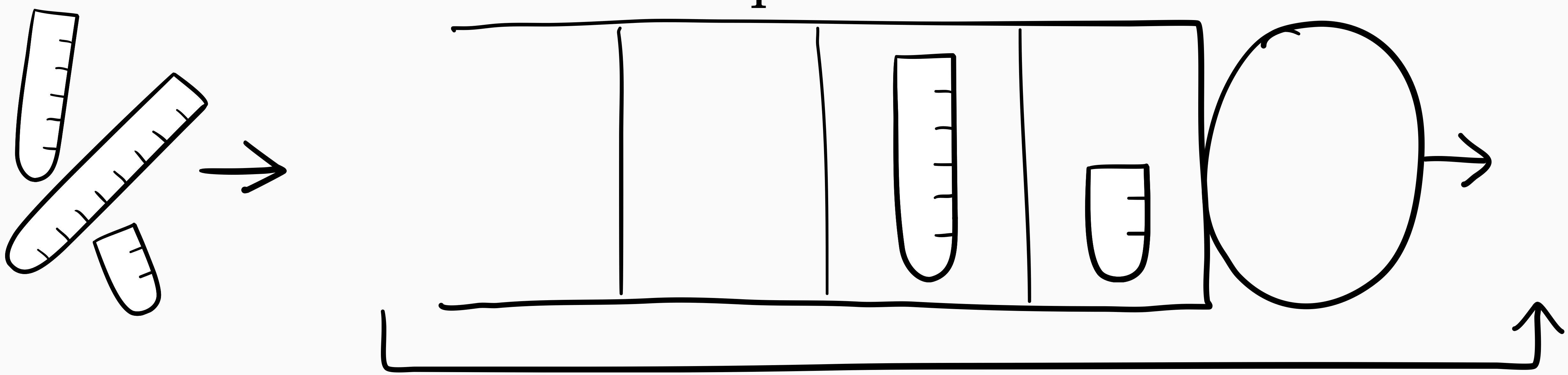


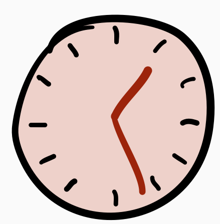
What are scheduling and delay?

stochastic arrivals

queue

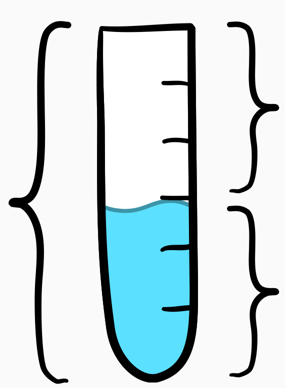
server





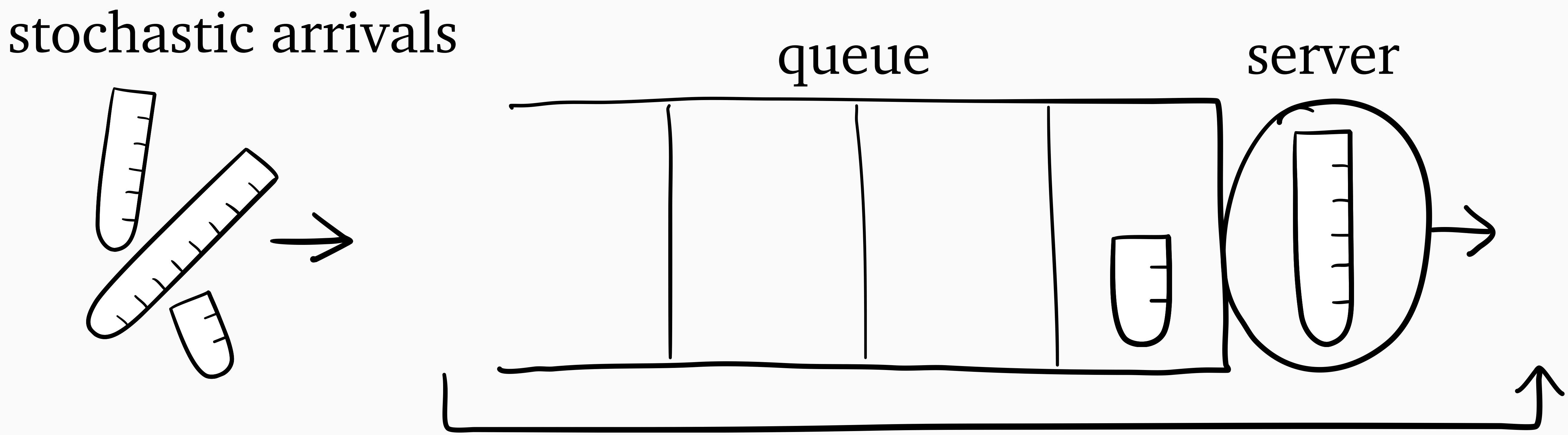
"delay" = *response time*

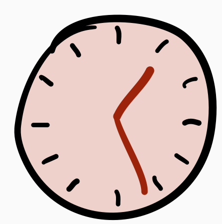
job



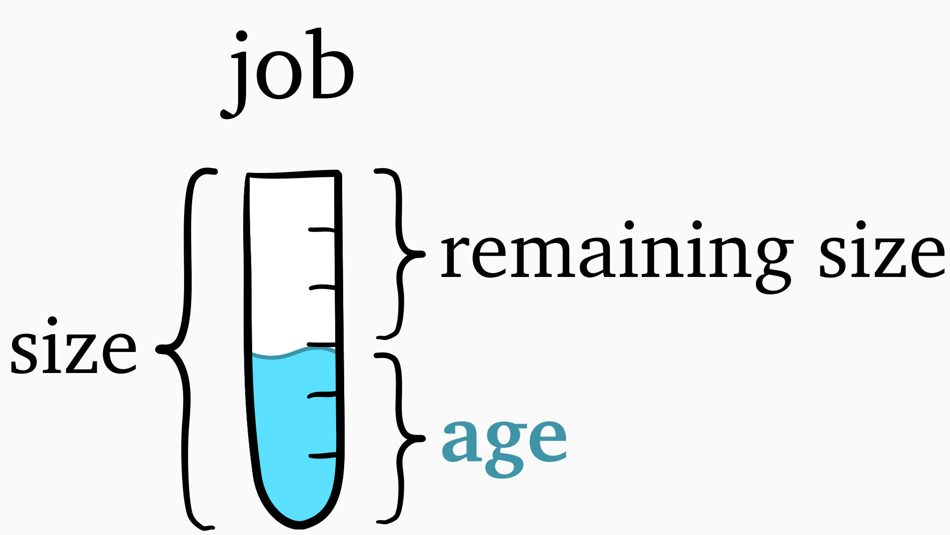
Scheduling policy:
decides which job to serve

What are scheduling and delay?



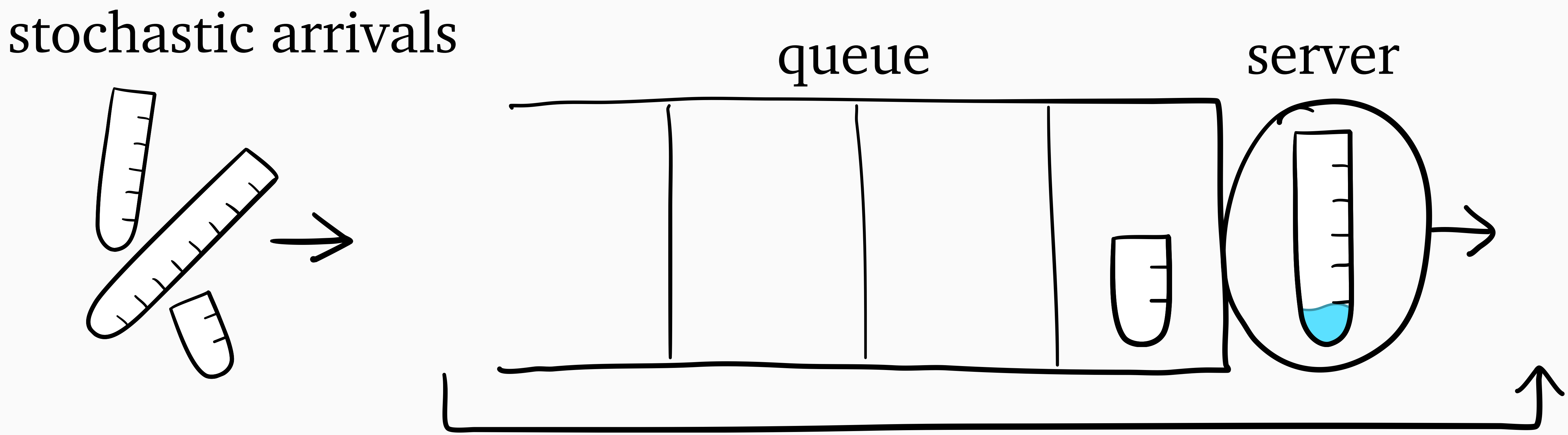


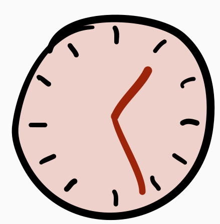
“delay” = *response time*



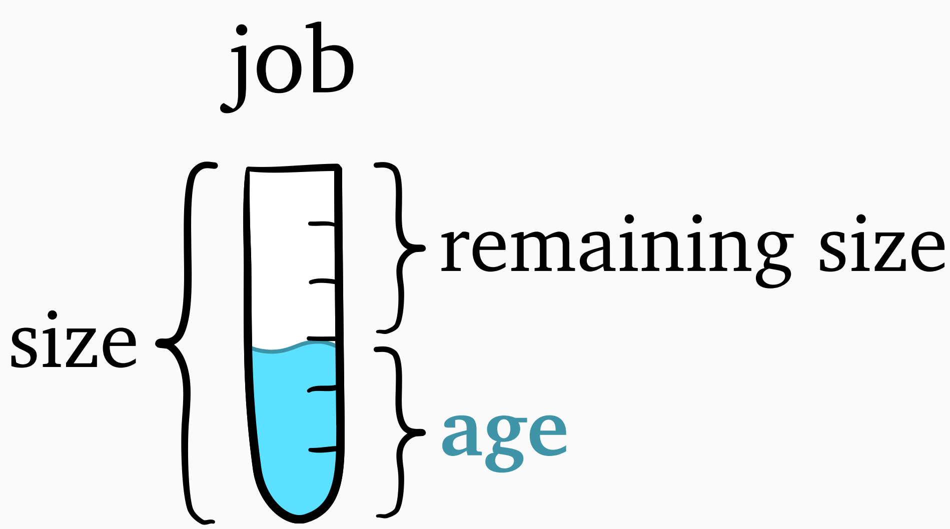
Scheduling policy:
decides which job to serve

What are scheduling and delay?



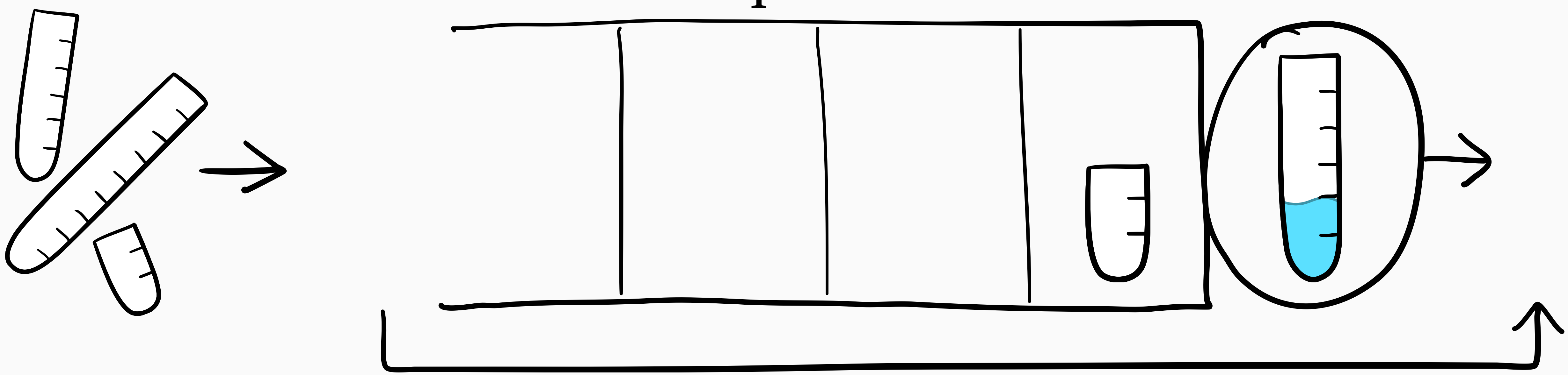


"delay" = *response time*

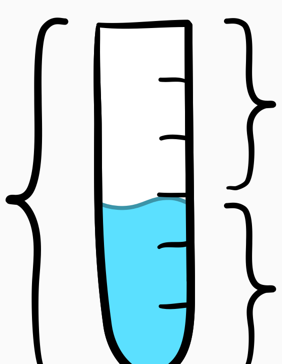


Scheduling policy:
decides which job to serve

What are scheduling and delay?

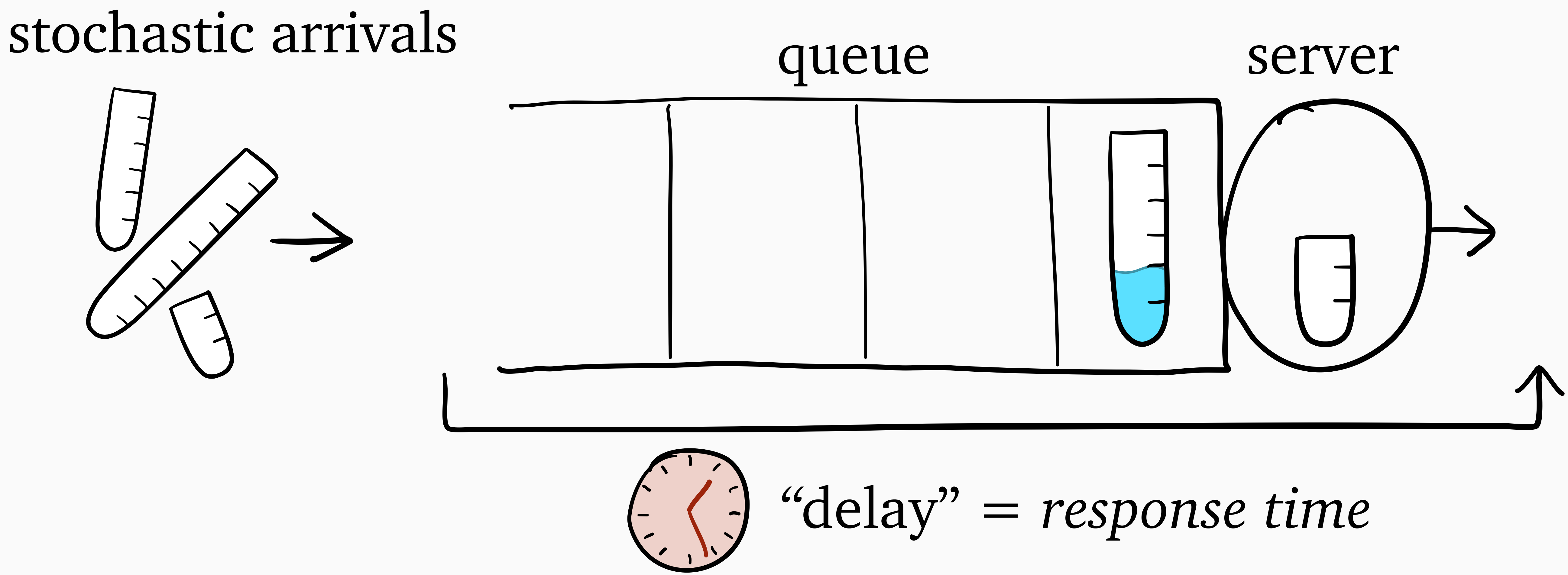


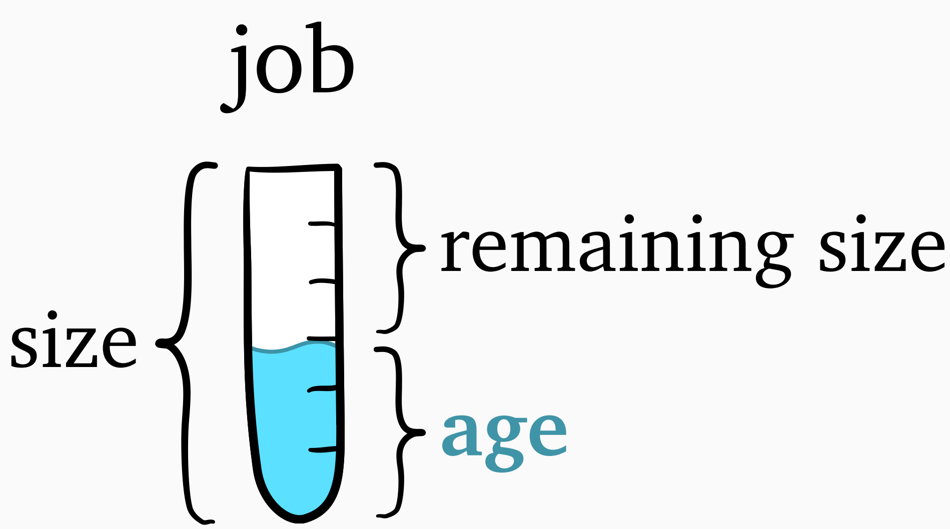
"delay" = *response time*



Scheduling policy:
decides which job to serve

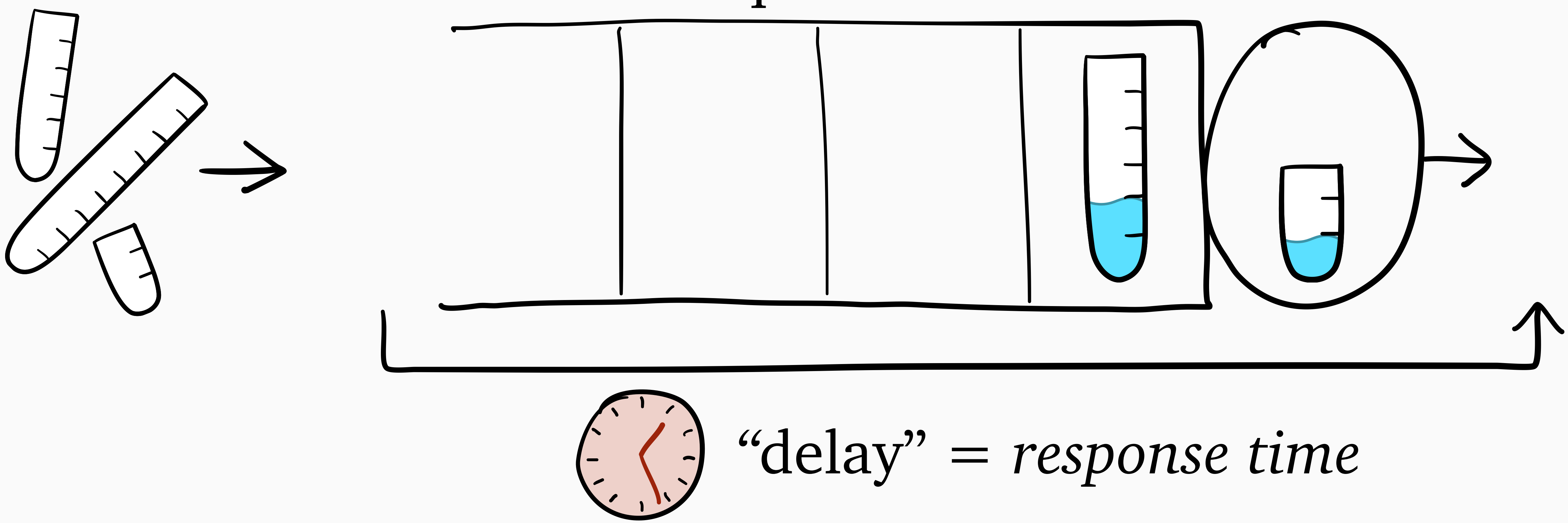
What are scheduling and delay?

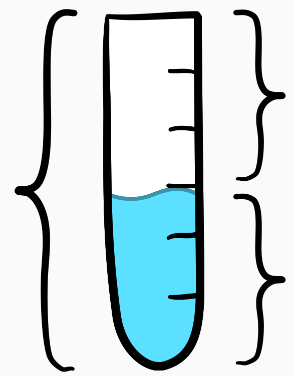




Scheduling policy:
decides which job to serve

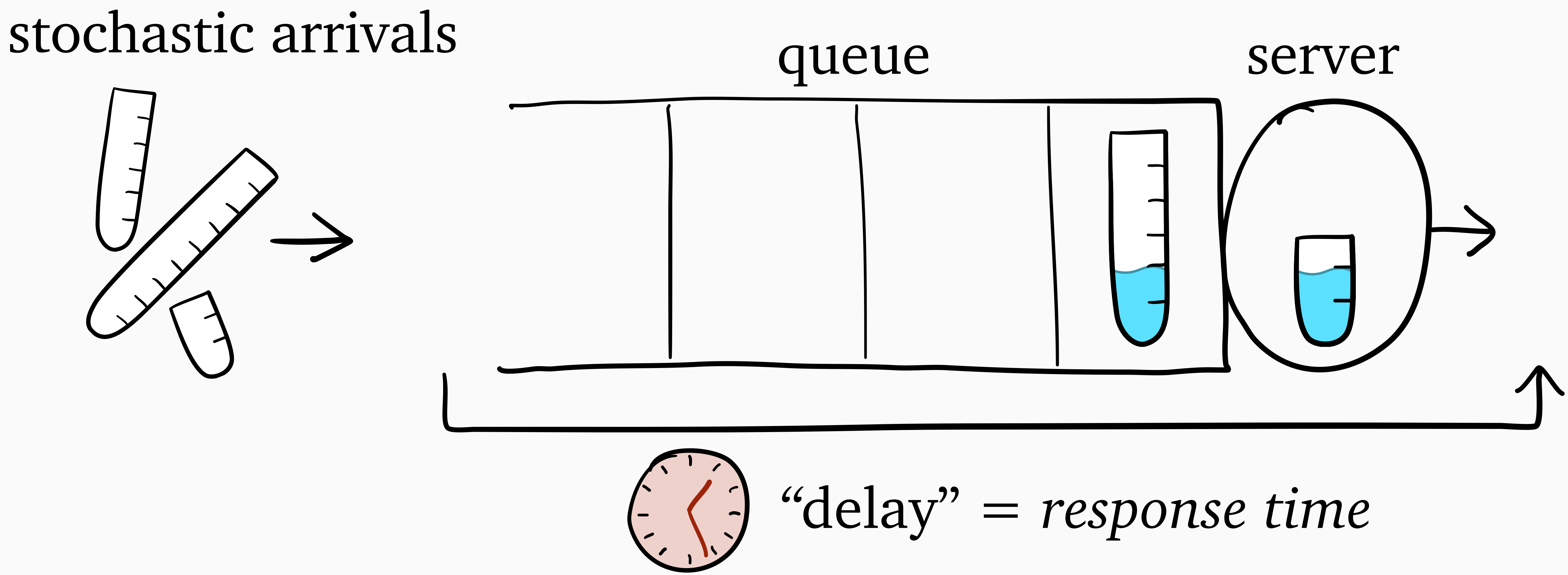
What are scheduling and delay?

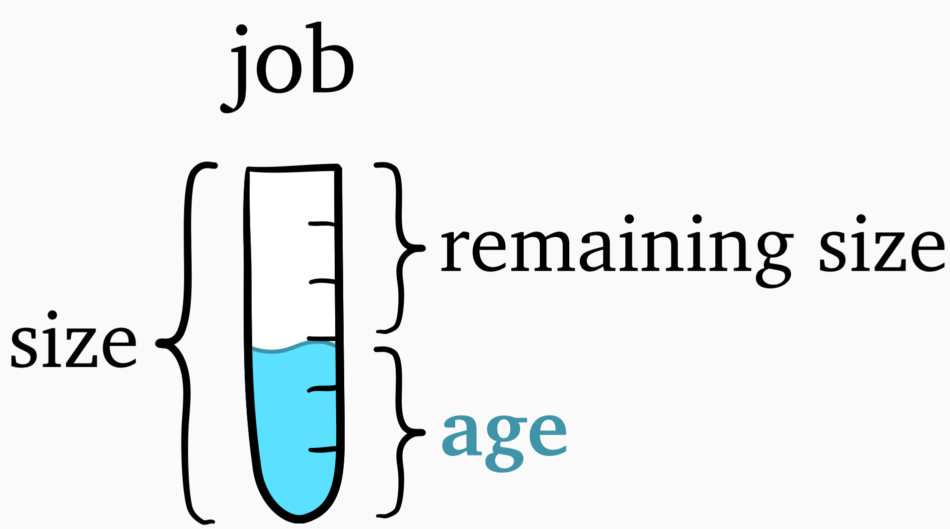




Scheduling policy:
decides which job to serve

What are scheduling and delay?



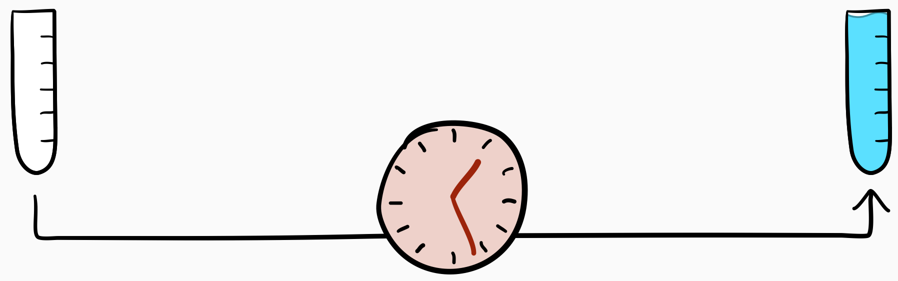


Scheduling policy:
decides which job to serve

Optimizing delay

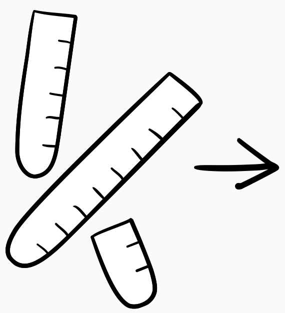
Optimizing delay

response time

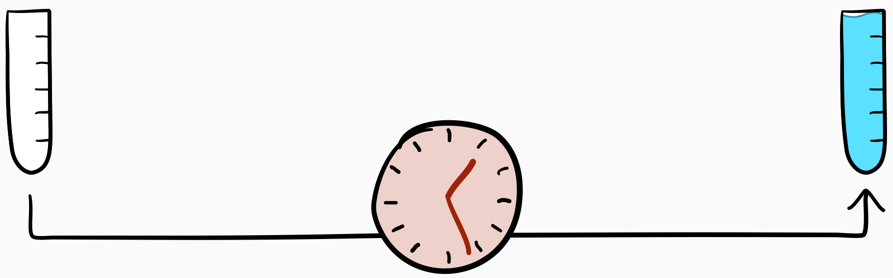


Optimizing delay

stochastic arrival process λ, S

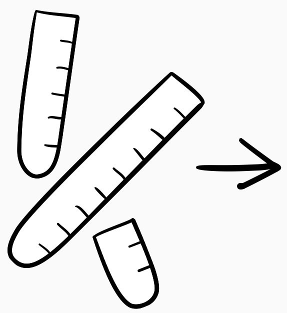


response time

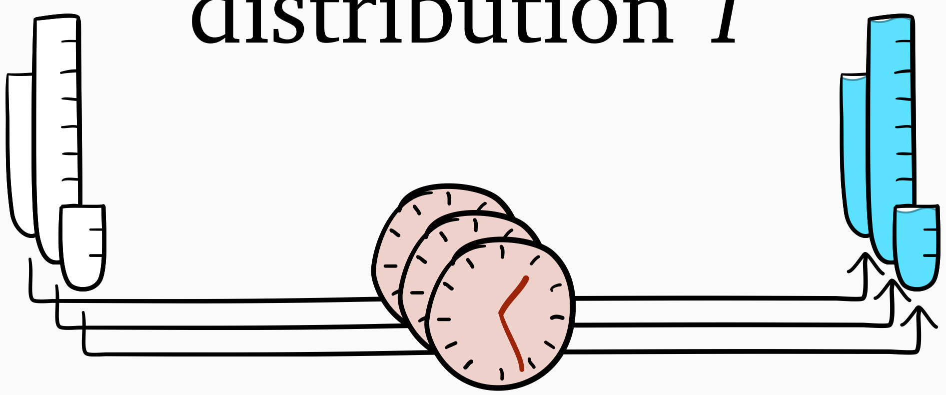


Optimizing delay

stochastic arrival
process λ, S

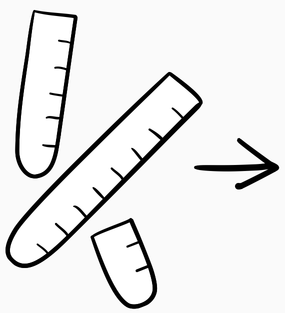


response time
distribution T



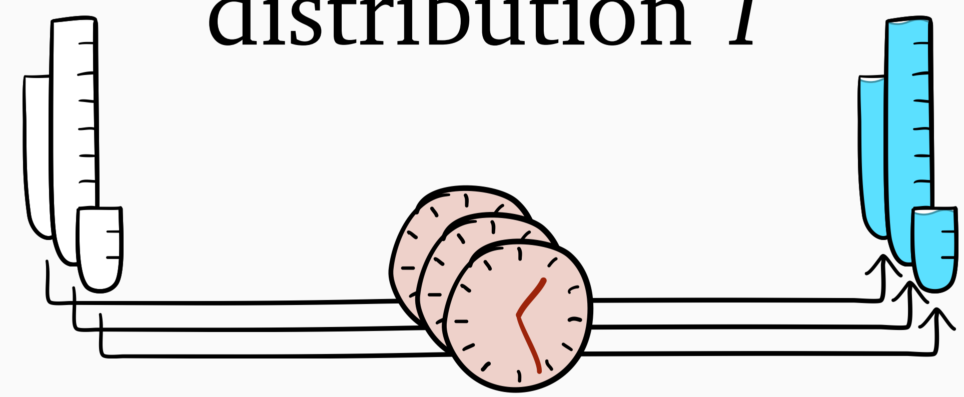
Optimizing delay

stochastic arrival
process λ, S

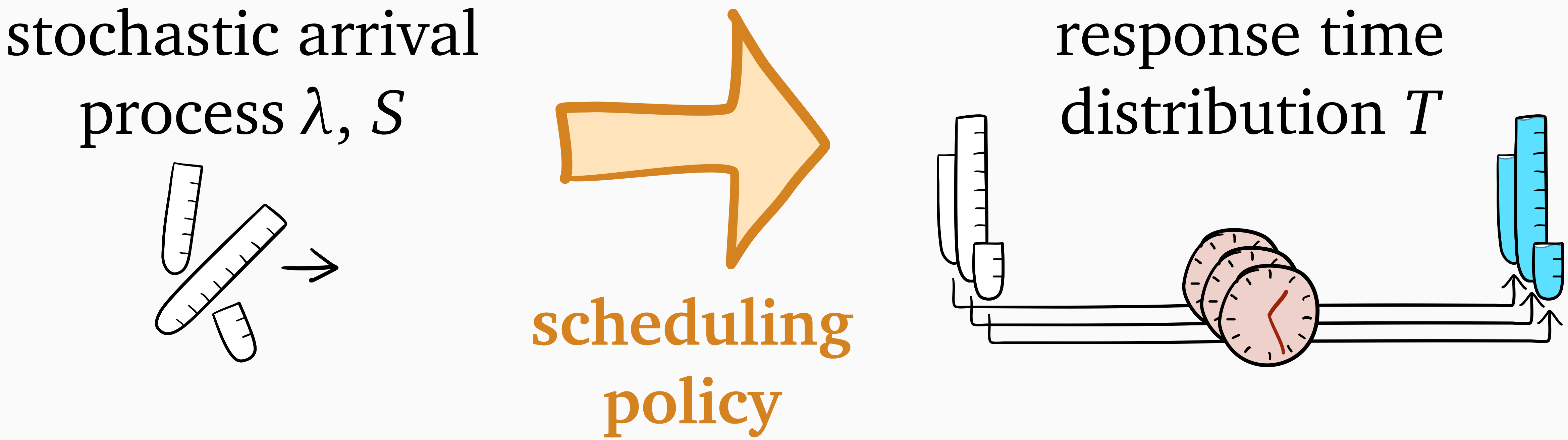


**scheduling
policy**

response time
distribution T

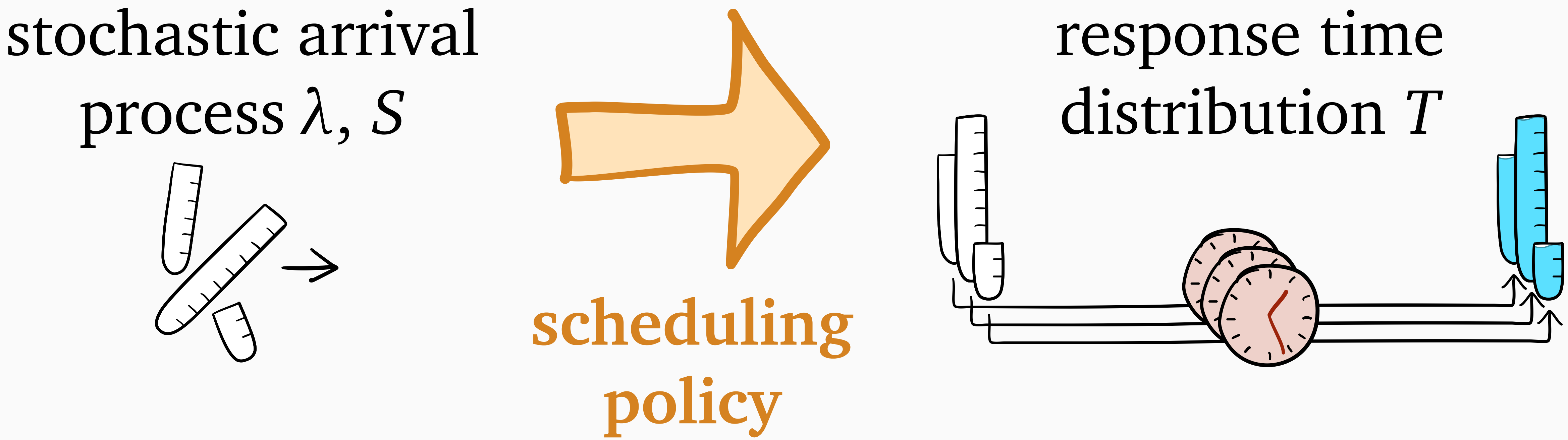


Optimizing delay



Goal: minimize *mean response time* $\mathbf{E}[T]$

Optimizing delay

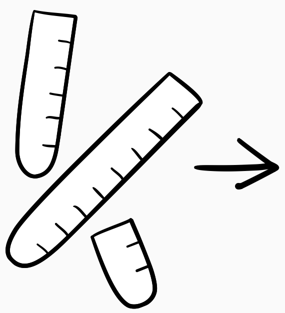


Goal: minimize *mean response time* $\mathbf{E}[T]$

Warmup: if sizes known?

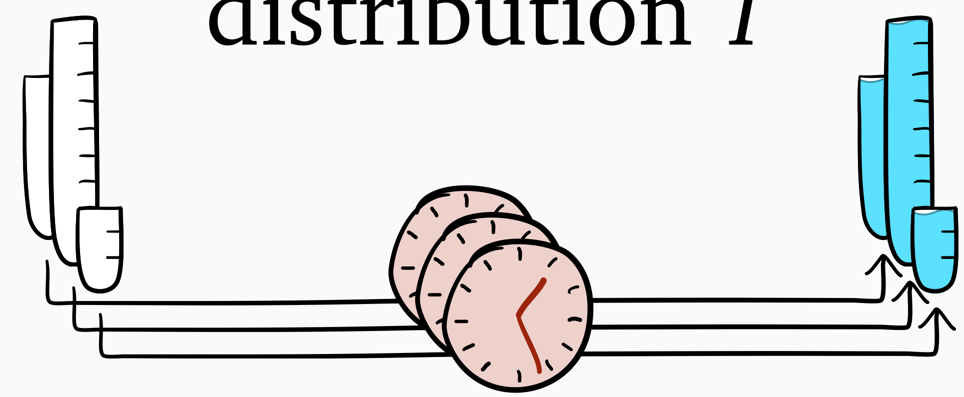
Optimizing delay

stochastic arrival process λ, S



scheduling policy

response time distribution T

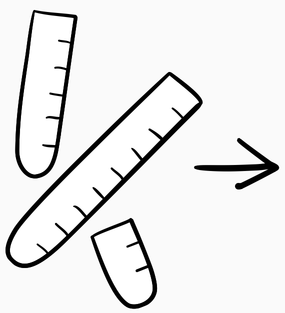


Goal: minimize *mean response time* $\mathbf{E}[T]$

Warmup: if sizes known? **SRPT**

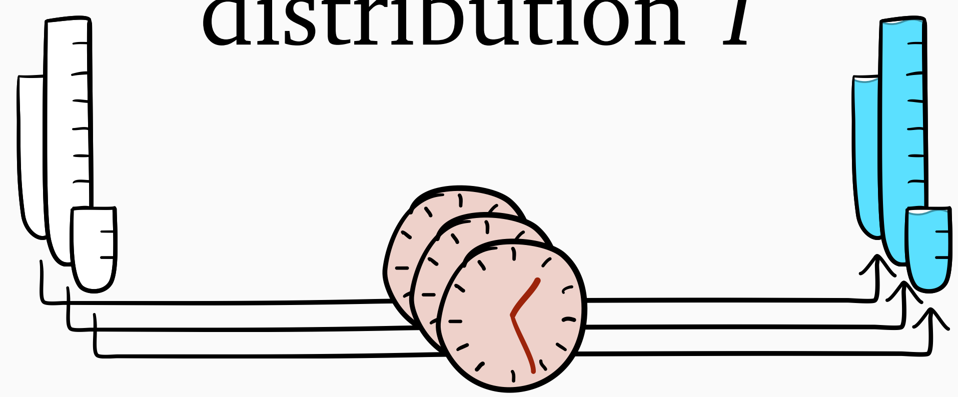
Optimizing delay

stochastic arrival
process λ, S



scheduling policy

response time
distribution T

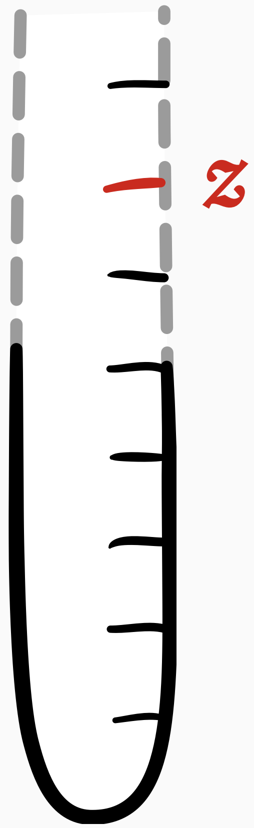


Goal: minimize *mean response time* $\mathbf{E}[T]$

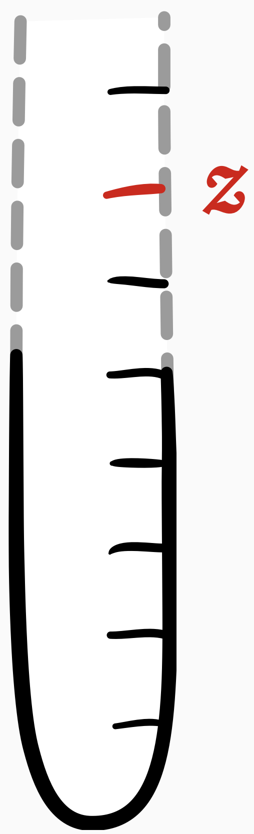
Warmup: if sizes known? **SRPT**

always serves job of *least remaining size*

What job size noise model?



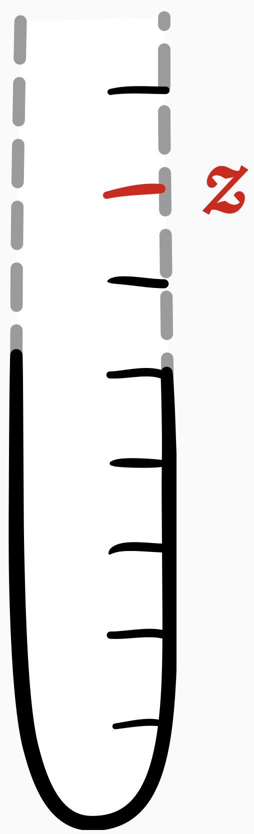
What job size noise model?

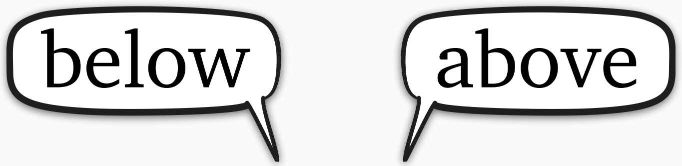


Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

What job size noise model?

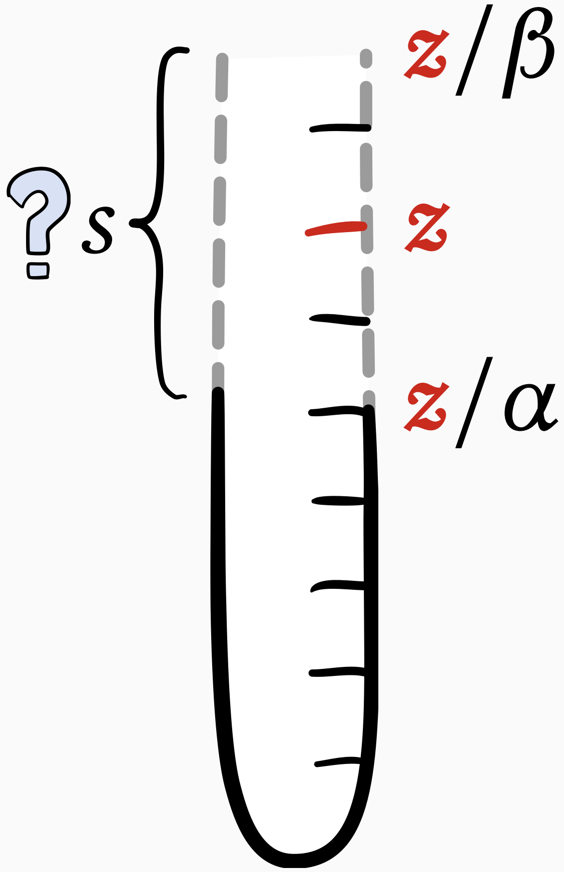




Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

What job size noise model?

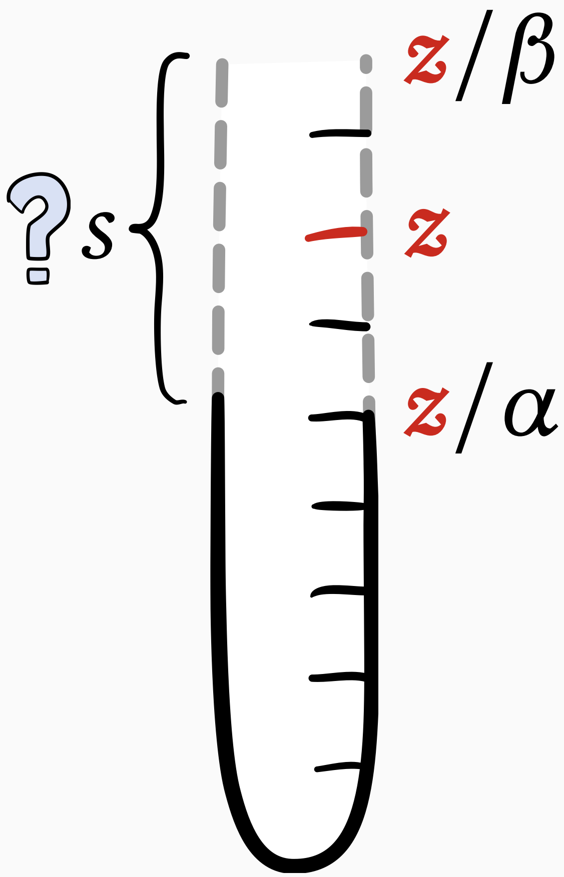


below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

What job size noise model?



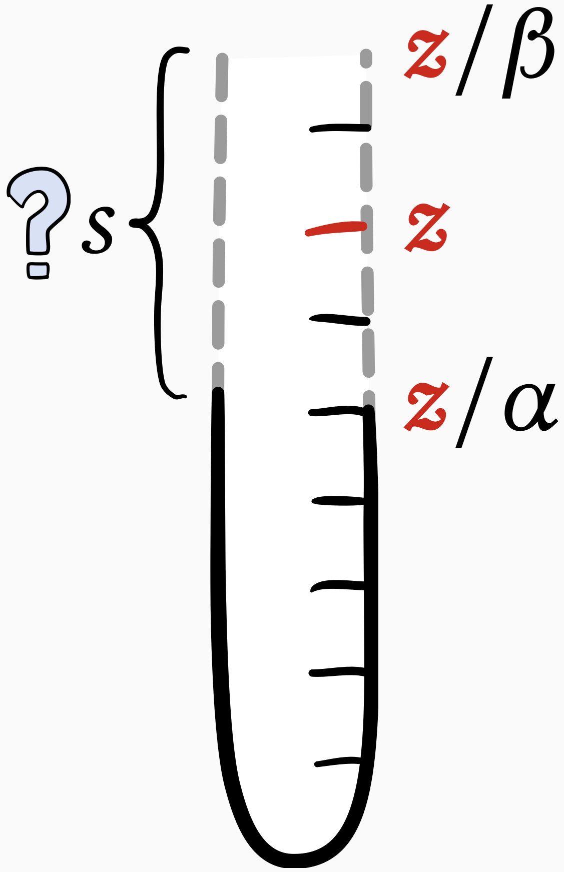
below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Generated by *joint distribution* (S, Z)

What job size noise model?



below above

Model: (β, α)-bounded noise

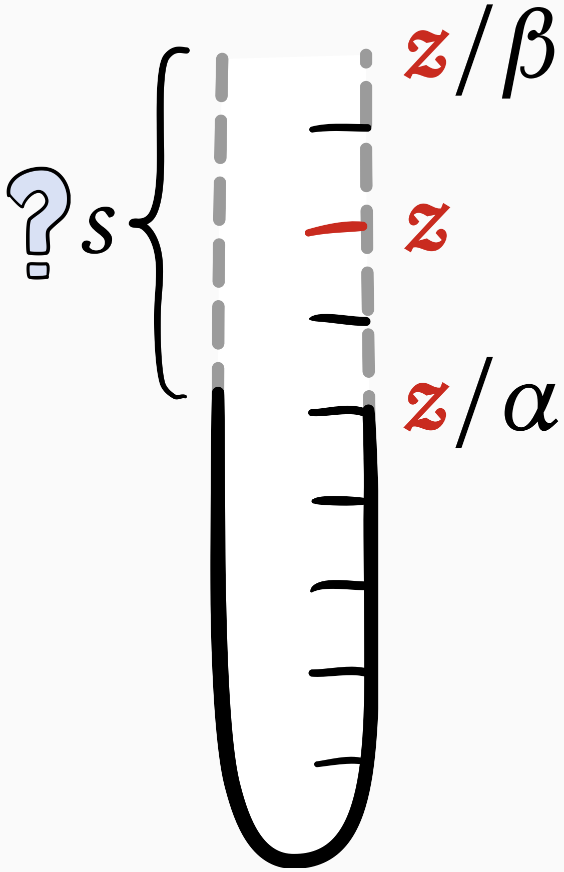
true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Generated by *joint distribution* (S, Z)

Goal: design a policy with "good" $\mathbf{E}[T]$ for

- *any* joint distribution (S, Z)
- *any* values of α, β

What job size noise model?



below above

Model: (β, α)-bounded noise

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

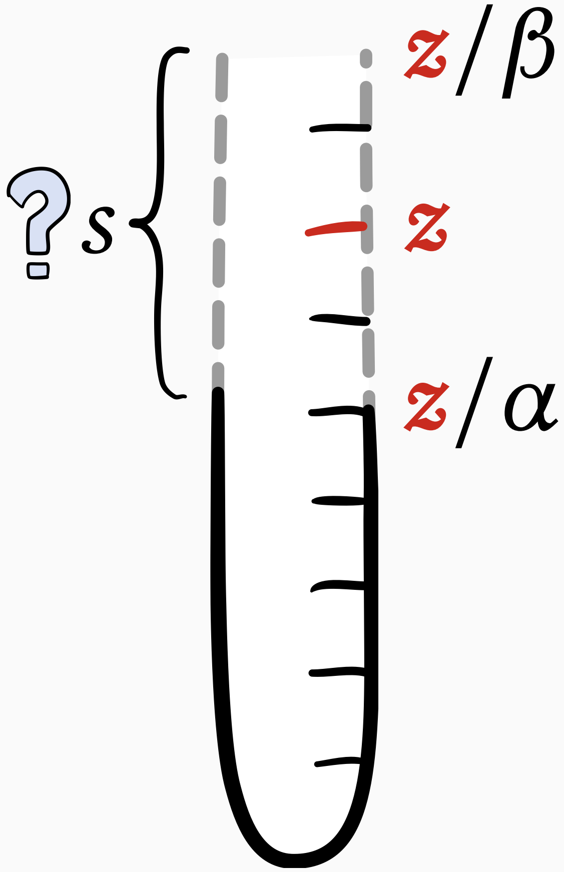
Generated by *joint distribution* (S, Z)

Goal: design a policy with "good" $\mathbf{E}[T]$ for

- *any* joint distribution (S, Z)
- *any* values of α, β

semi-adversarial

What job size noise model?



below above

Model: (β, α)-bounded n

Related: worst case
Azar, Leonardi, & Touitou
(STOC 2021, SODA 2022)

true size s $\Rightarrow$ **estimated** size $z \in [\beta s, \alpha s]$

Generated by *joint distribution* (S, Z)

Goal: design a policy with "good" $\mathbf{E}[T]$ for

- *any* joint distribution (S, Z)
- *any* values of α, β

semi-adversarial

What can we hope to achieve?

What can we hope to achieve?

C-consistent: $$\frac{\mathbf{E}[T_{P}]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \to C \qquad \text{as } \alpha, \beta \to 1$$

What can we hope to achieve?

C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C \qquad \text{as } \alpha, \beta \to 1$$

R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le R \qquad \text{for all } \alpha, \beta$$

What can we hope to achieve?

C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \to C \qquad \text{as } \alpha, \beta \to 1$$

R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \leq R \qquad \text{for all } \alpha, \beta$$

What can we hope to achieve?

C-consistent: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C$ as $\alpha, \beta \to 1$

G-graceful: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \leq G \cdot \dfrac{\alpha}{\beta}$ for all α, β

✗ R-robust: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \leq R$ for all α, β

What can we hope to achieve?

✓ C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \to C \qquad \text{as } \alpha, \beta \to 1$$

✓ G-graceful: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \le G \cdot \frac{\alpha}{\beta} \qquad \text{for all } \alpha, \beta$$

✗ R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \le R \qquad \text{for all } \alpha, \beta$$

Our contribution: *first policy* ***P*** that's consistent and graceful

- $G = 3.5$
- $C = 1$

What can we hope to achieve?

✓ C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C \qquad \text{as } \alpha, \beta \to 1$$

✓ G-graceful: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le G \cdot \frac{\alpha}{\beta} \qquad \text{for all } \alpha, \beta$$

✗ R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le R \qquad \text{for all } \alpha, \beta$$

Our contribution: *first policy* ***P*** NEW!
that's consistent and graceful

- $G = 3.5$
- NEW! $C = 1$

What can we hope to achieve?

✓ C-consistent: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C$ as $\alpha, \beta \to 1$

✓ G-graceful: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le G \cdot \dfrac{\alpha}{\beta}$ for all α, β

✗ R-robust: $\dfrac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le R$ for all α, β

Our ... *first policy* P NEW!

that ... graceful

impossible in worst case
Azar, Leonardi, & Touitou
(STOC 2021)

- $G = $ ...
- NEW! $C = 1$

Our contribution: *first policy* $\boldsymbol{P}$ that's consistent and graceful

$C = 1$

$G = 3.5$

Our contribution: *first policy* $\boldsymbol{P}$ that's consistent and graceful

$C = 1$

$G = 3.5$

What is the new policy?

Our contribution: *first policy* ***P***
that's consistent and graceful

$C = 1$ $G = 3.5$

What is the new policy?

How do we bound its performance?

Our contribution: *first* policy P
that's consistent and graceful

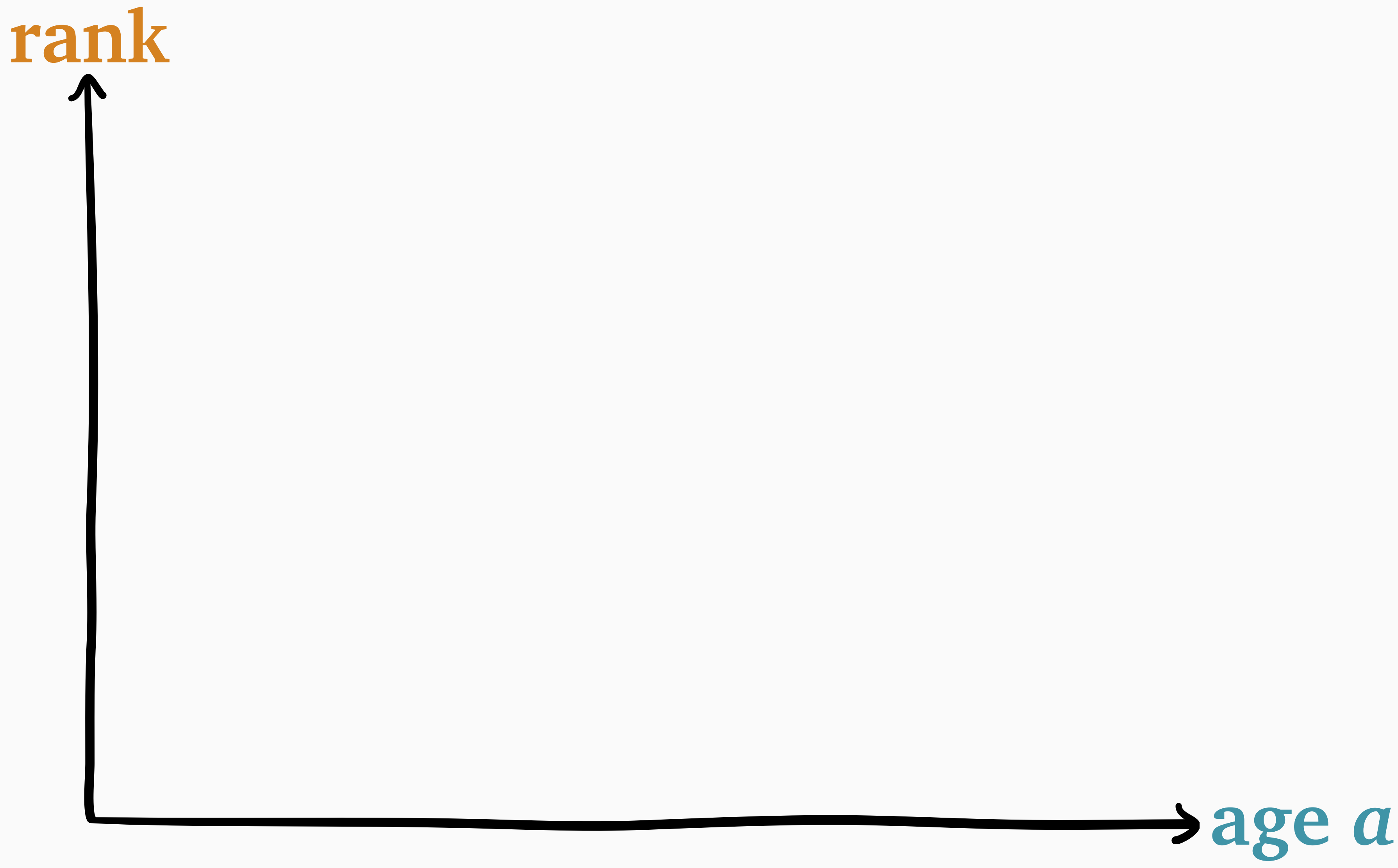
$C = 1$ $G = 3.5$

What does "policy" mean?

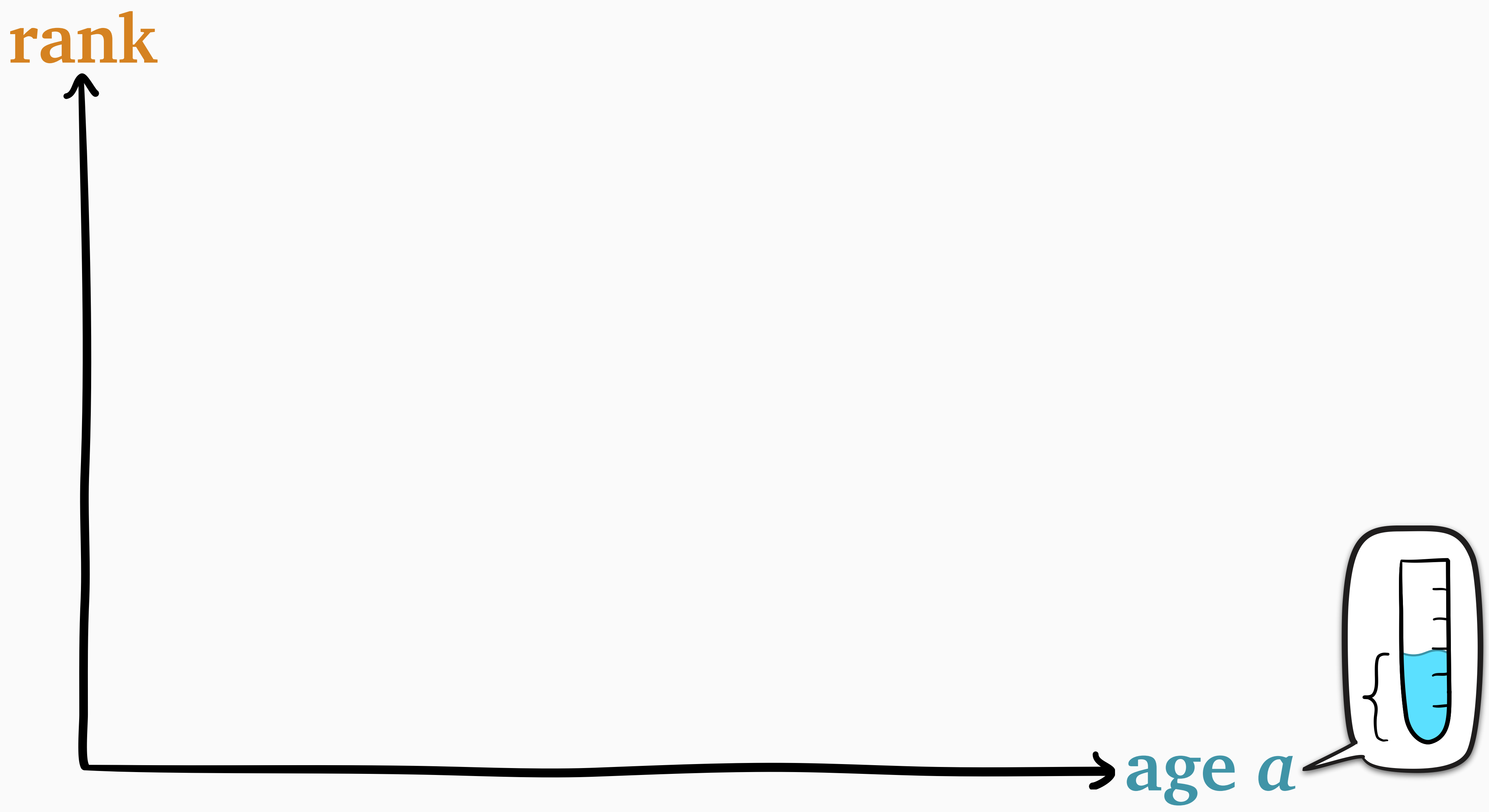
What is the new policy?

How do we bound its performance?

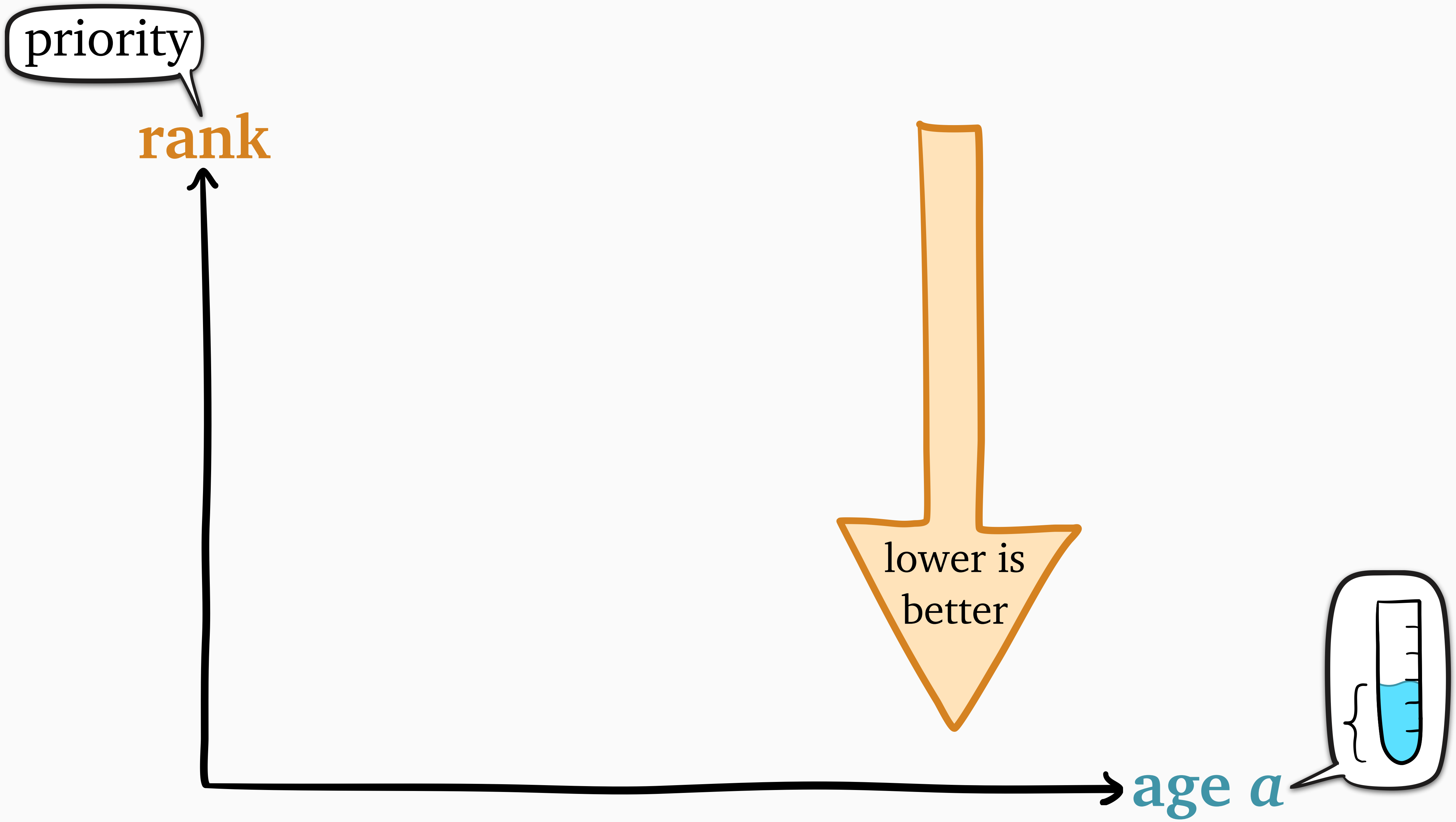
Scheduling with **rank** functions



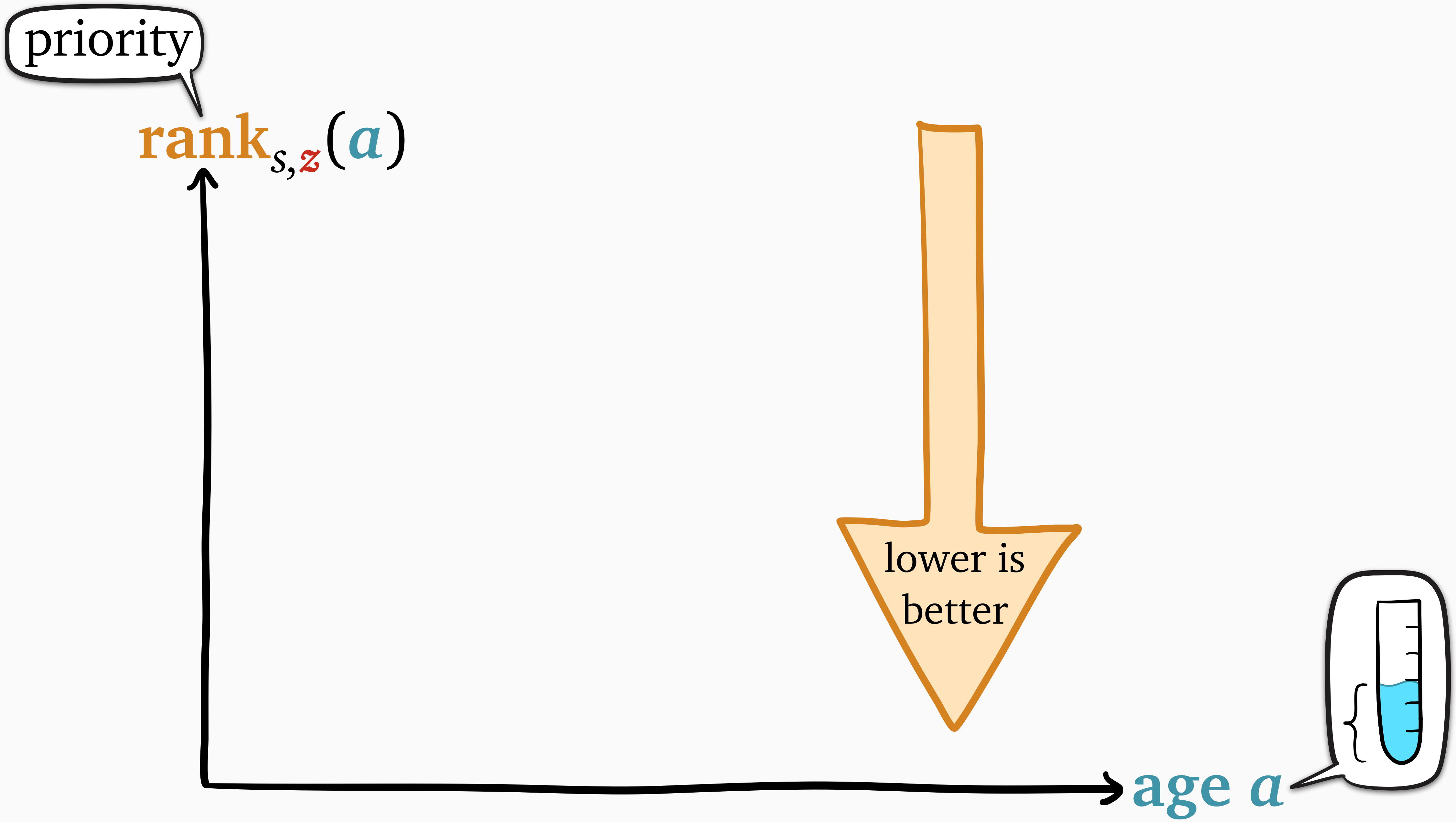
Scheduling with **rank** functions



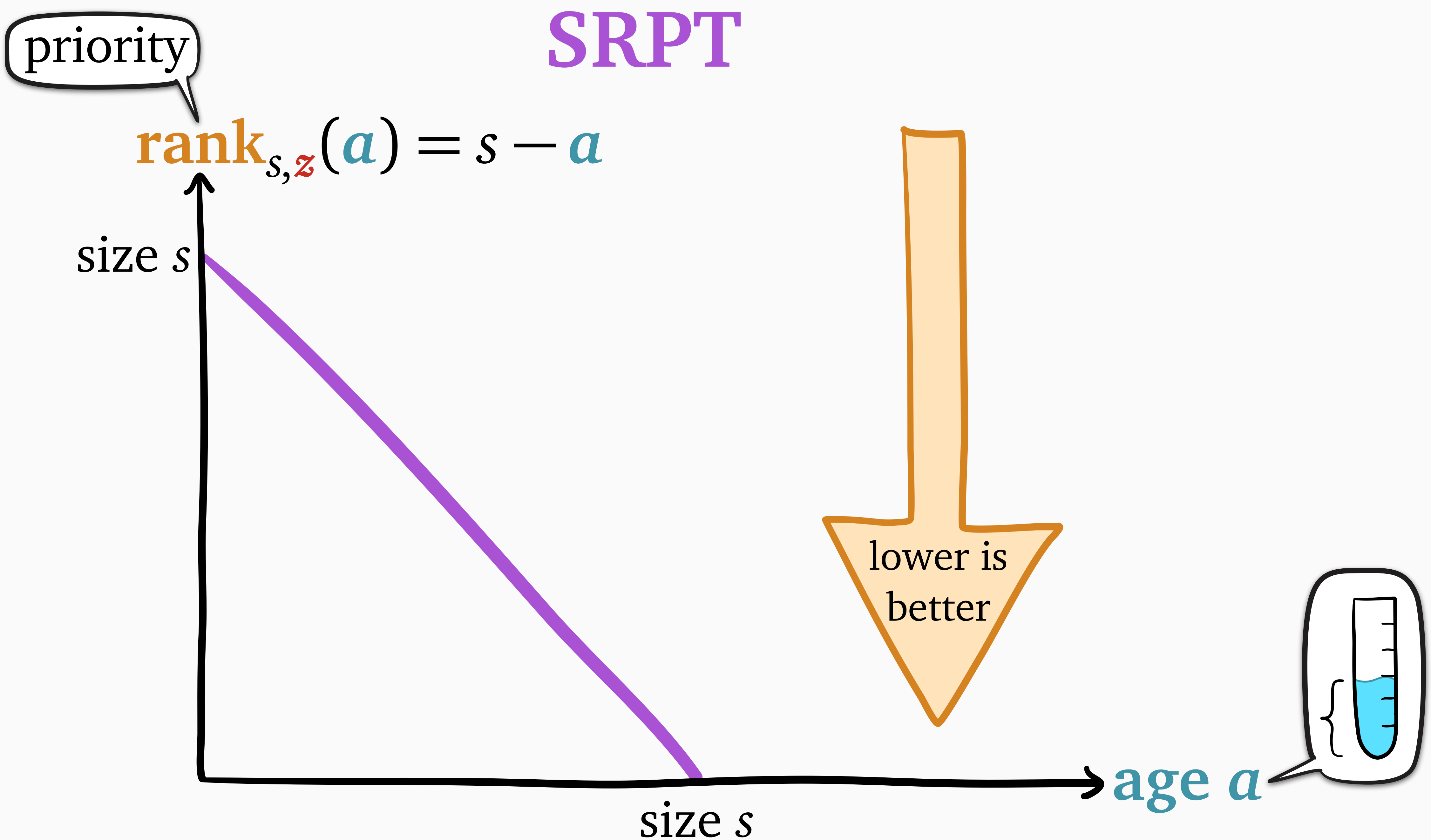
Scheduling with rank functions



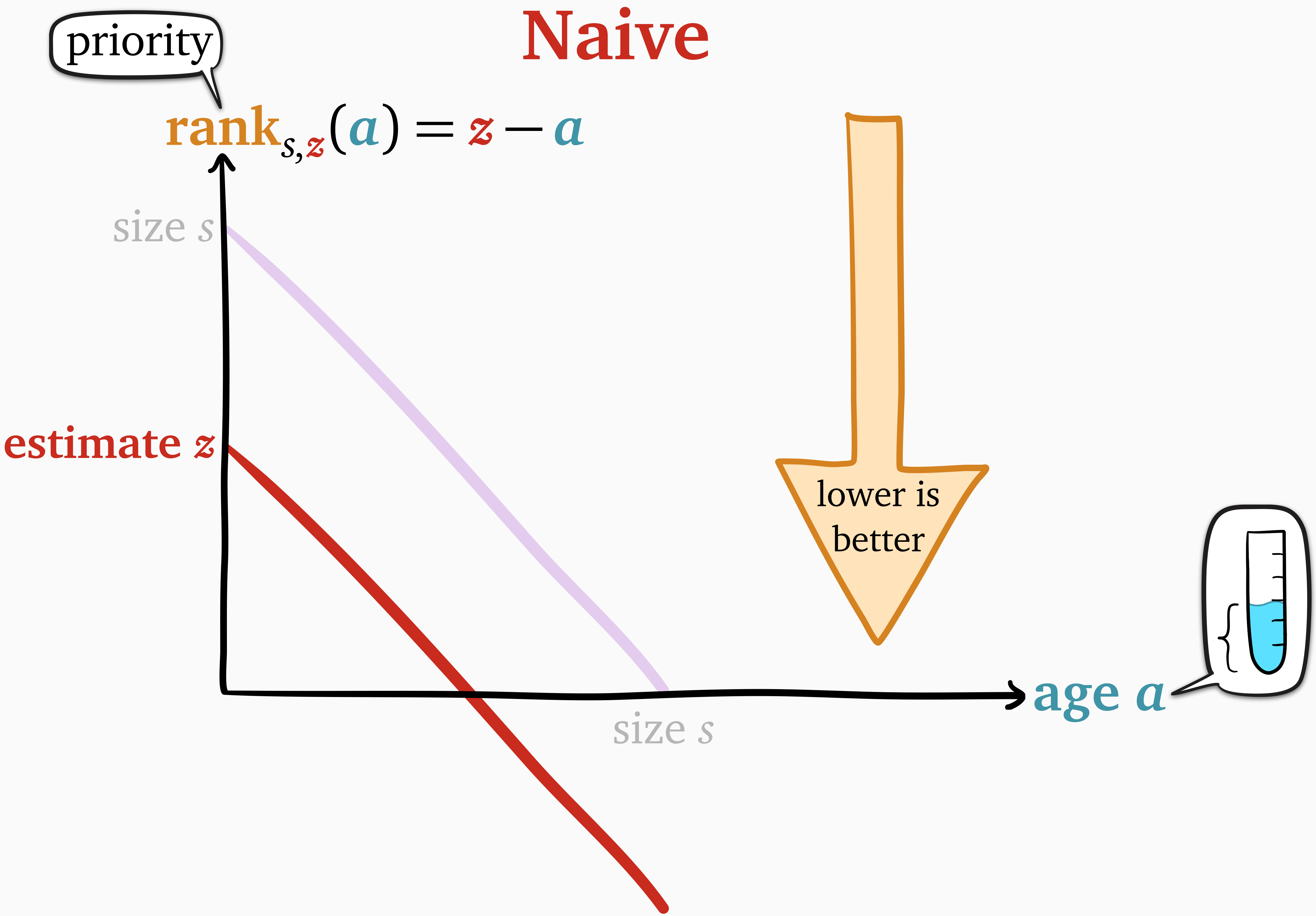
Scheduling with **rank** functions

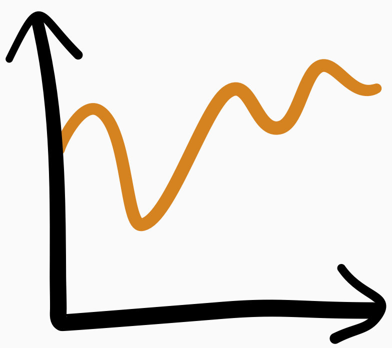


Scheduling with **rank** functions



Scheduling with rank functions

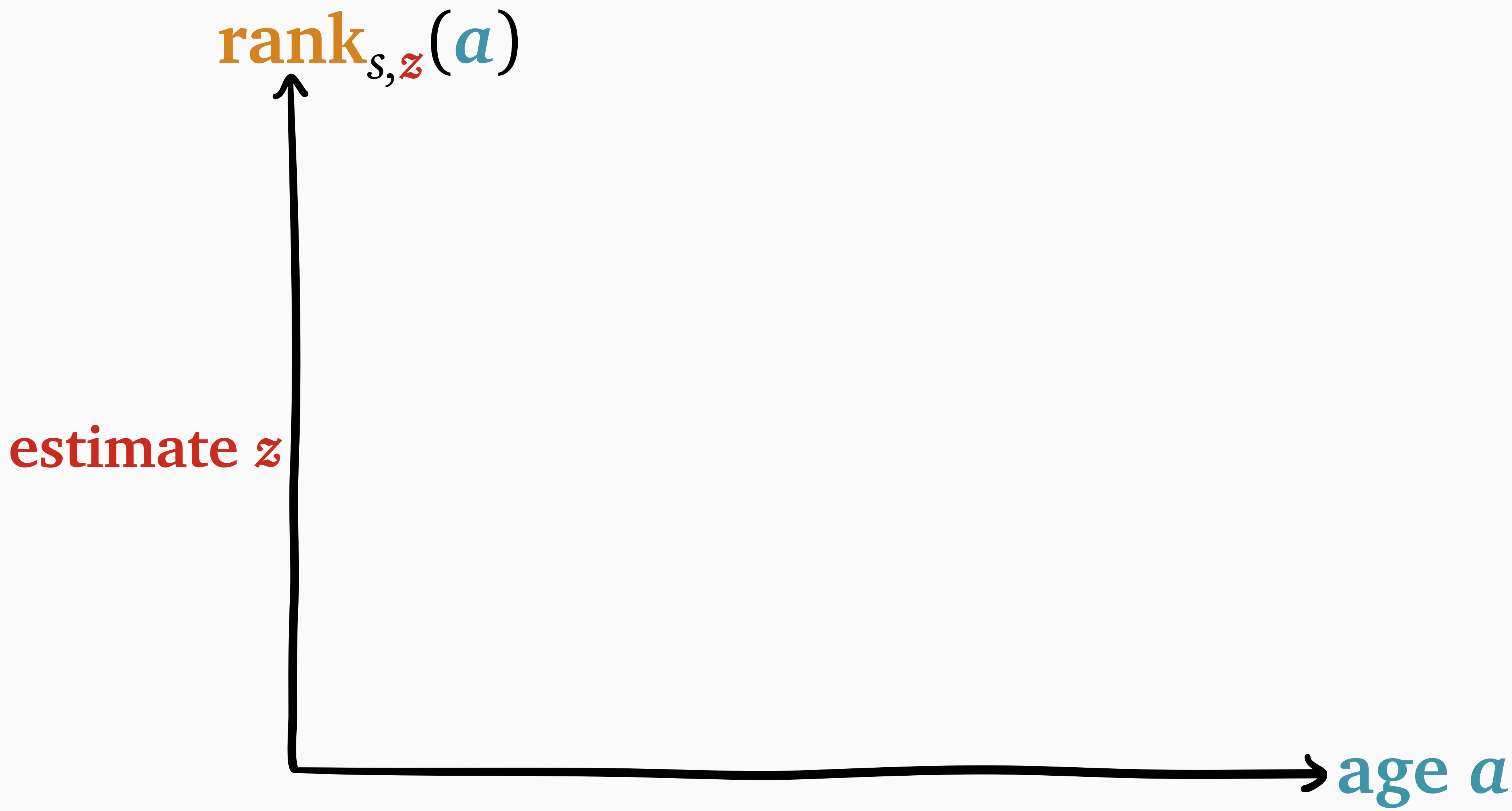




Policy design space:

rank functions

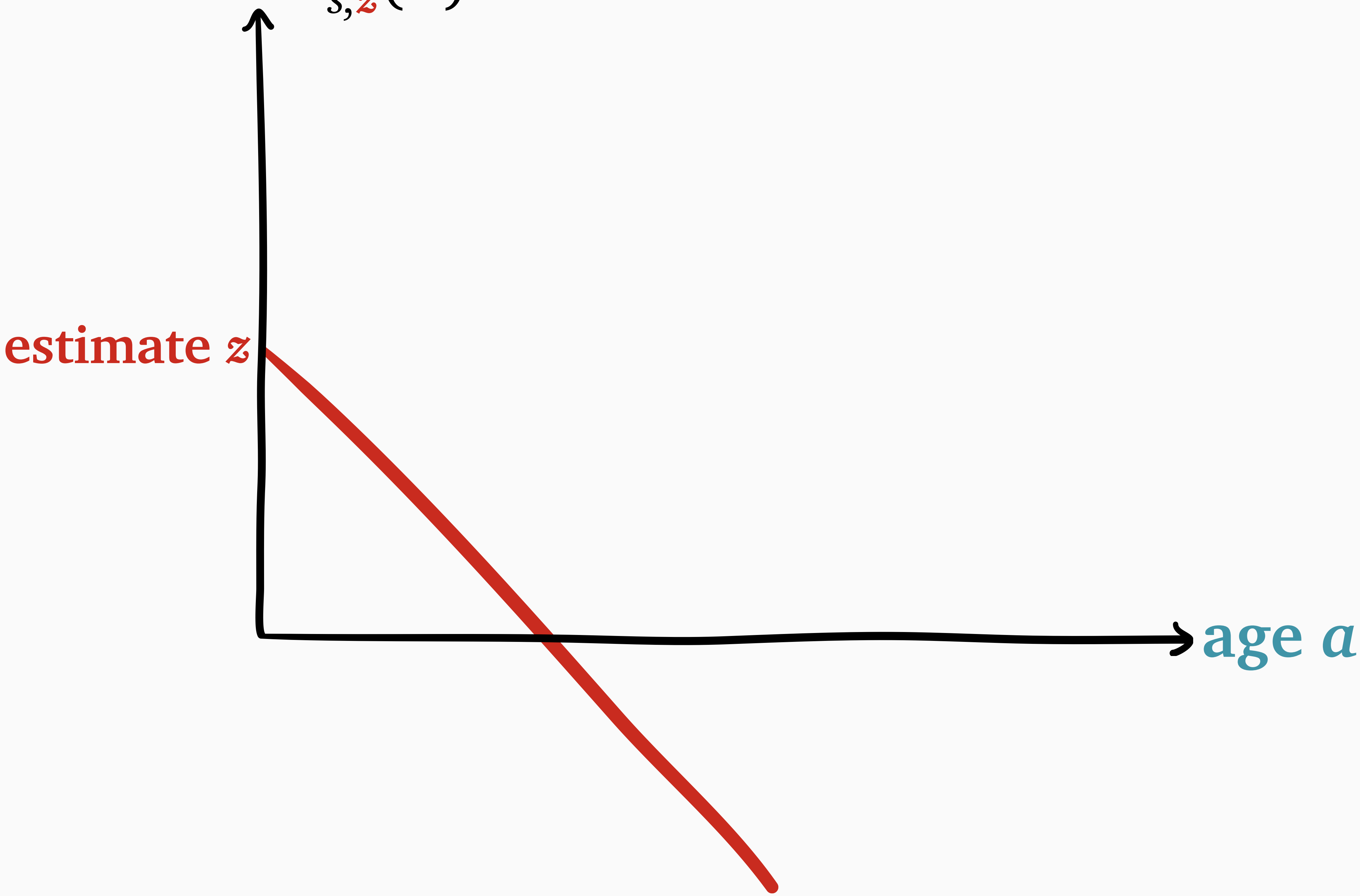
What's the right **rank** function?



What's the right **rank** function?

Naive

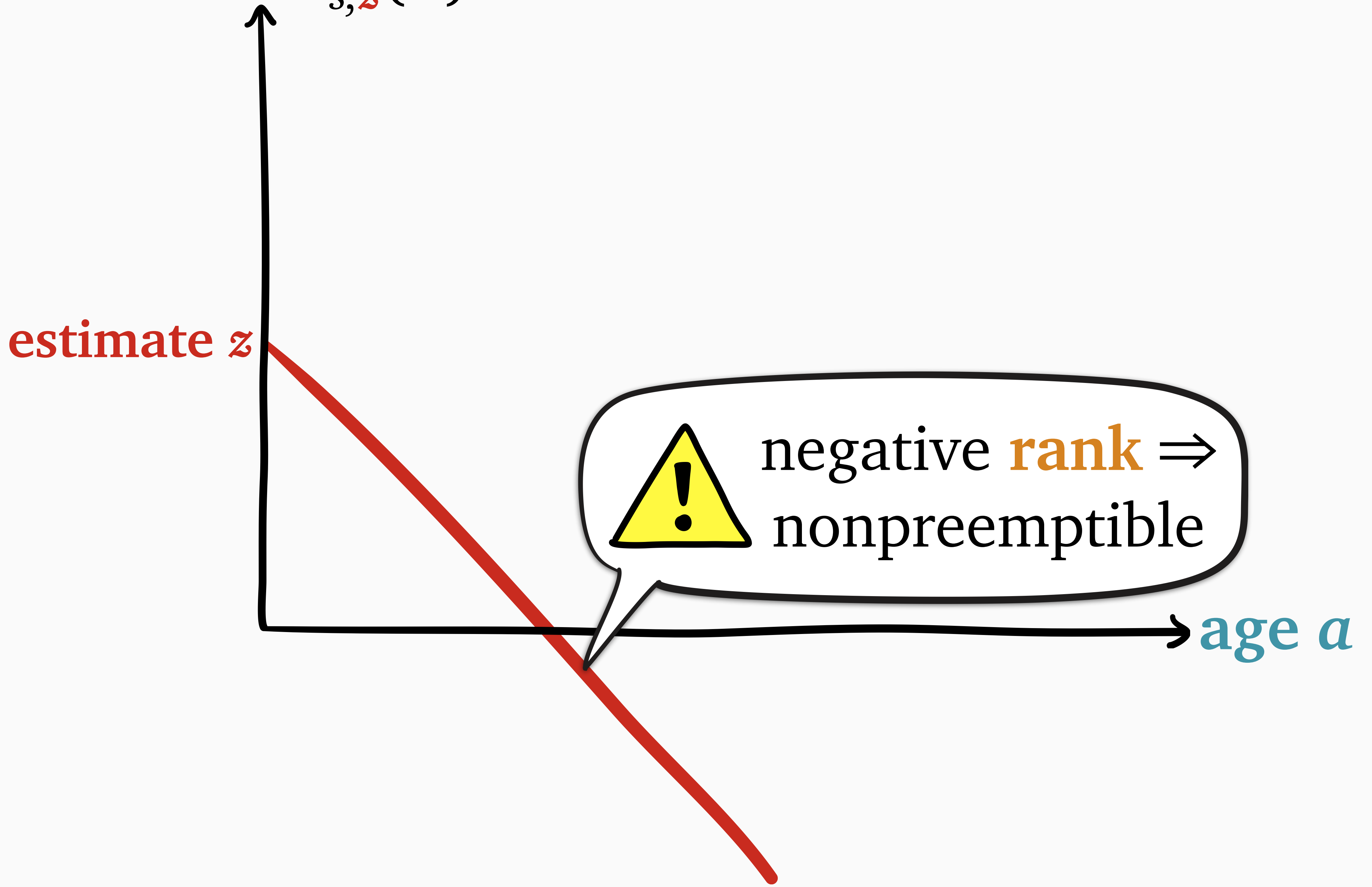
$\text{rank}_{s,z}(a) = z - a$



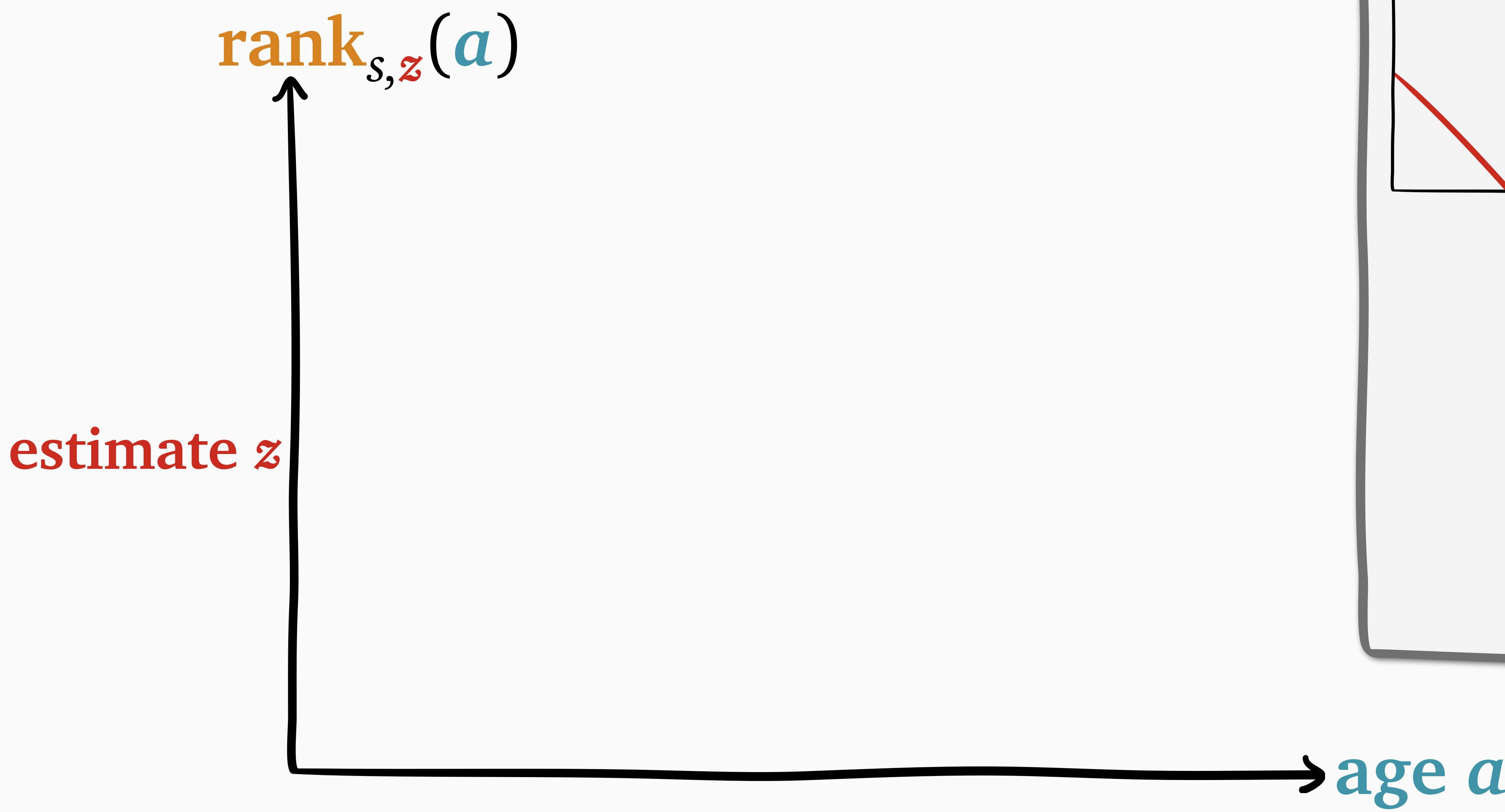
What's the right rank function?

Naive

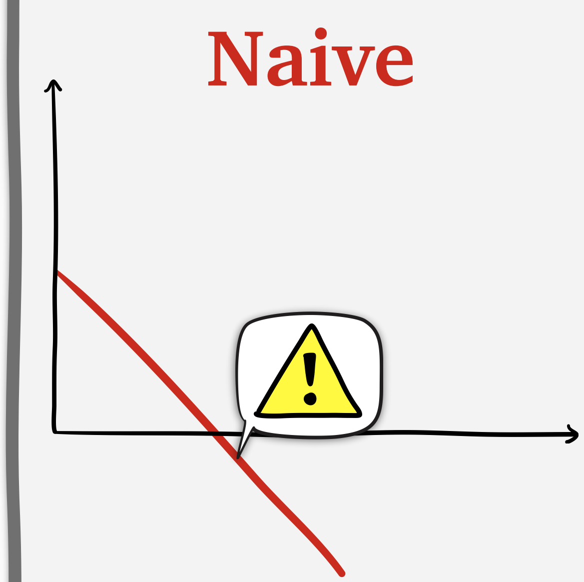
$\mathrm{rank}_{s,z}(a) = z - a$



What's the right **rank** function?



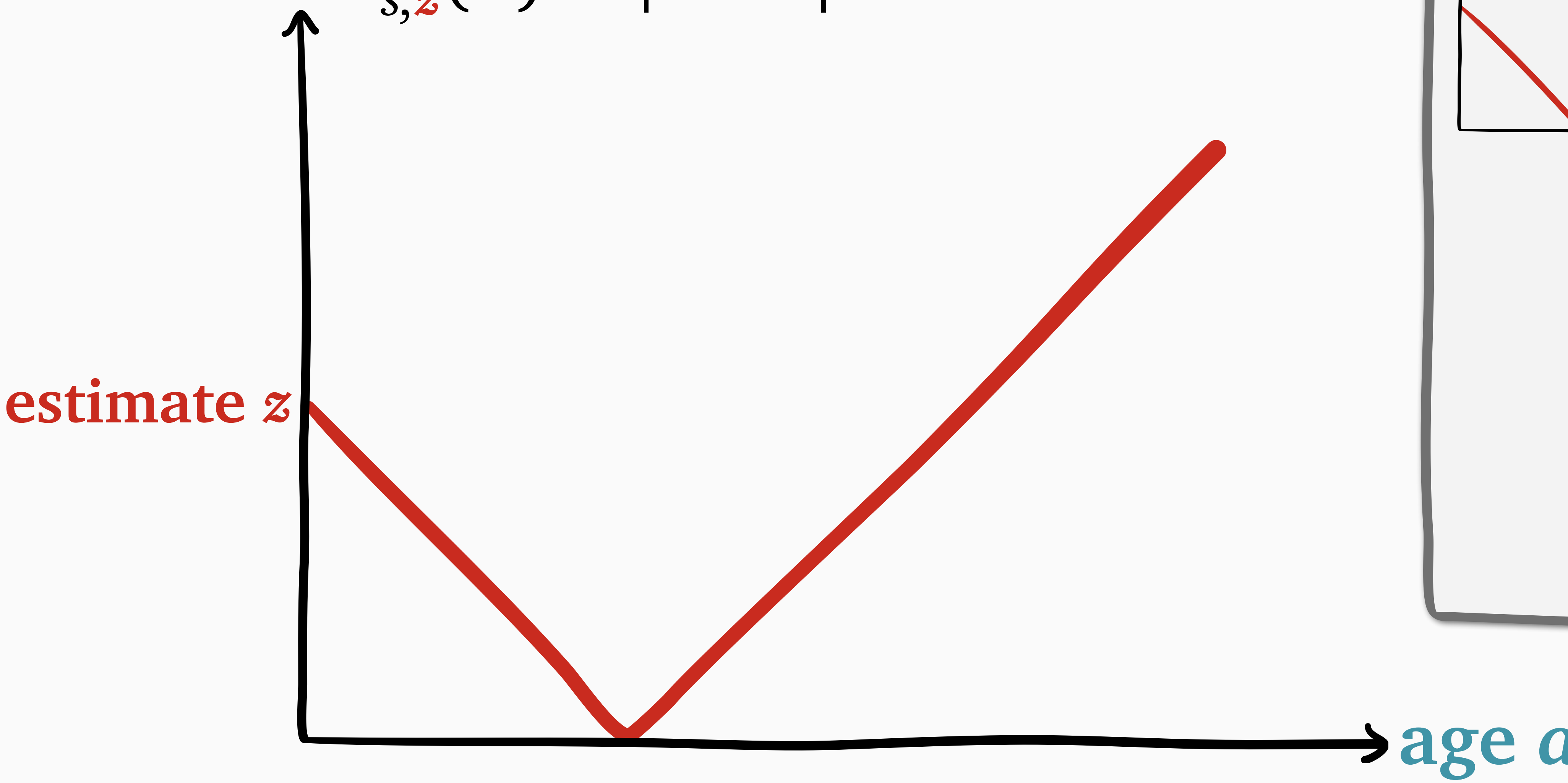
Not these…



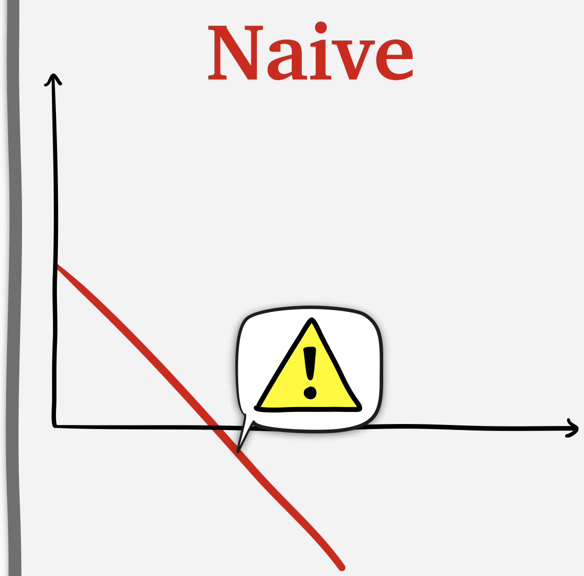
What's the right rank function?

Checkmark

$\text{rank}_{s,z}(a) = |z - a|$



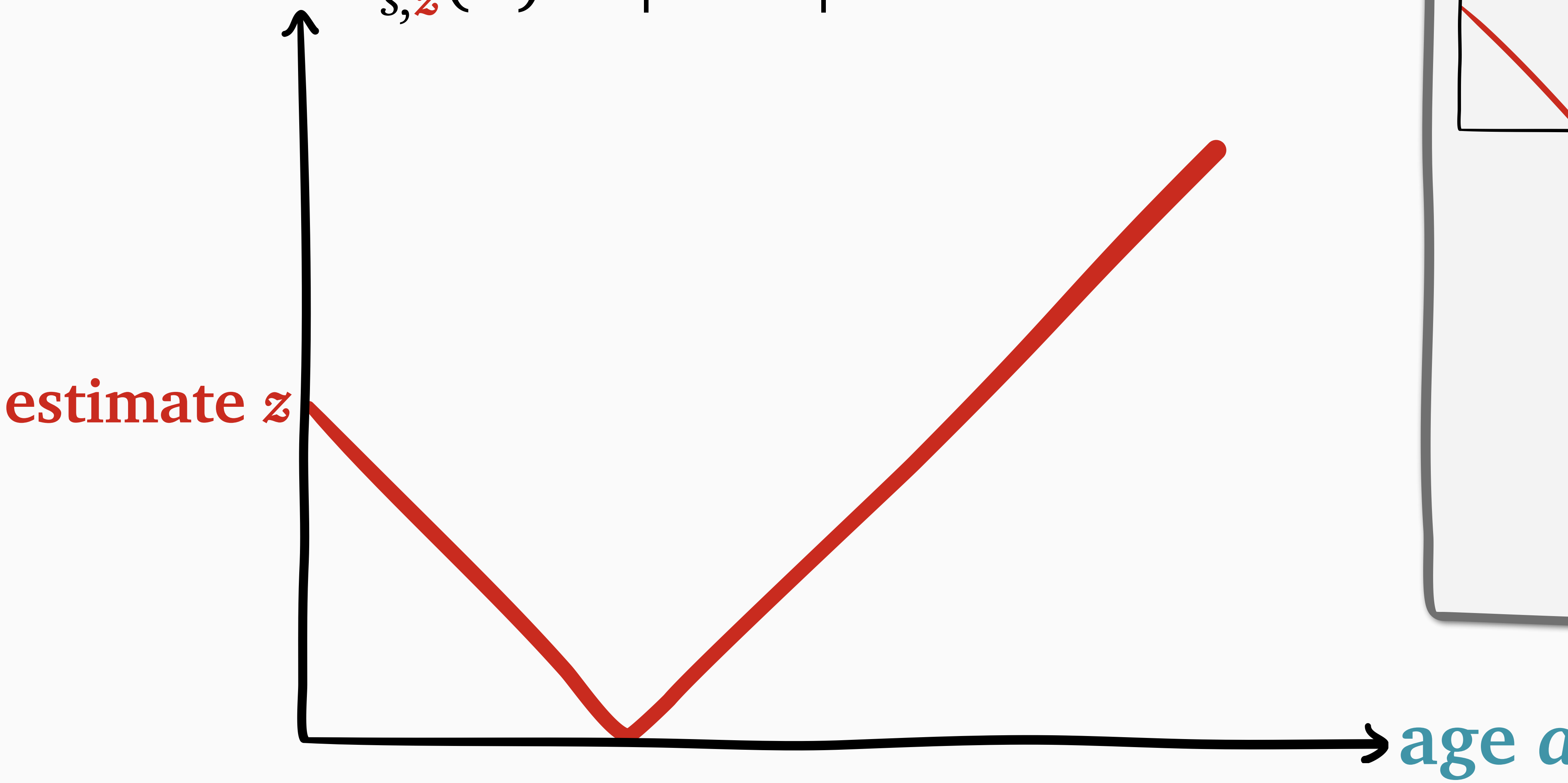
Not these…

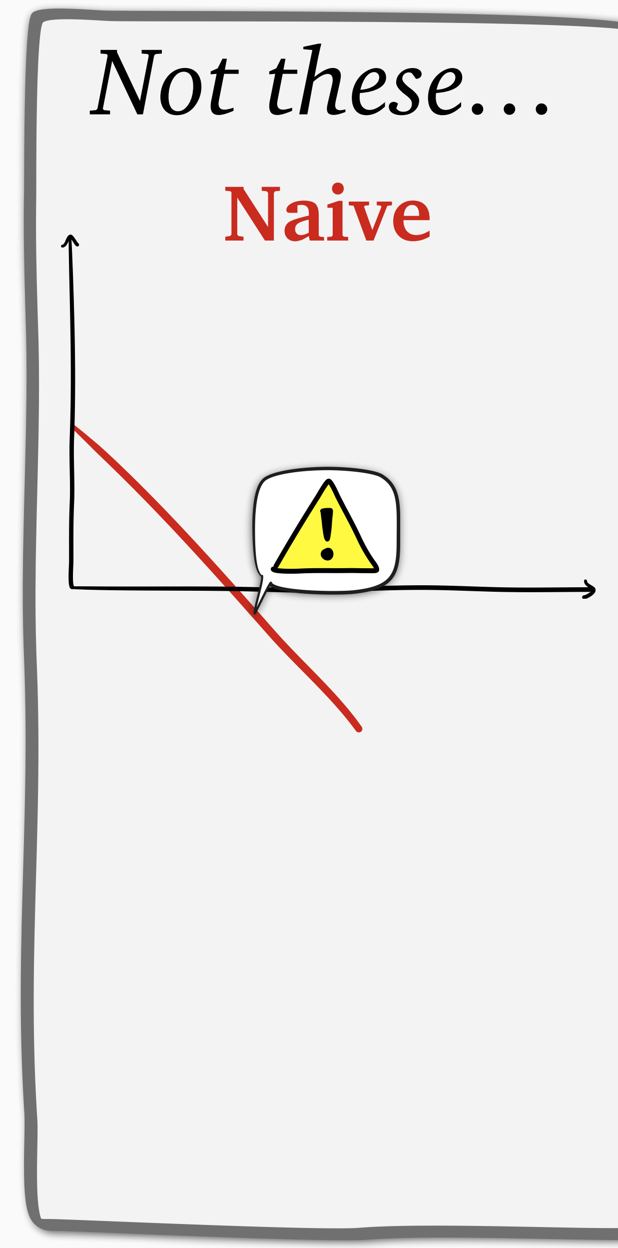


What's the right rank function?

Checkmark

$$\text{rank}_{s,z}(a) = |z - a|$$



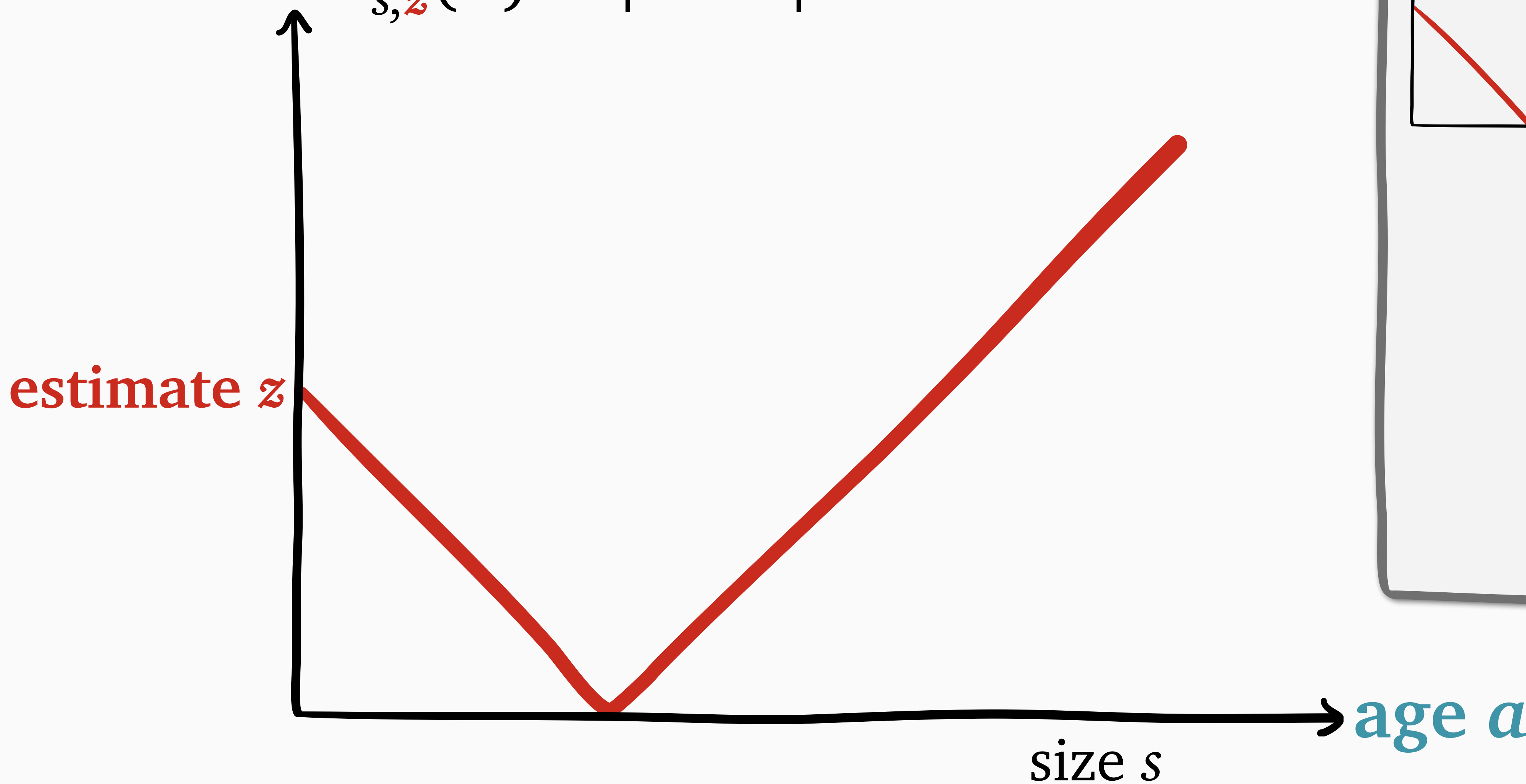


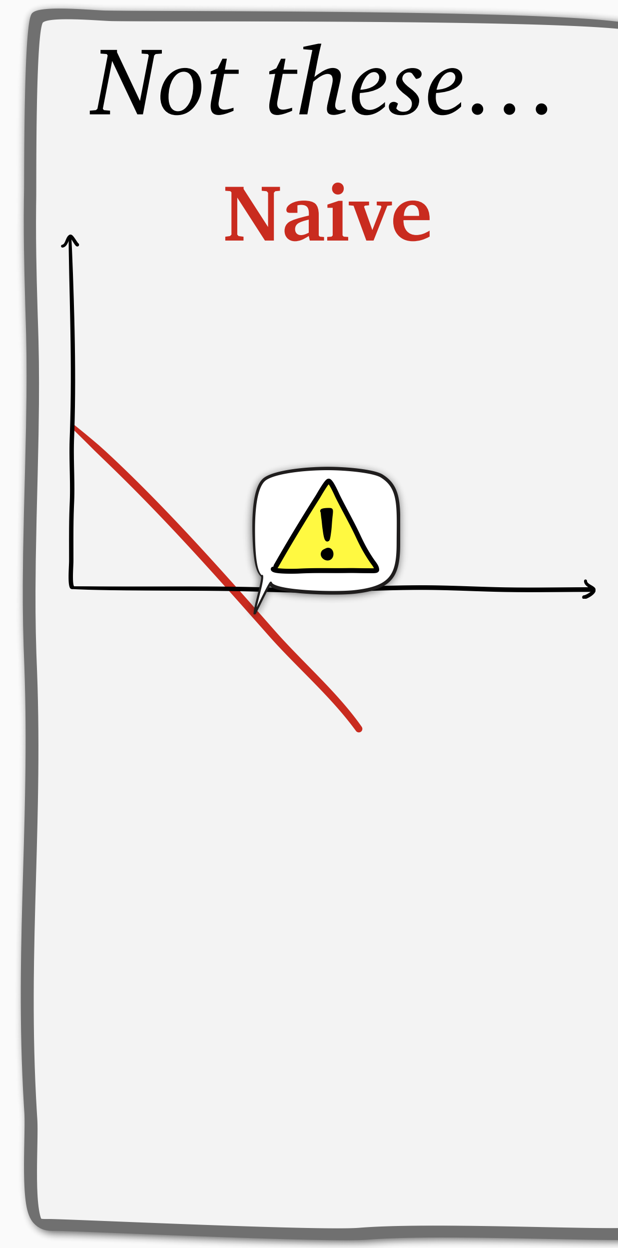
What if $\beta < \frac{1}{2}$?

What's the right **rank** function?

Checkmark

$$\text{rank}_{s,z}(a) = |z - a|$$



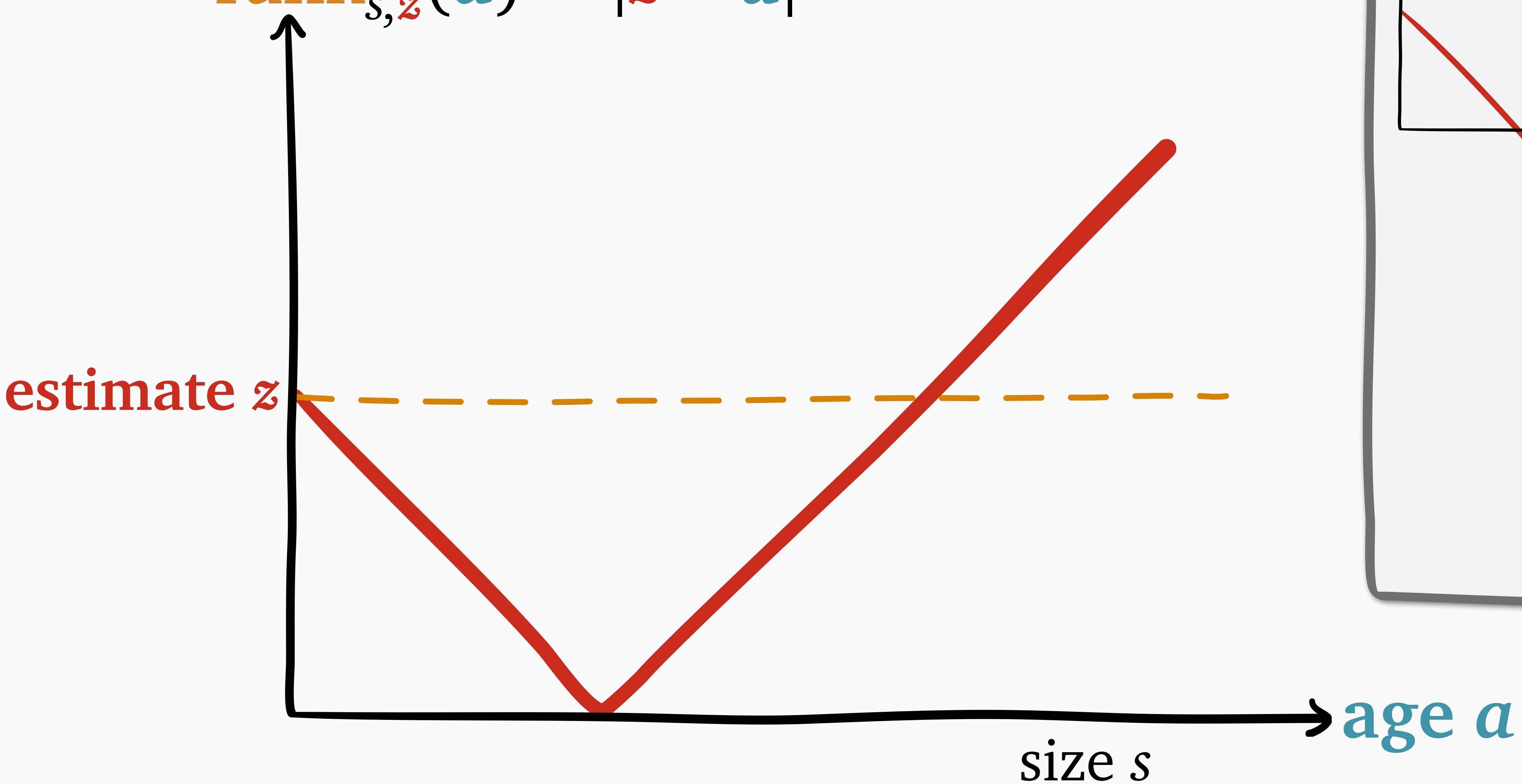


What if $\beta < \frac{1}{2}$?

What's the right rank function?

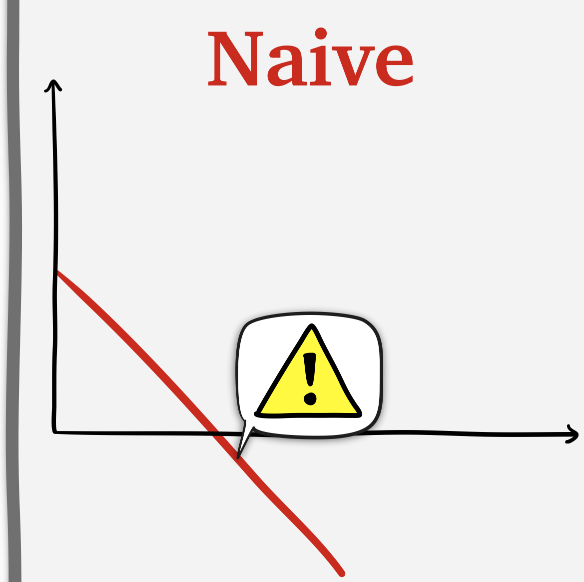
Checkmark

$\text{rank}_{s,z}(a) = |z - a|$



What if $\beta < \frac{1}{2}$?

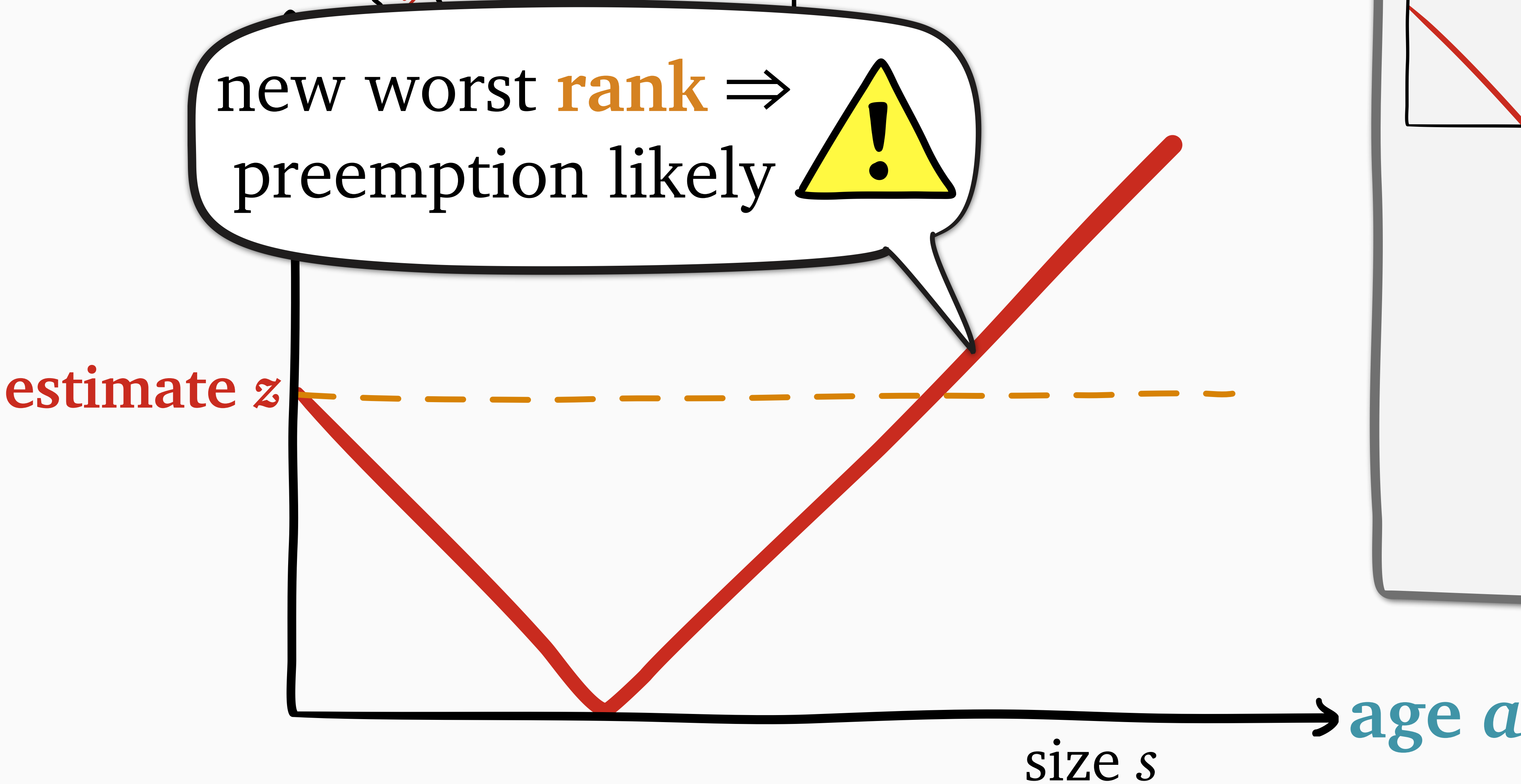
Not these…

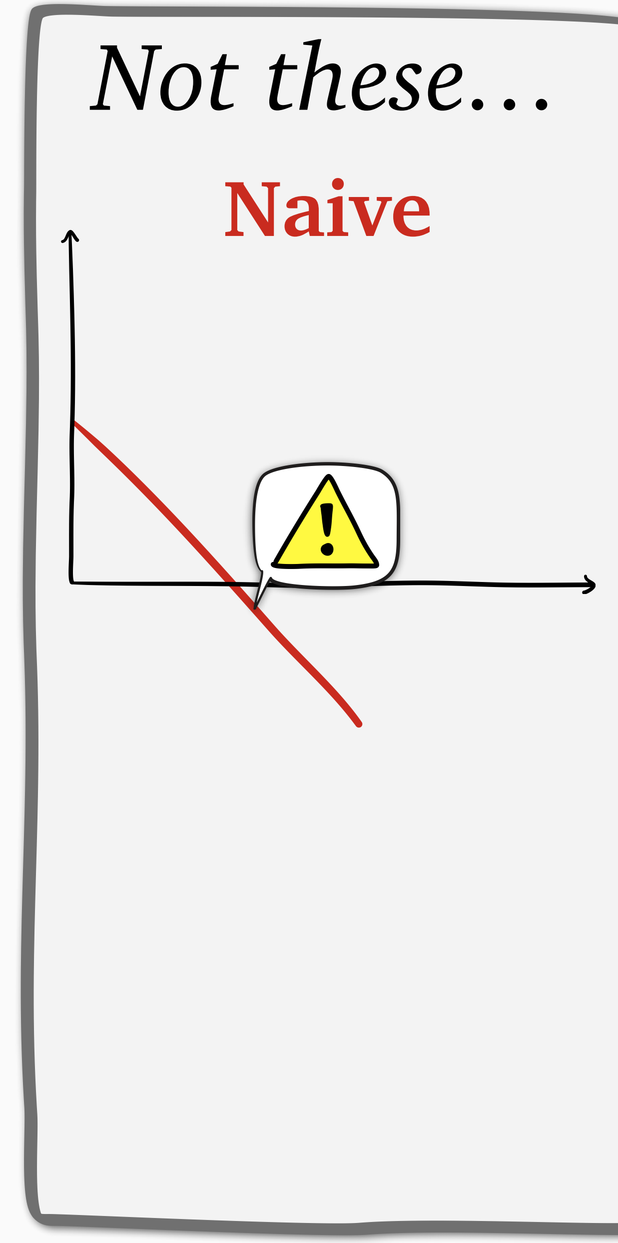


What's the right **rank** function?

Checkmark

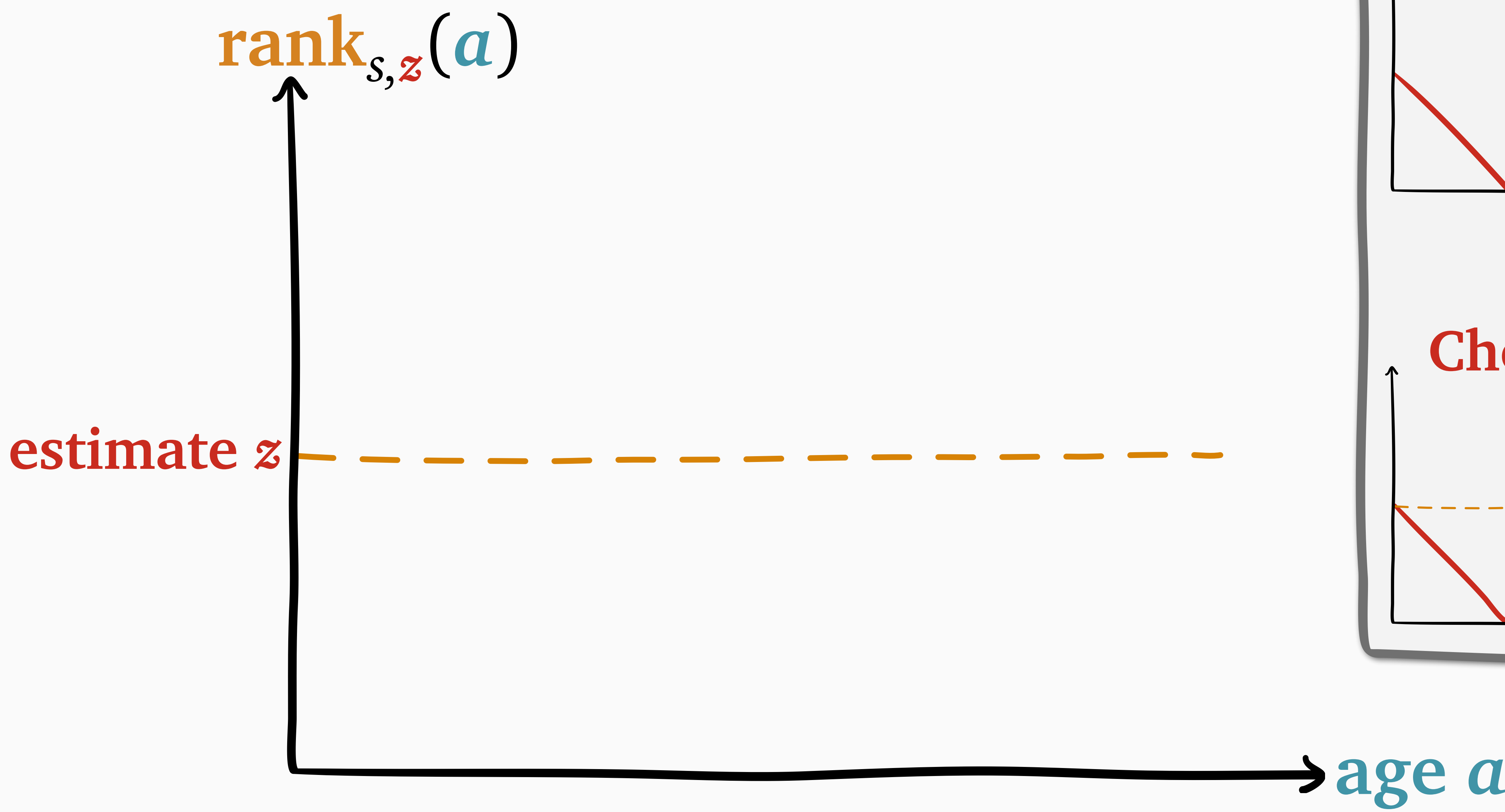
$\mathrm{rank}_{s,z}(a) = |z - a|$

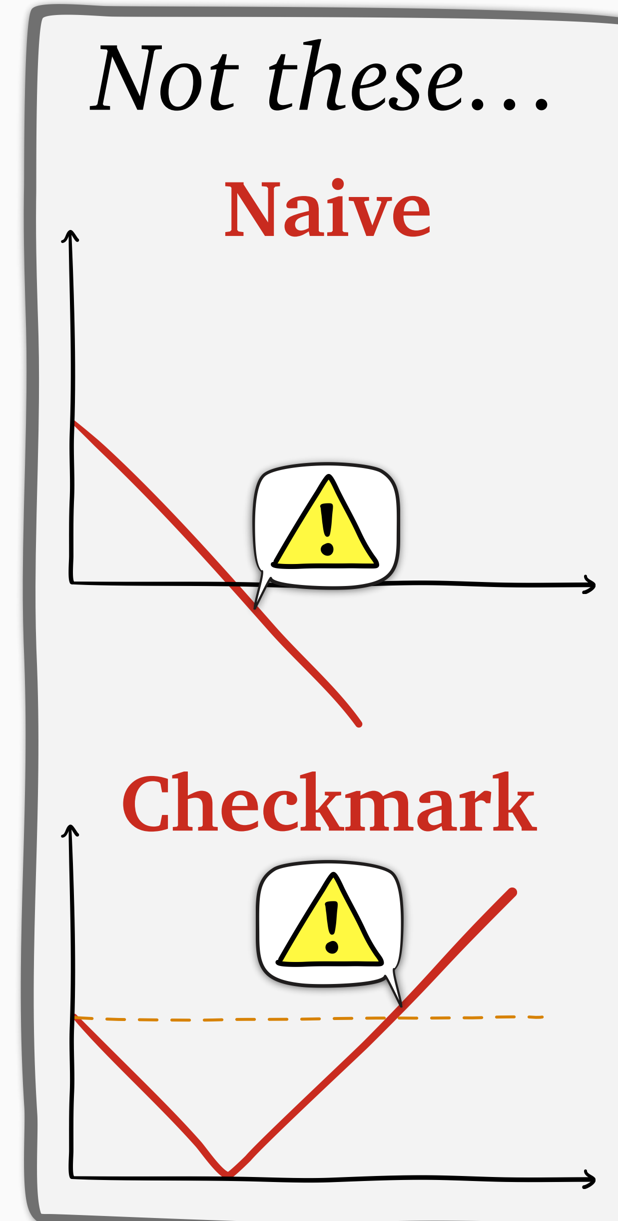




What if $\beta < \frac{1}{2}$?

What's the right rank function?

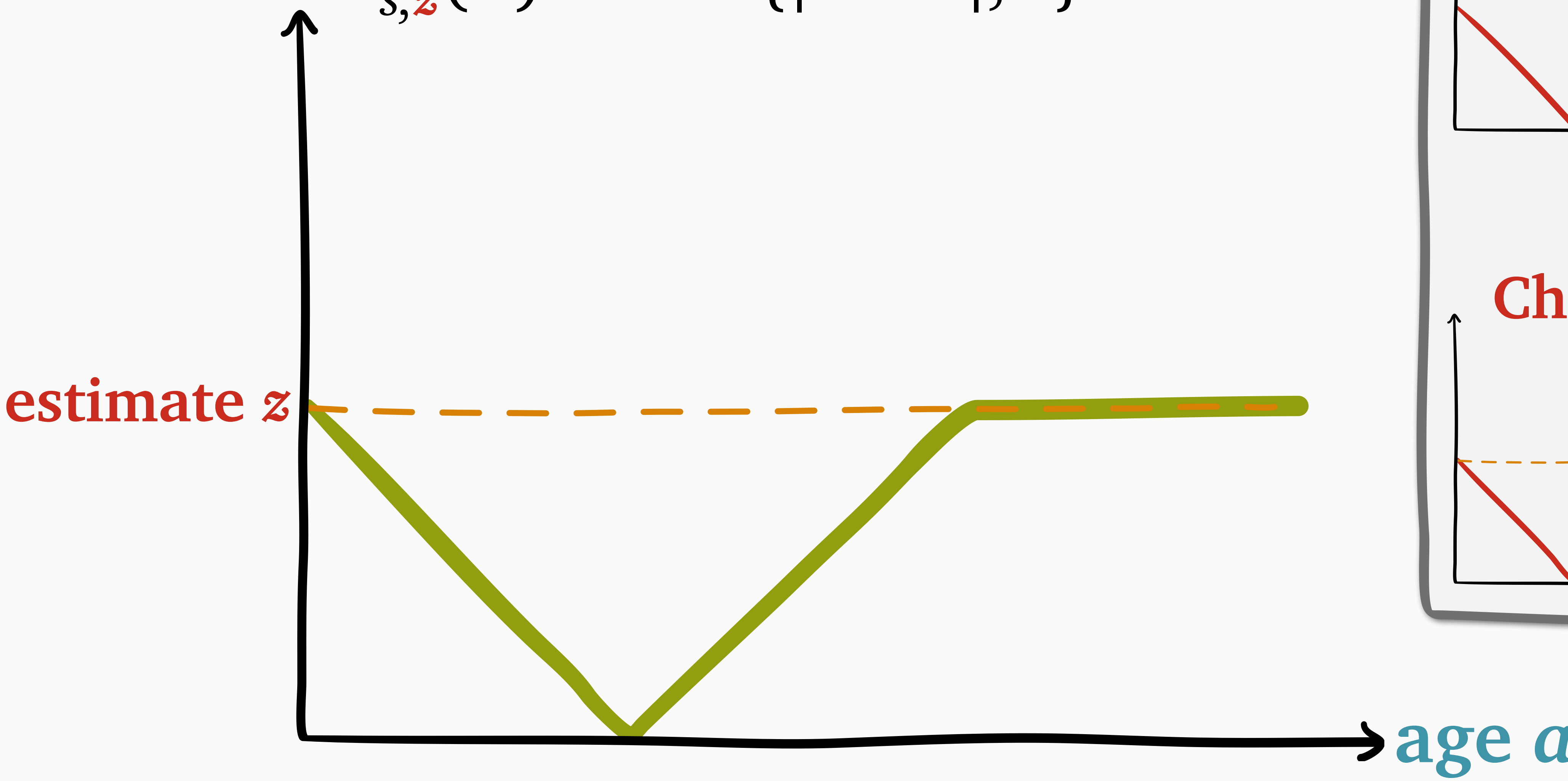


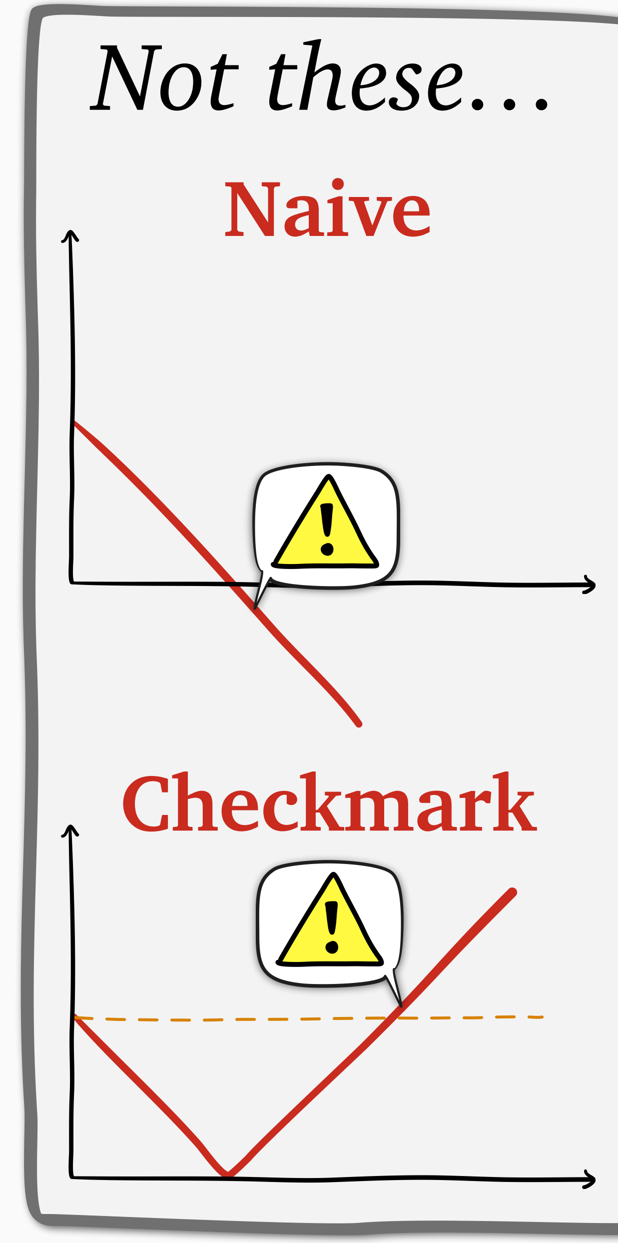


What’s the right **rank** function?

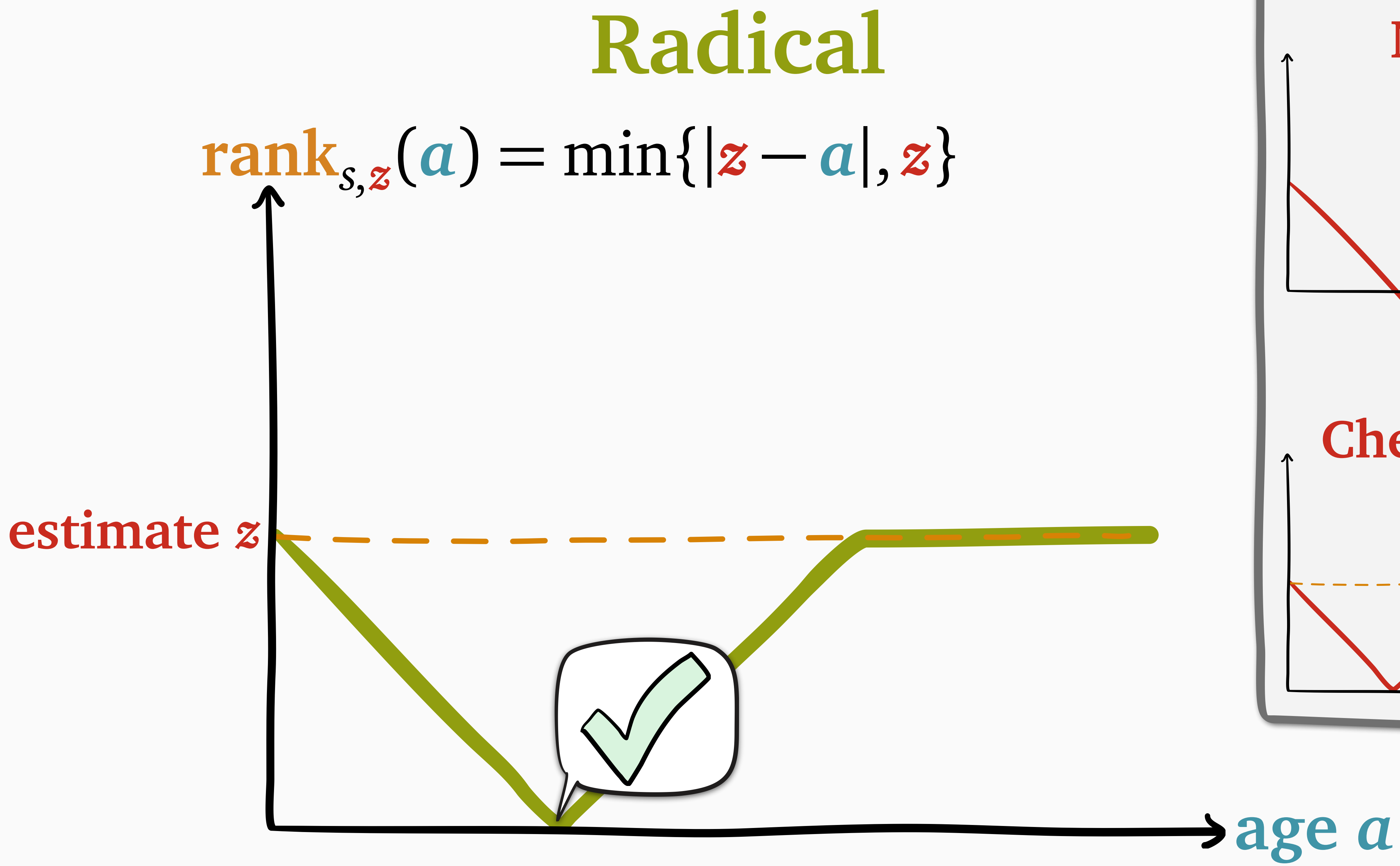
Radical

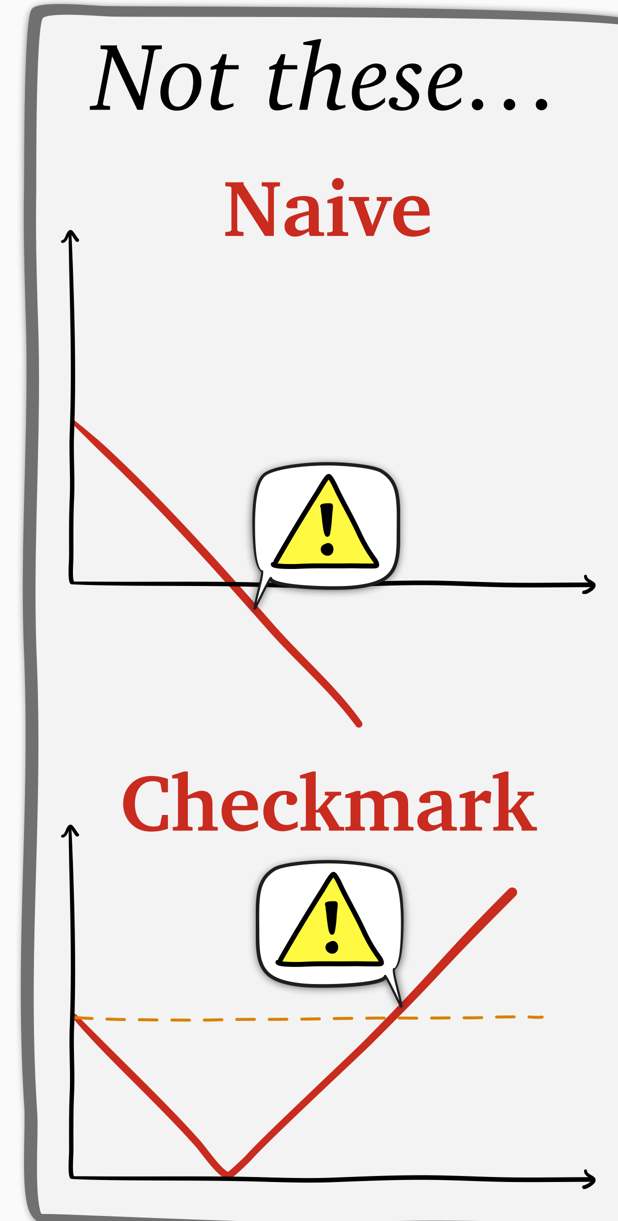
$$\text{rank}_{s,z}(a) = \min\{|z - a|, z\}$$



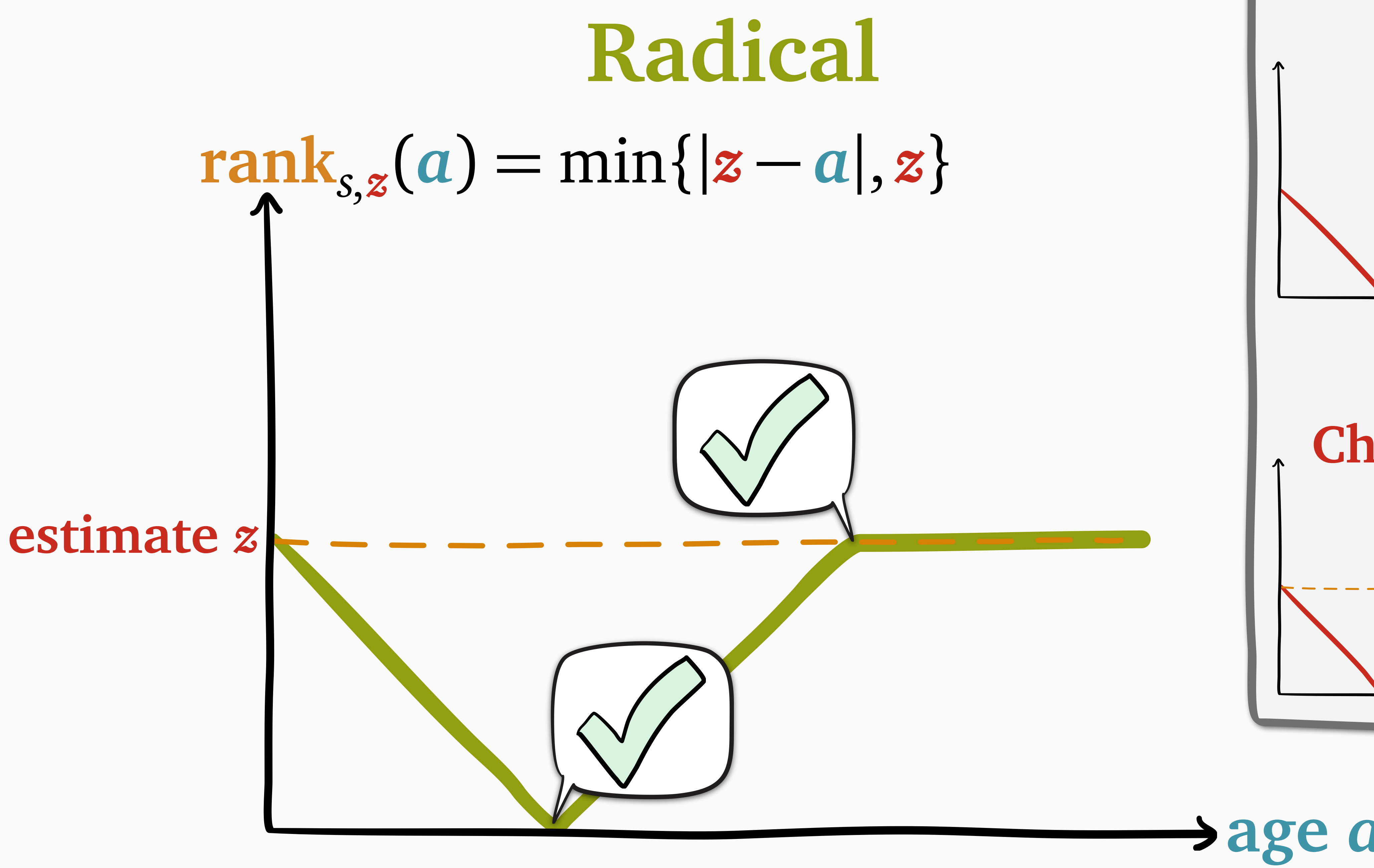


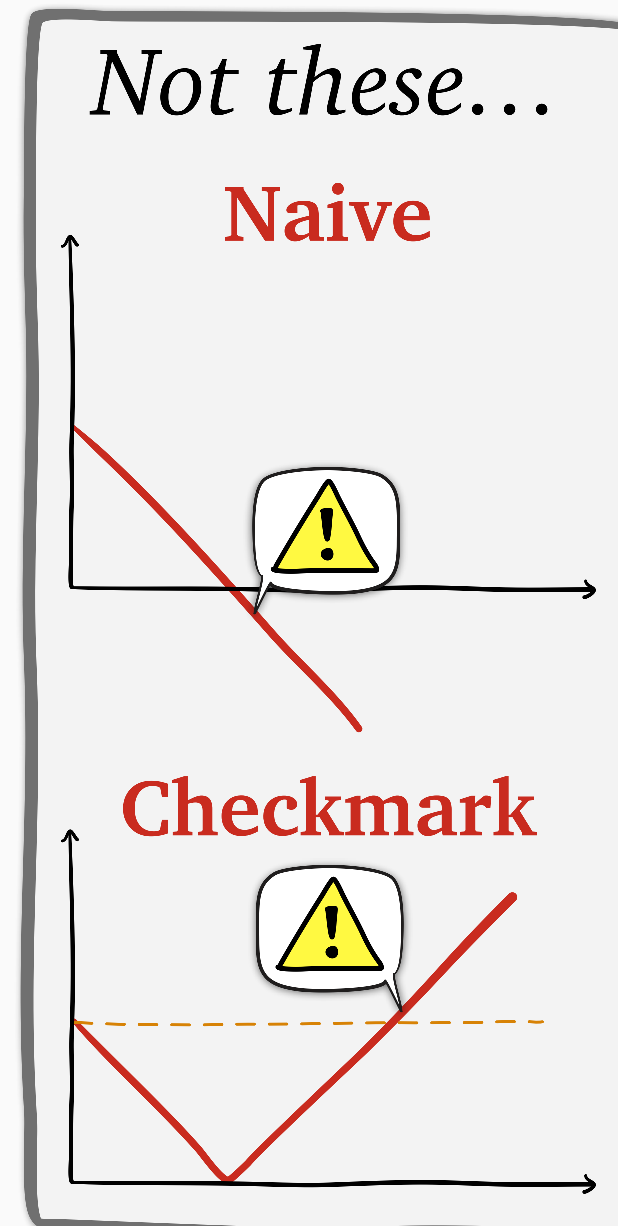
What's the right rank function?





What's the right **rank** function?

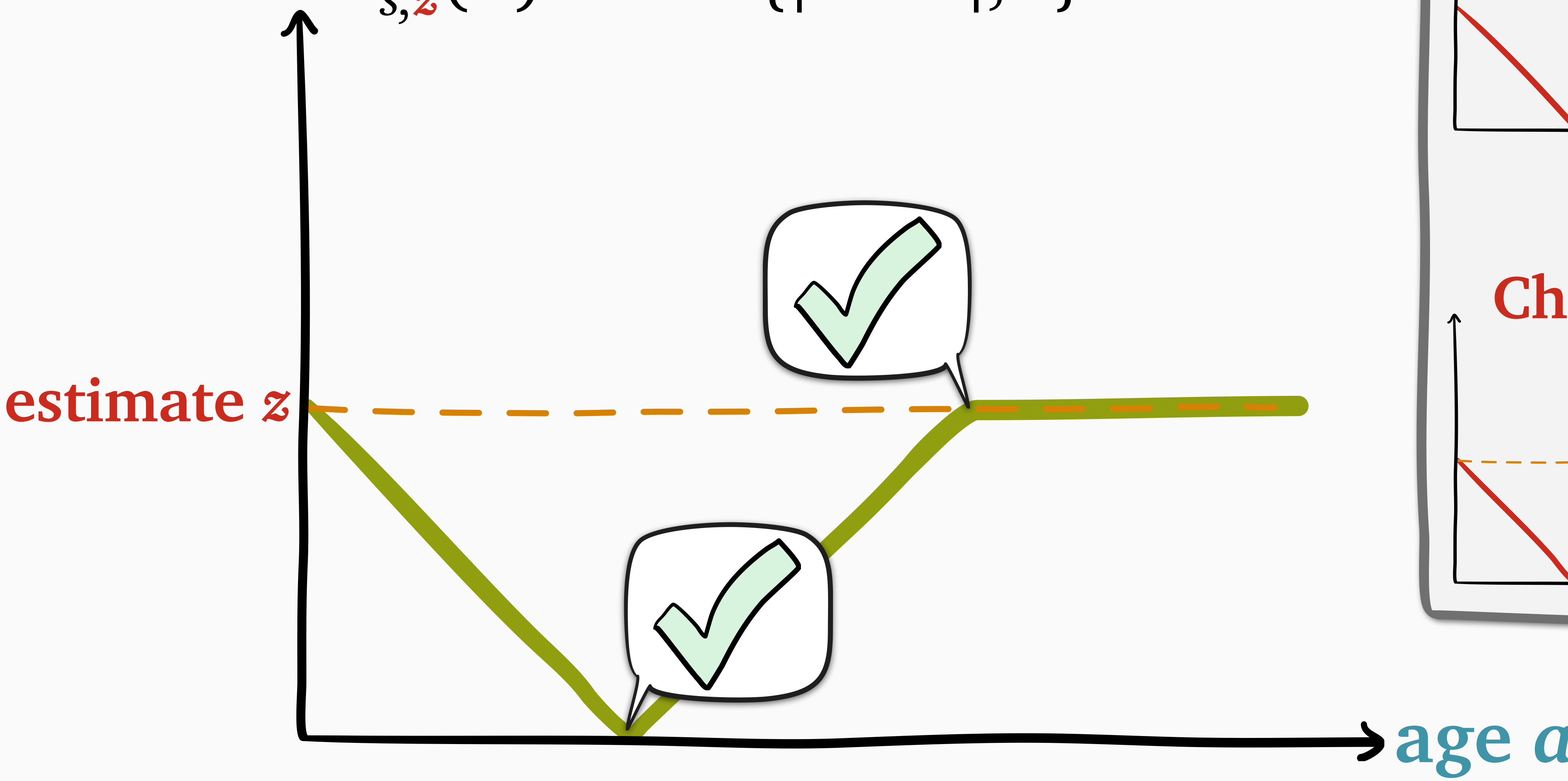


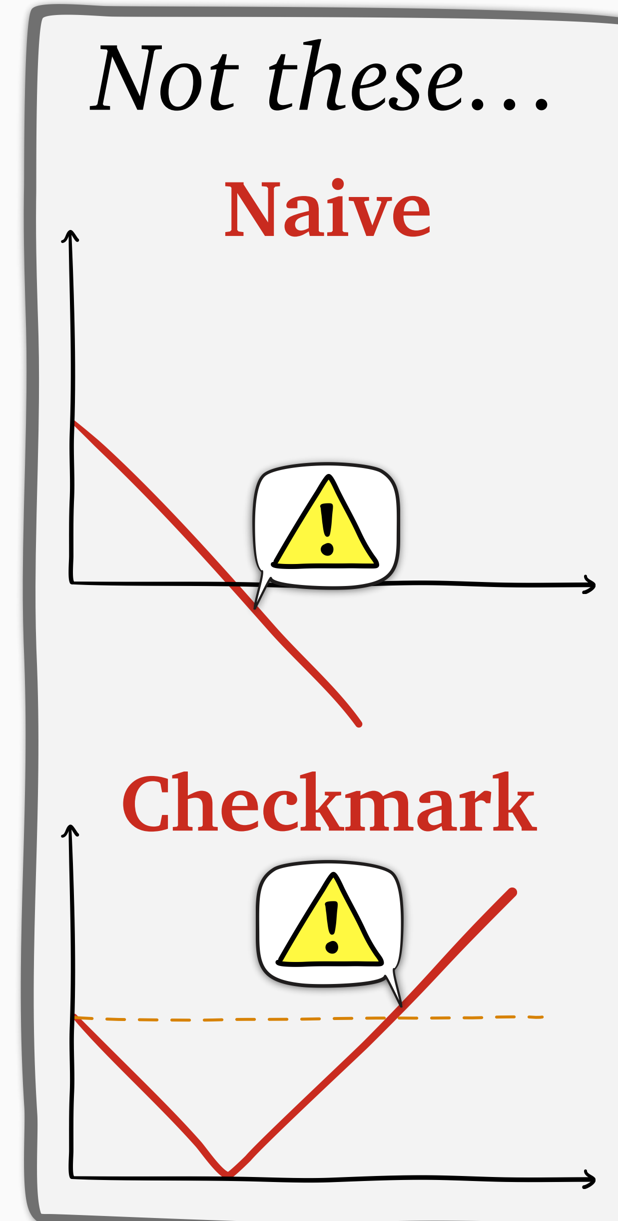


What's the right rank function?

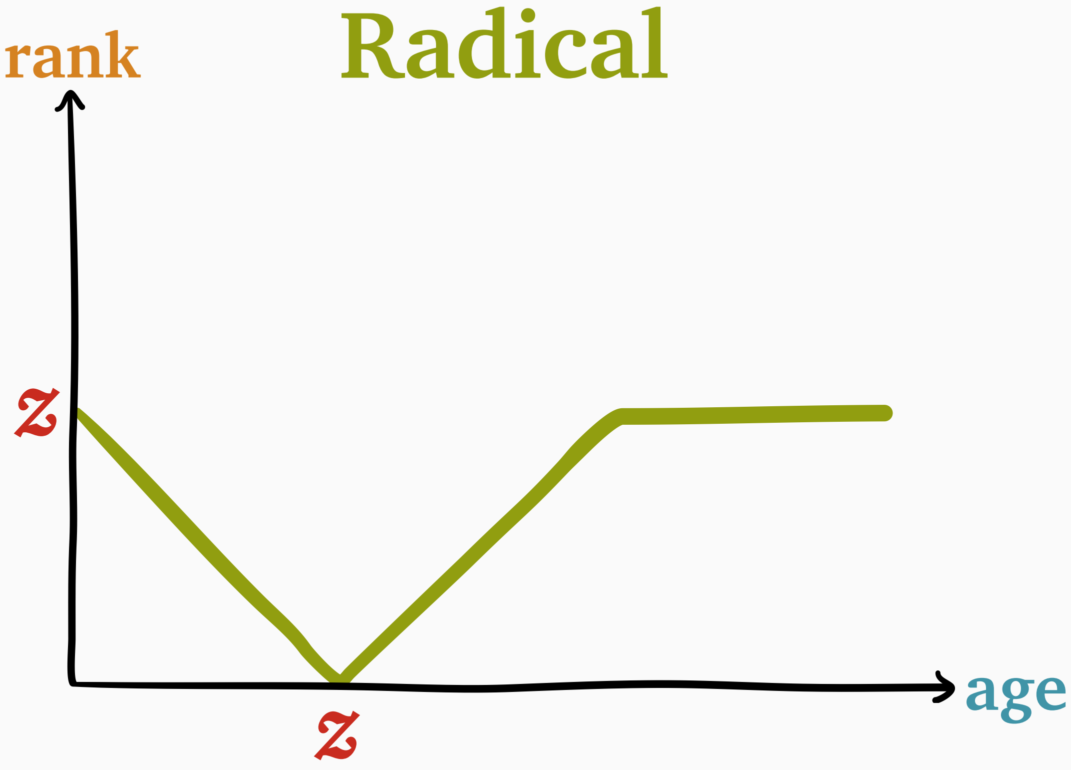
Radical

$$\mathrm{rank}_{s,z}(a) = \min\{|z - a|, z\}$$

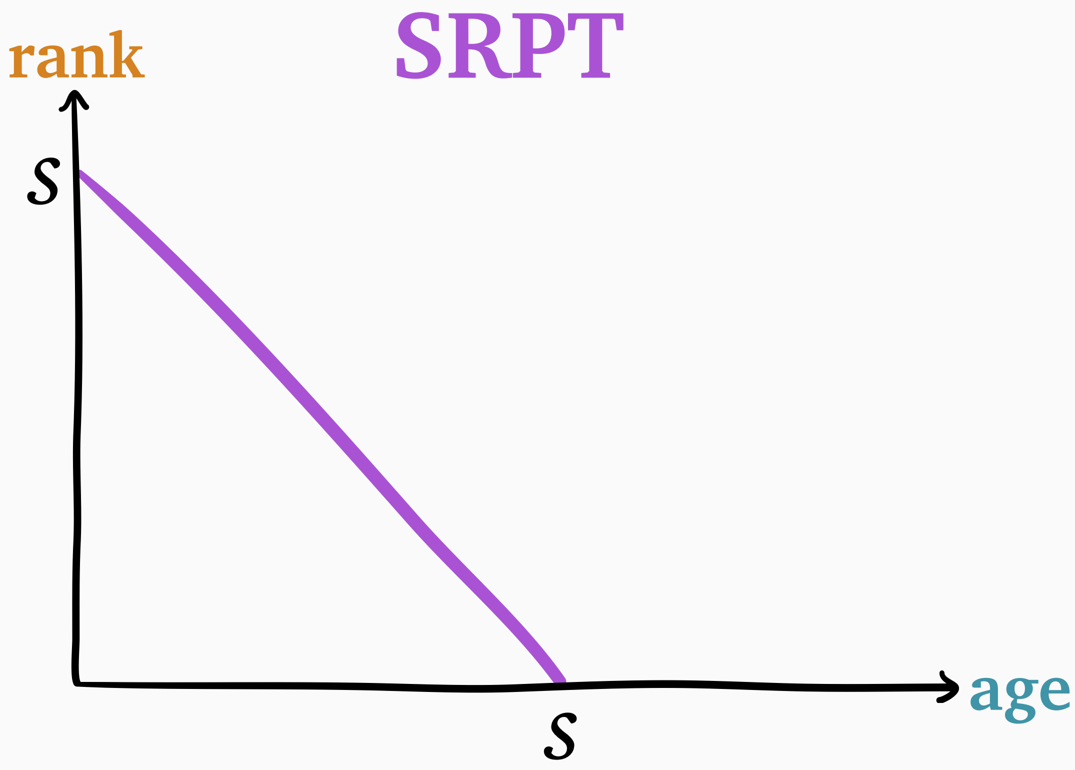




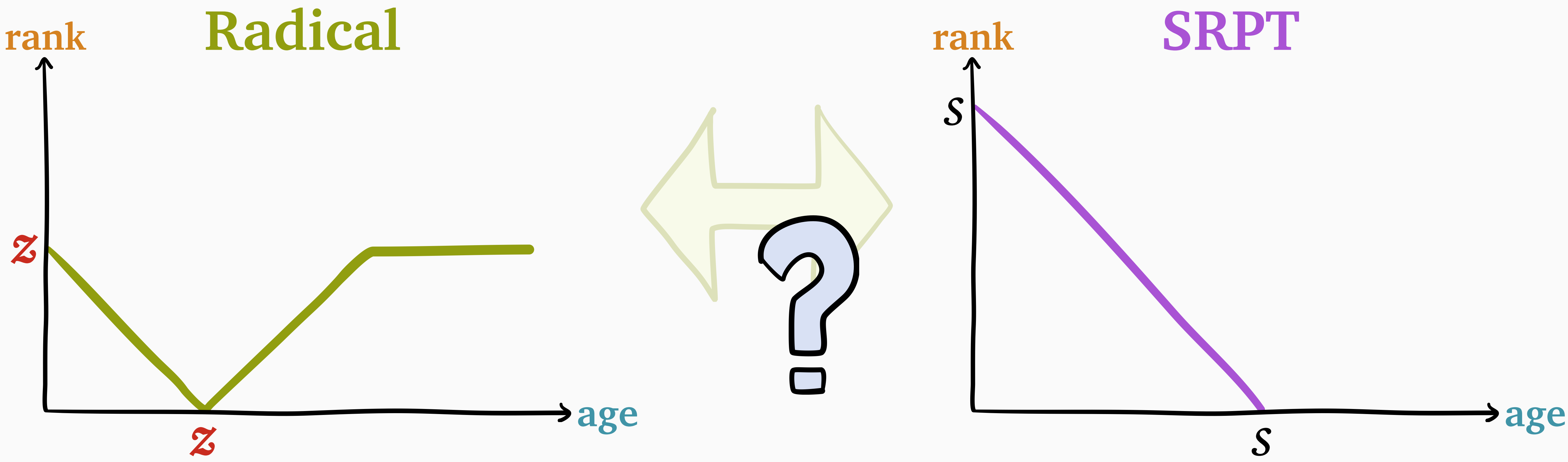
Theorem: Radical is 1-consistent, 3.5-graceful

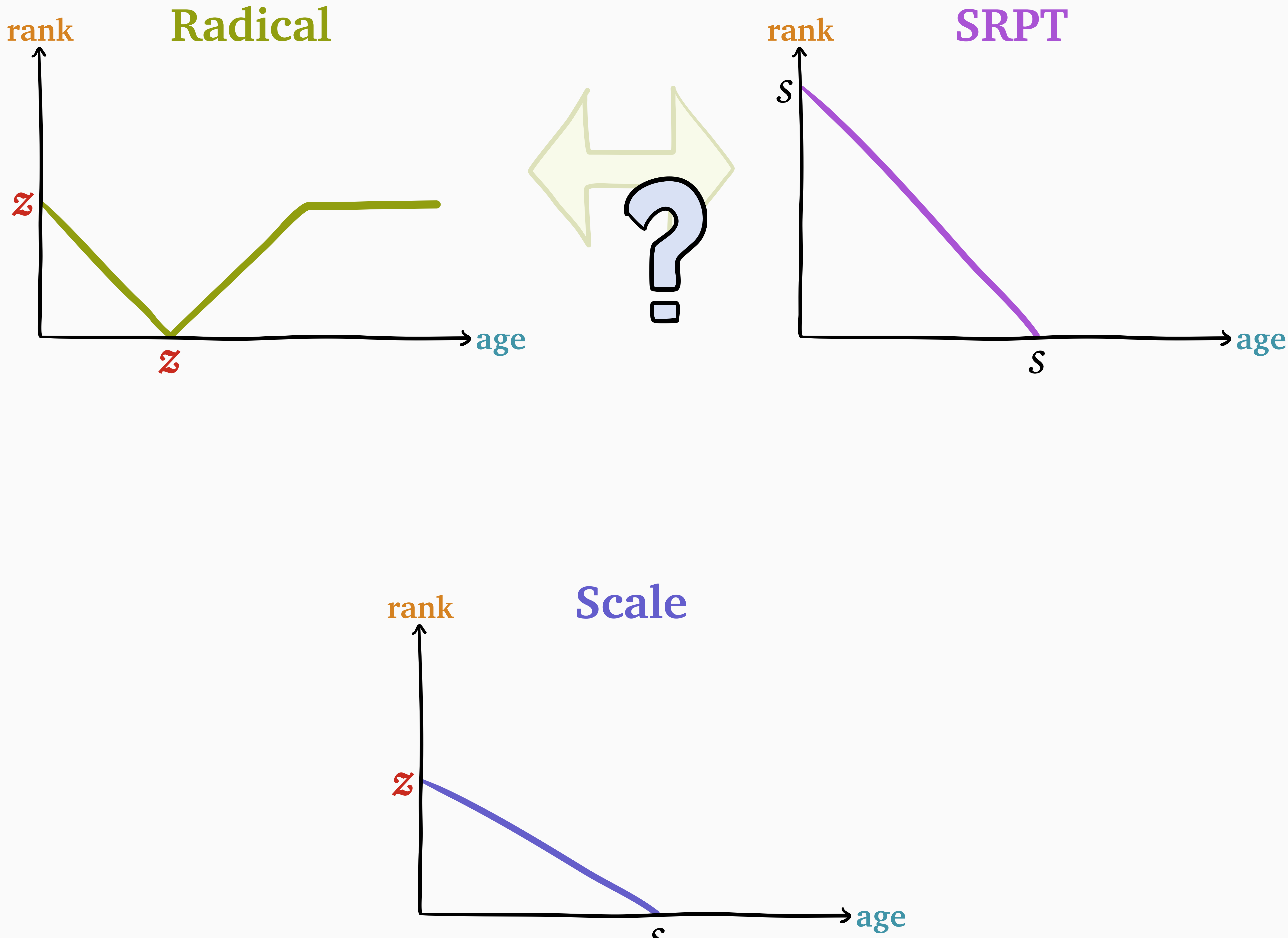


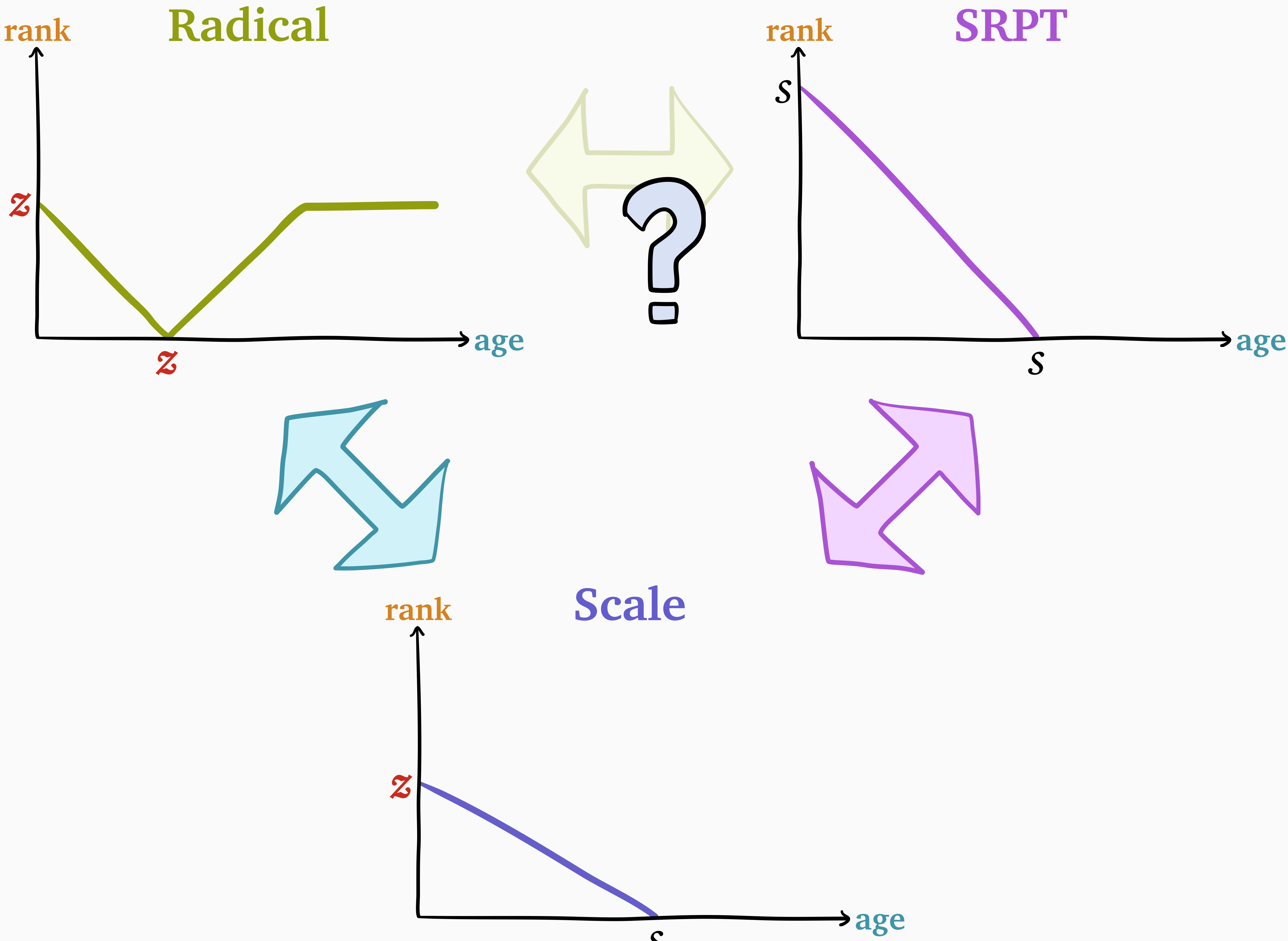
Radical
rank
z
z
age

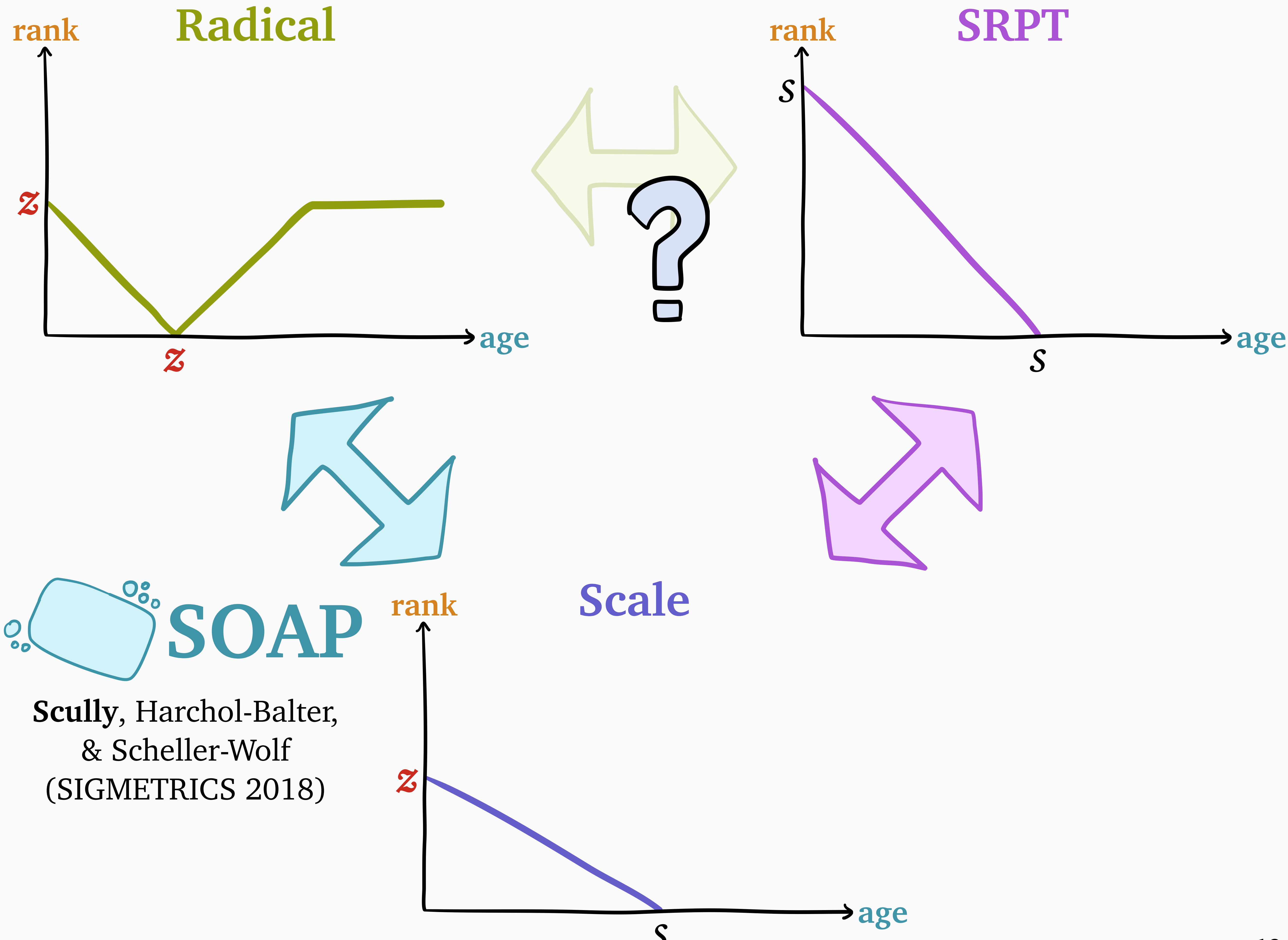


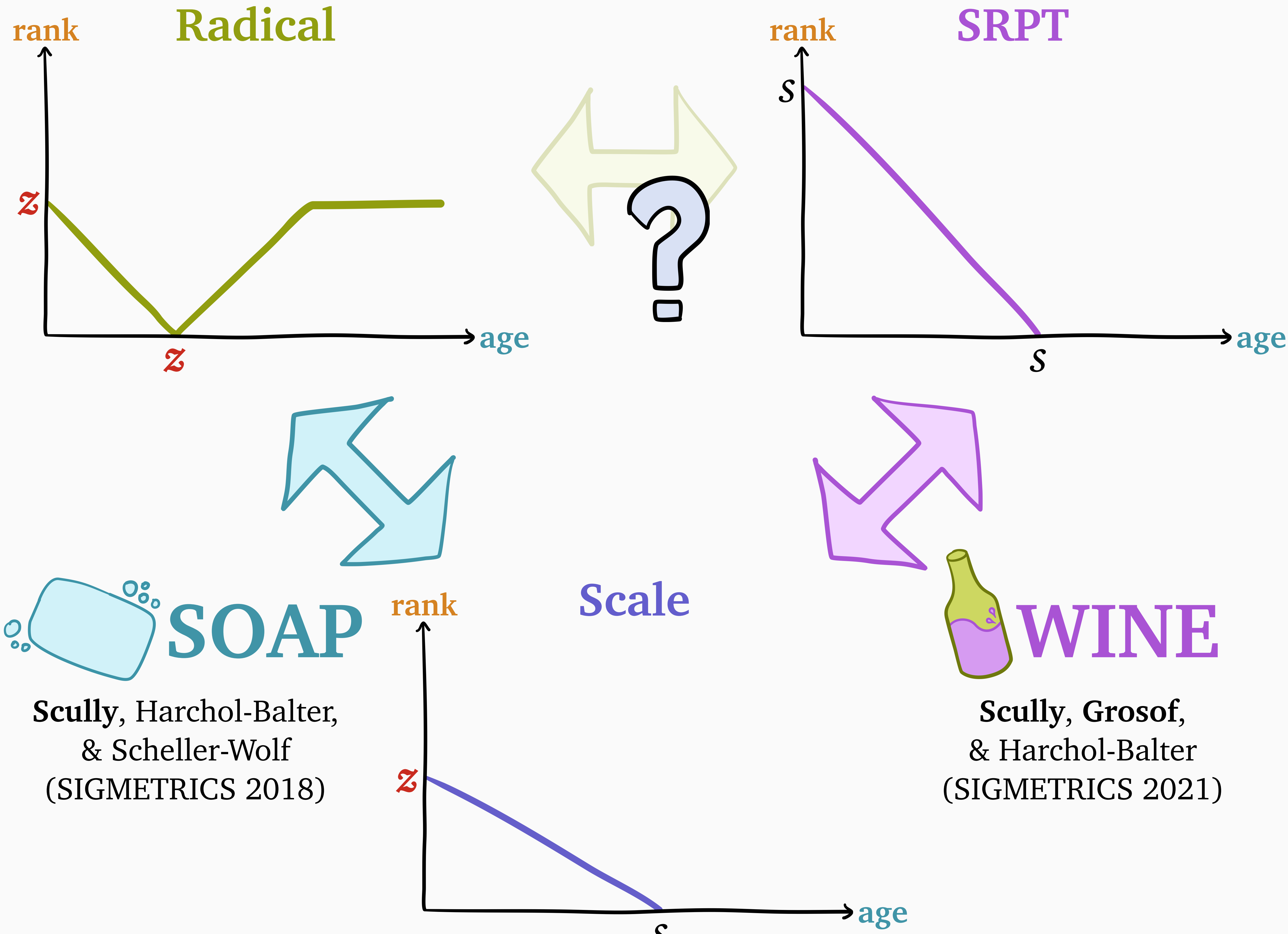
SRPT
rank
s
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scully, Harchol-Balter,
& Scheller-Wolf
(SIGMETRICS 2018)
Scale
rank
z
s
age


Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scale
rank
z
s
age
WINE
Scully, Harchol-Balter,
& Scheller-Wolf
(SIGMETRICS 2018)
Scully, Grosof,
& Harchol-Balter
(SIGMETRICS 2021)

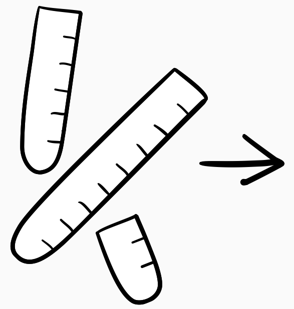
SOAP

Schedule Ordered by Age-based Priority

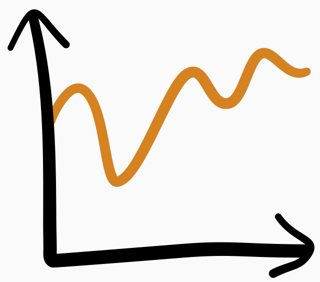
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

stochastic arrival process λ, $(S, \boldsymbol{Z})$



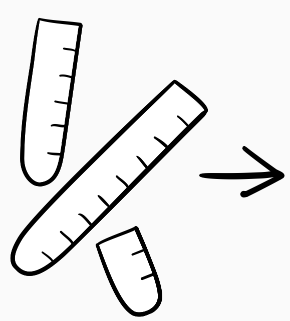
any **rank** function



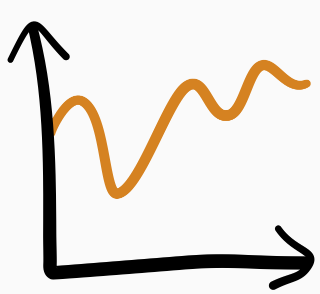
SOAP

Schedule **O**rdered by **A**ge-based **P**riority

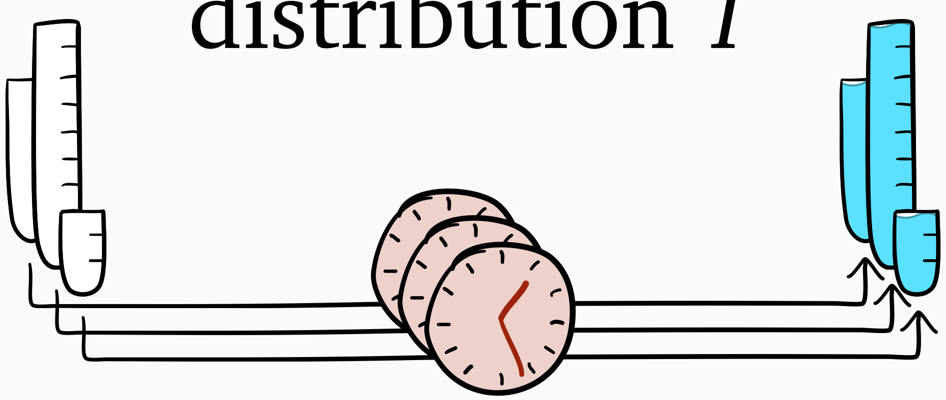
stochastic arrival process λ, $(S, \boldsymbol{Z})$

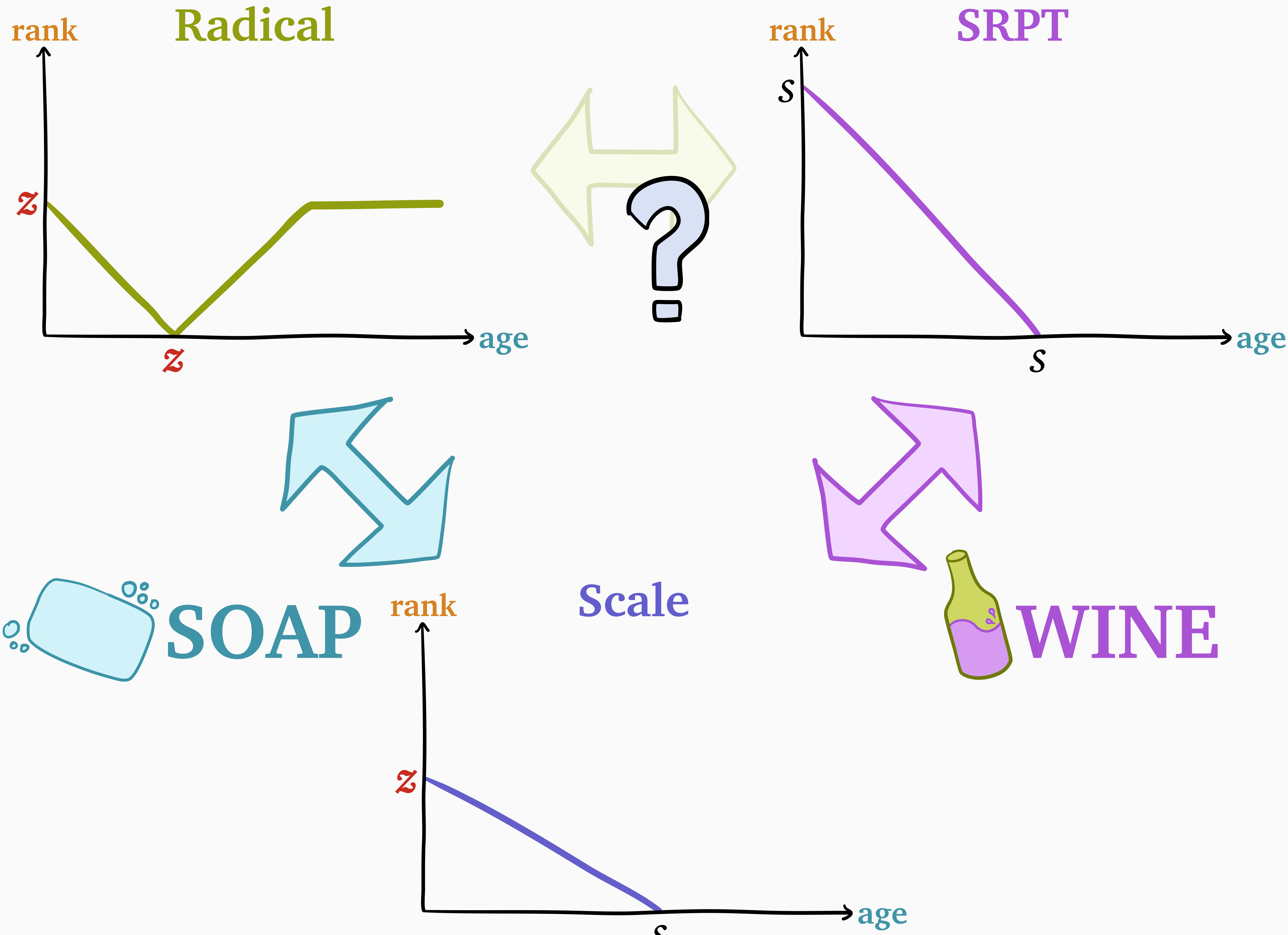


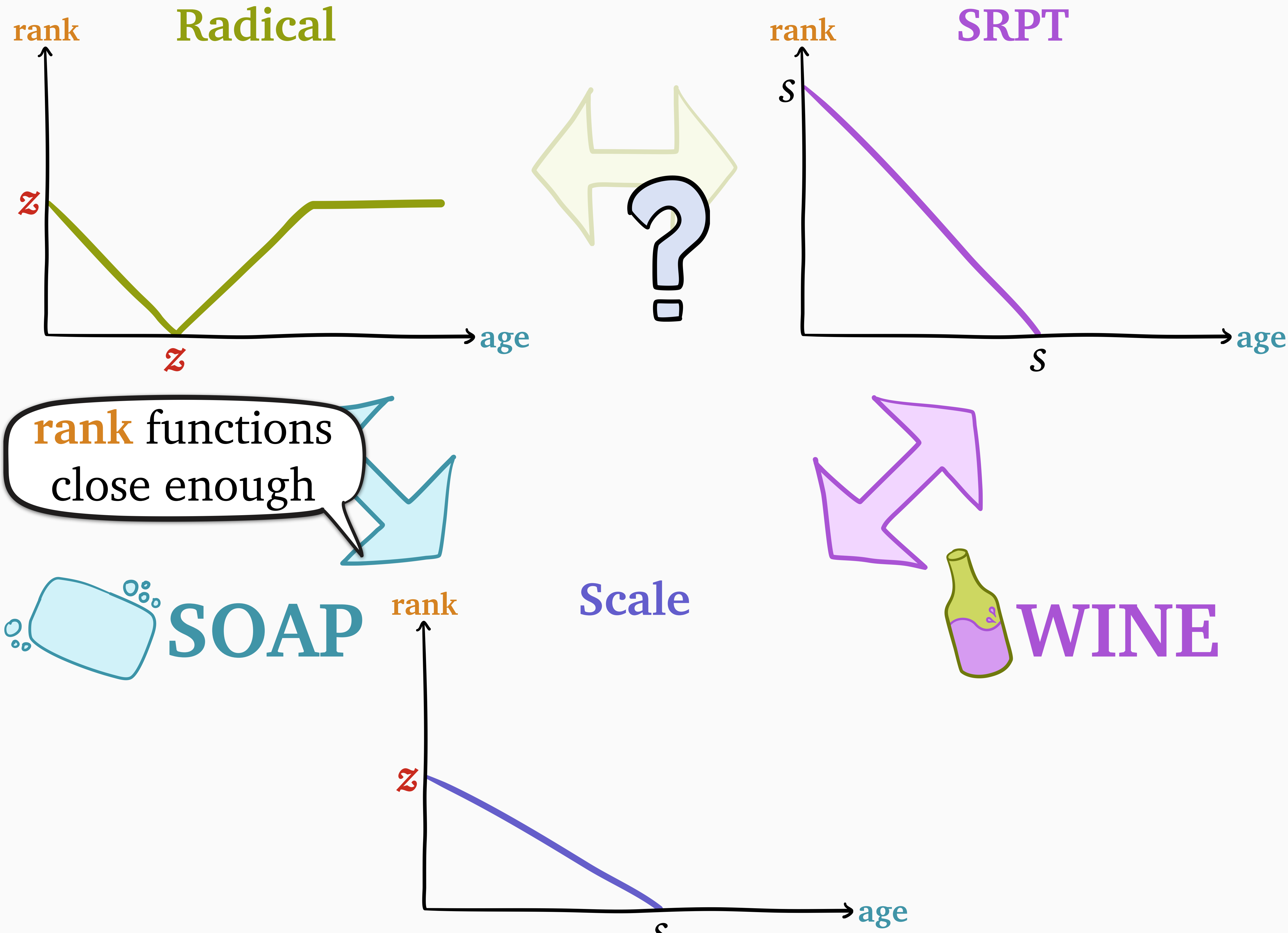
any **rank** function



response time distribution T




Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scale
rank
z
s
age
WINE



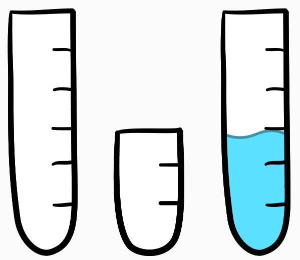
rank
Radical
z
z
age
rank
SRPT
s
s
age
rank functions close enough
SOAP
rank
Scale
z
s
age
WINE

WINE

Work Integral Number Equality

WINE

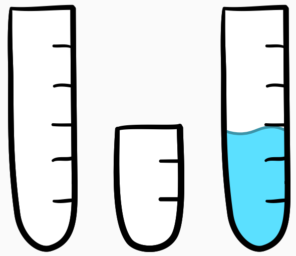
Work Integral Number Equality



number of jobs N

WINE

Work **I**ntegral **N**umber **E**quality

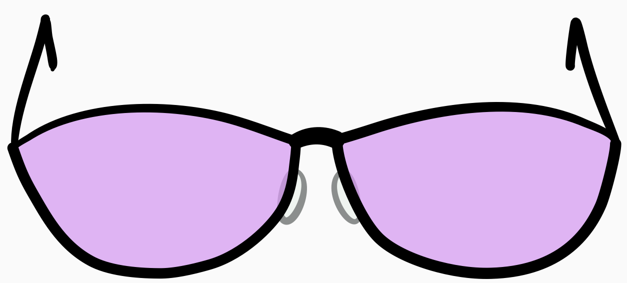


number of jobs N

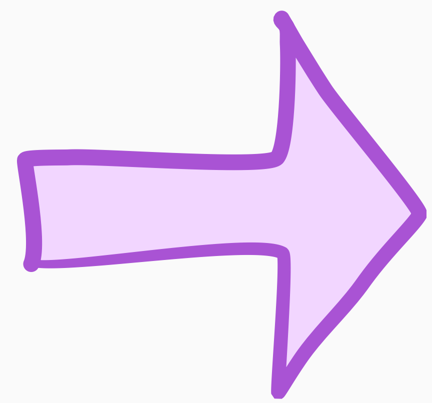
Little's Law:
$\mathbf{E}[T] = \mathbf{E}[N]/\lambda$

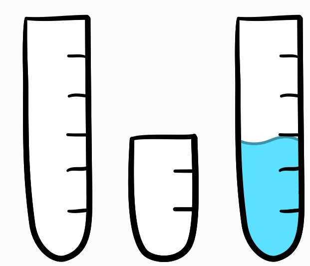
WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$





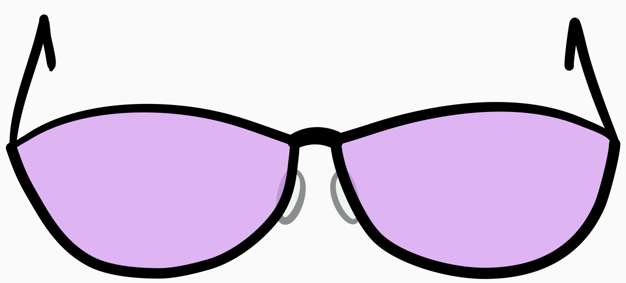
number of jobs N

Little's Law:

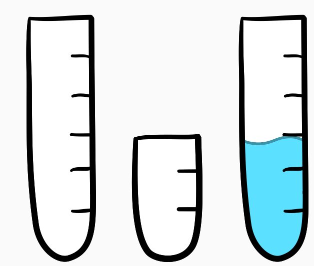
$\mathbf{E}[T] = \mathbf{E}[N]/\lambda$

WINE

Work **I**ntegral **N**umber **E**quality



r-work $W(r)$



number of jobs N

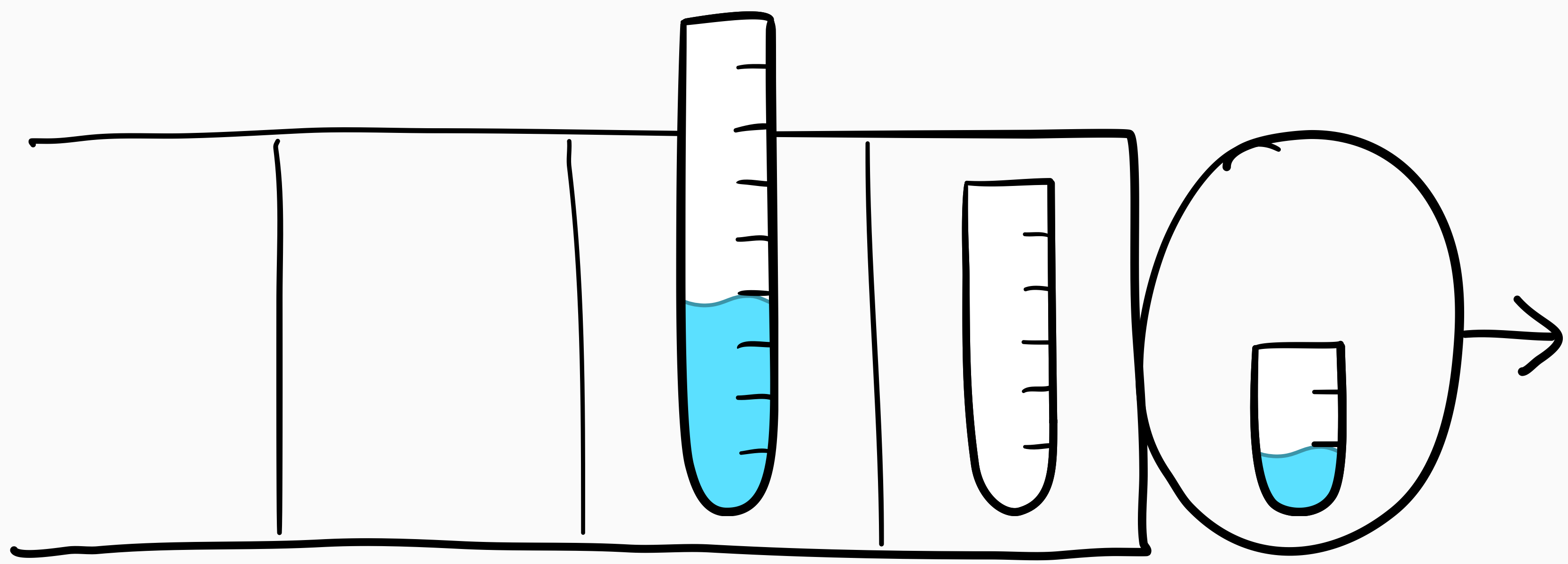
Little's Law:
$\mathbf{E}[T] = \mathbf{E}[N]/\lambda$

What is r-work?

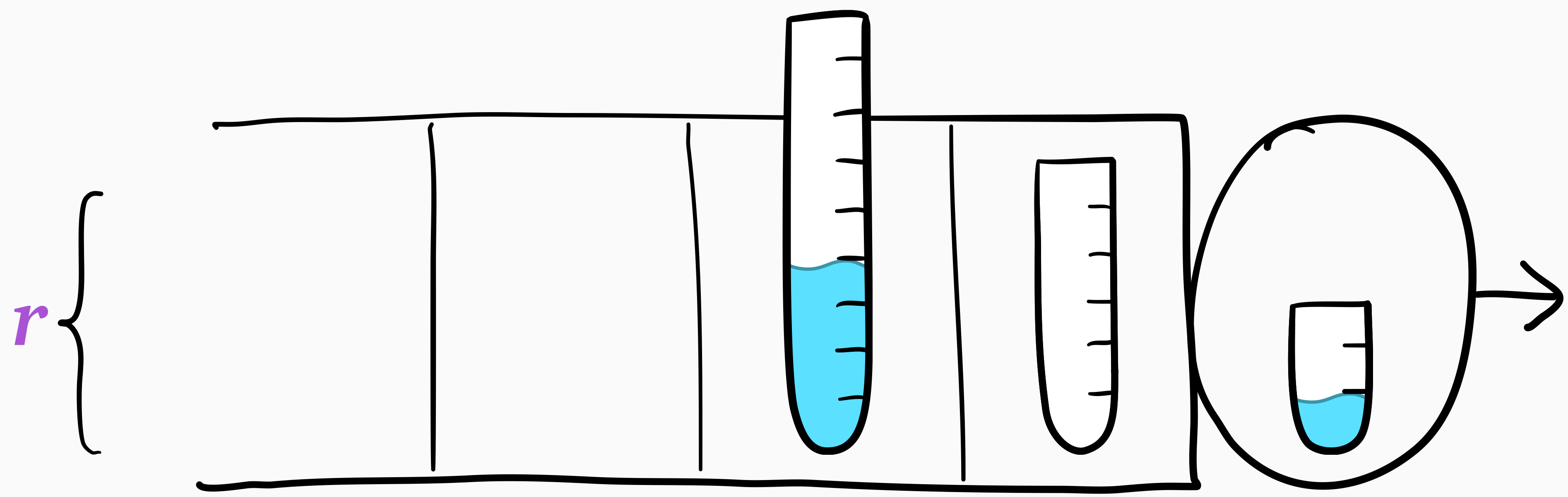
Get N from r-work?

Bound **Scale**'s $\mathbf{E}[T]$?

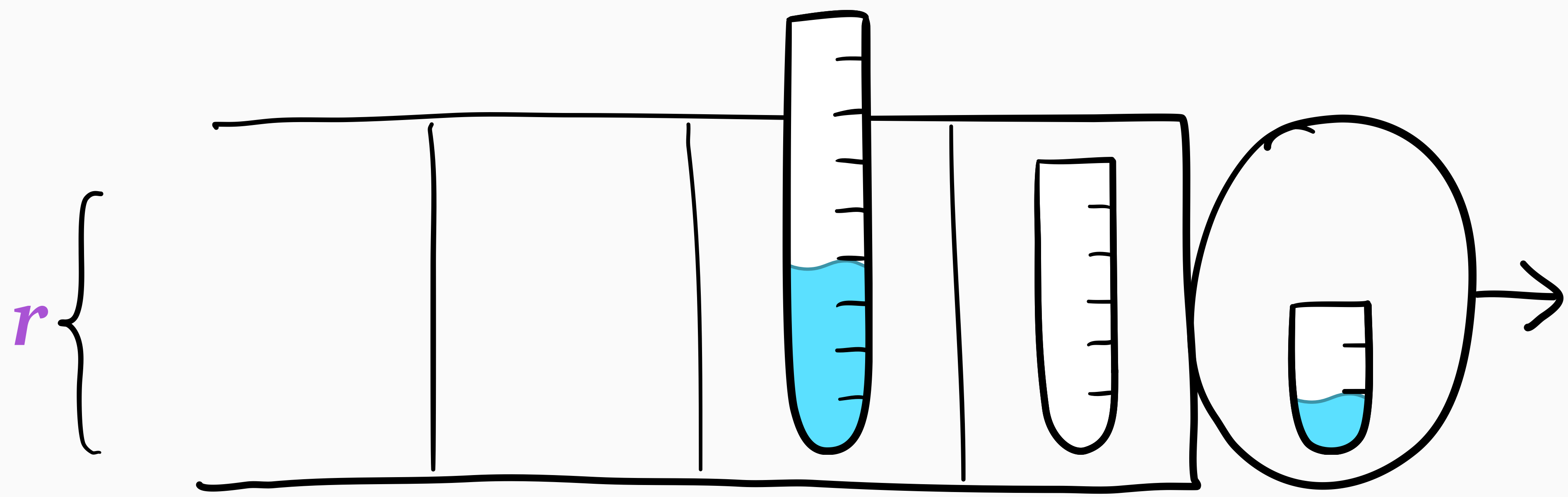
What is *r*-work?



What is *r*-work?



What is *r*-work?

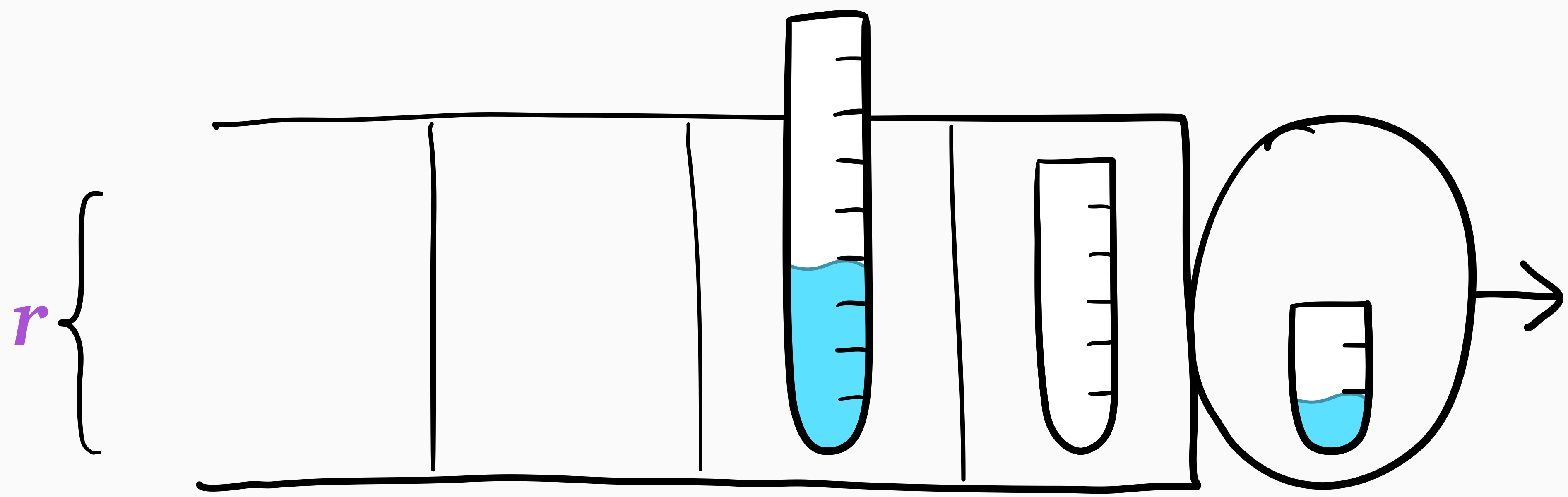


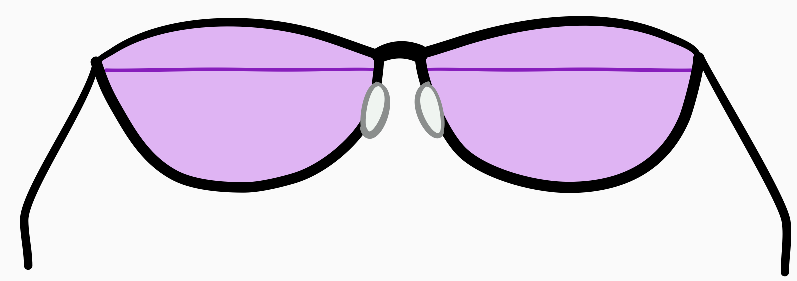
Definition:

$W(r)$ = total remaining size of jobs whose remaining size is $\leq r$

r-work

What is r-work?



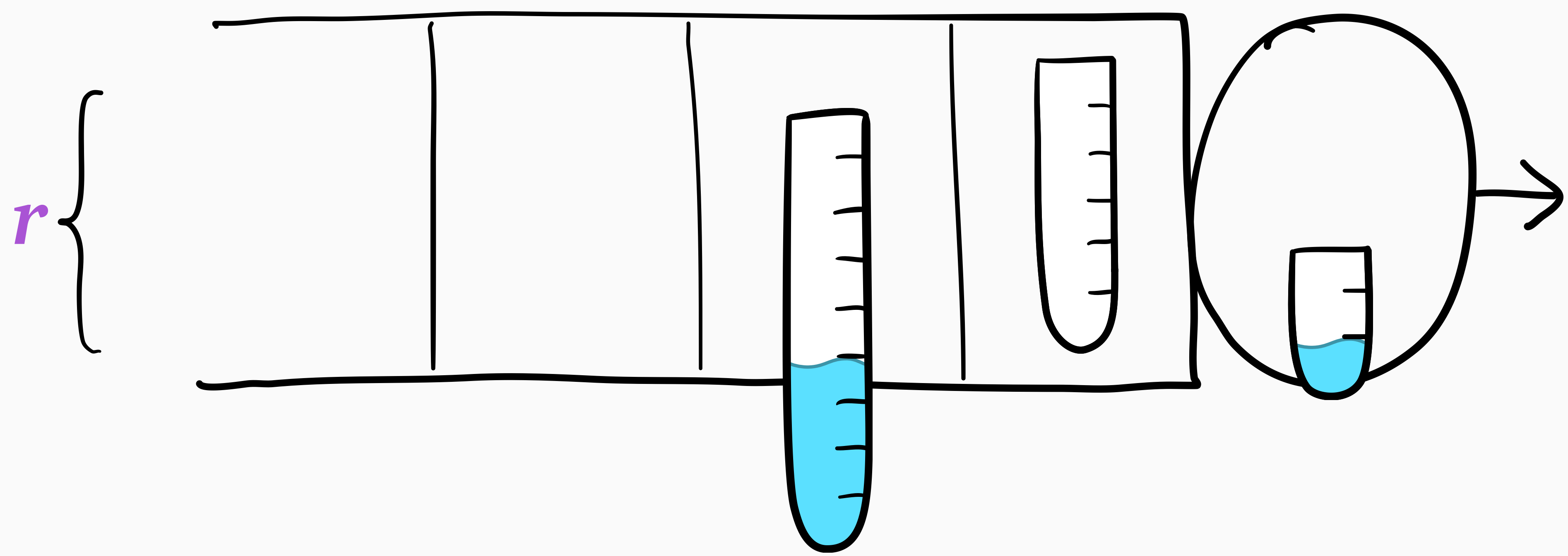


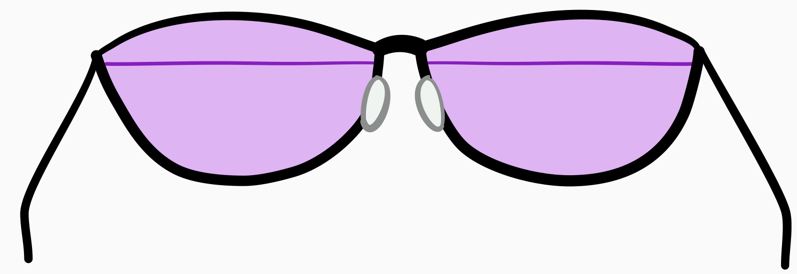
Definition:

$W(r)$ = total remaining size of jobs whose remaining size is $\leq r$

r-work

What is r-work?



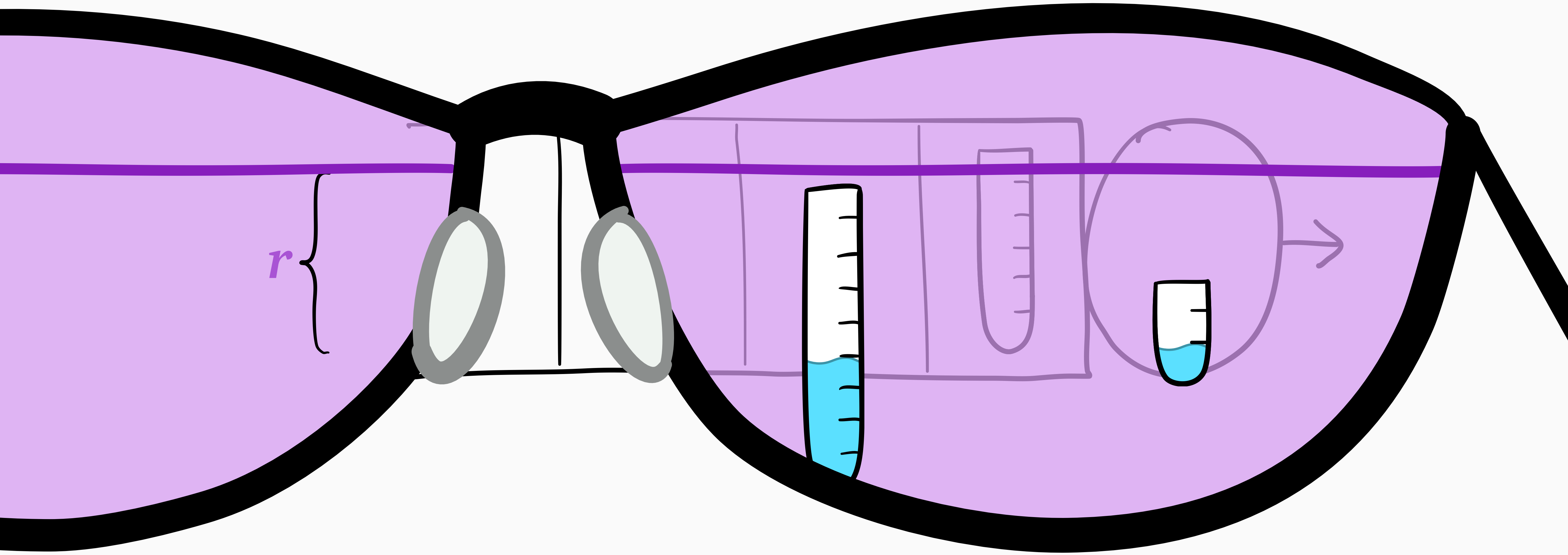


Definition:

$W(r)$ = total remaining size of jobs whose remaining size is $\leq r$

r-work

What is r-work?



Definition:

$W(r)$ = total remaining size of jobs whose remaining size is $\leq r$

r-work

 **WINE**: under *any* scheduling policy,

$$N = \int_0^\infty \frac{W(r)}{r^2} \, \mathrm{d}r$$

WINE: under *any* scheduling policy,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

NEW! **Lemma:**

$$\mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta} r\right)\right]$$

WINE: under *any* scheduling policy,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

NEW! **Lemma:**

$$\mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\tfrac{\alpha}{\beta} r\right)\right]$$

NEW! **Theorem:**

$$\mathbf{E}[T_{\mathrm{Scale}}] \le \tfrac{\alpha}{\beta}\mathbf{E}[T_{\mathrm{SRPT}}]$$

also holds in worst case

WINE: under *any* scheduling policy,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

NEW!

Lemma:

$$\mathbf{E}[W_{\text{Scale}}(r)] \leq \mathbf{E}\left[W_{\text{SRPT}}\left(\tfrac{\alpha}{\beta} r\right)\right]$$

NEW!

Theorem:

$$\mathbf{E}[T_{\text{Scale}}] \leq \tfrac{\alpha}{\beta}\mathbf{E}[T_{\text{SRPT}}]$$

also holds in worst case

WINE: under *any* scheduling policy,

$$N = \int_0^\infty \frac{W(r)}{r^2}\,\mathrm{d}r$$

NEW!

Lemma:

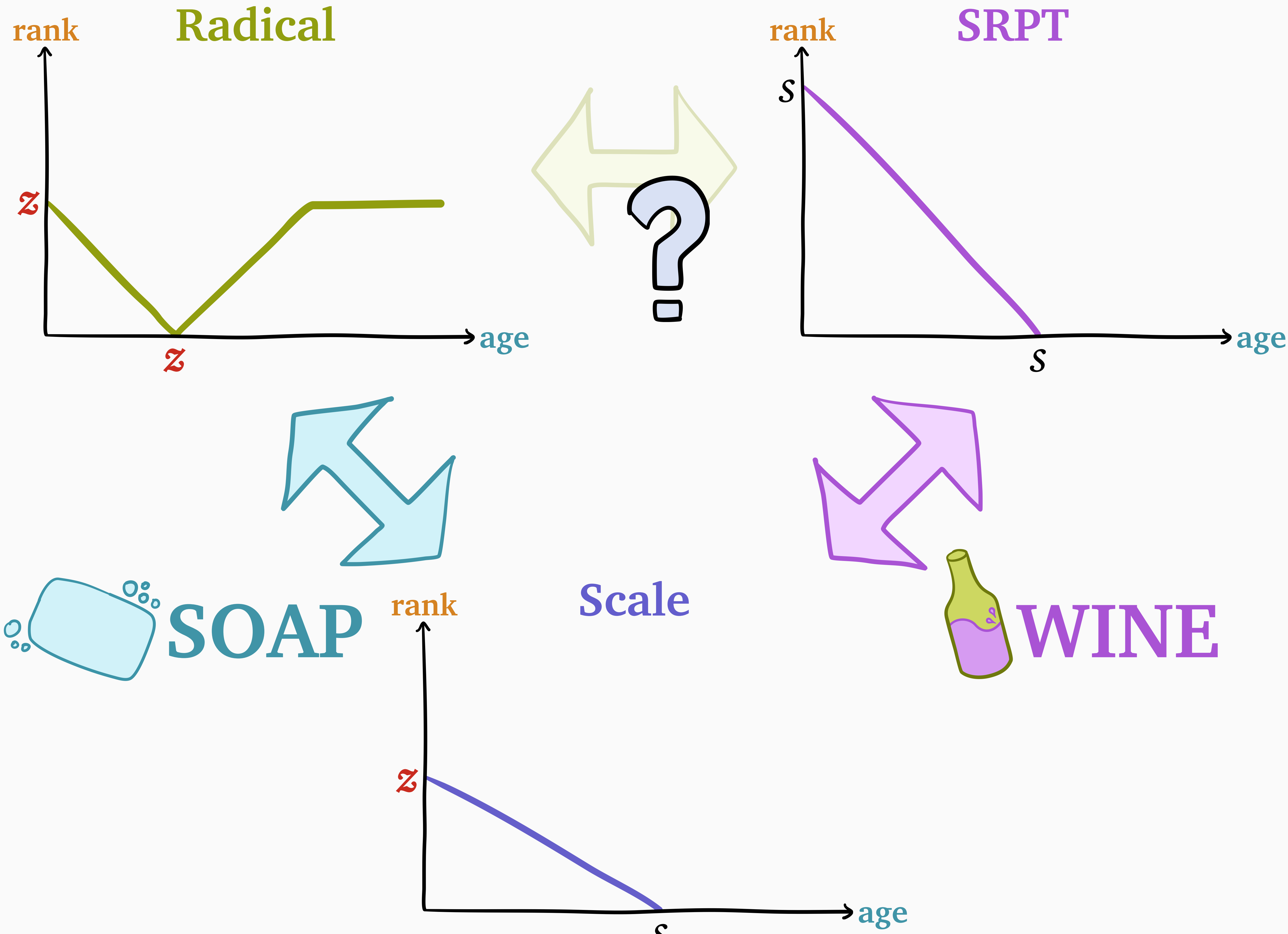
$$\mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}\Big[W_{\text{SRPT}}\Big(\tfrac{\alpha}{\beta} r\Big)\Big]$$

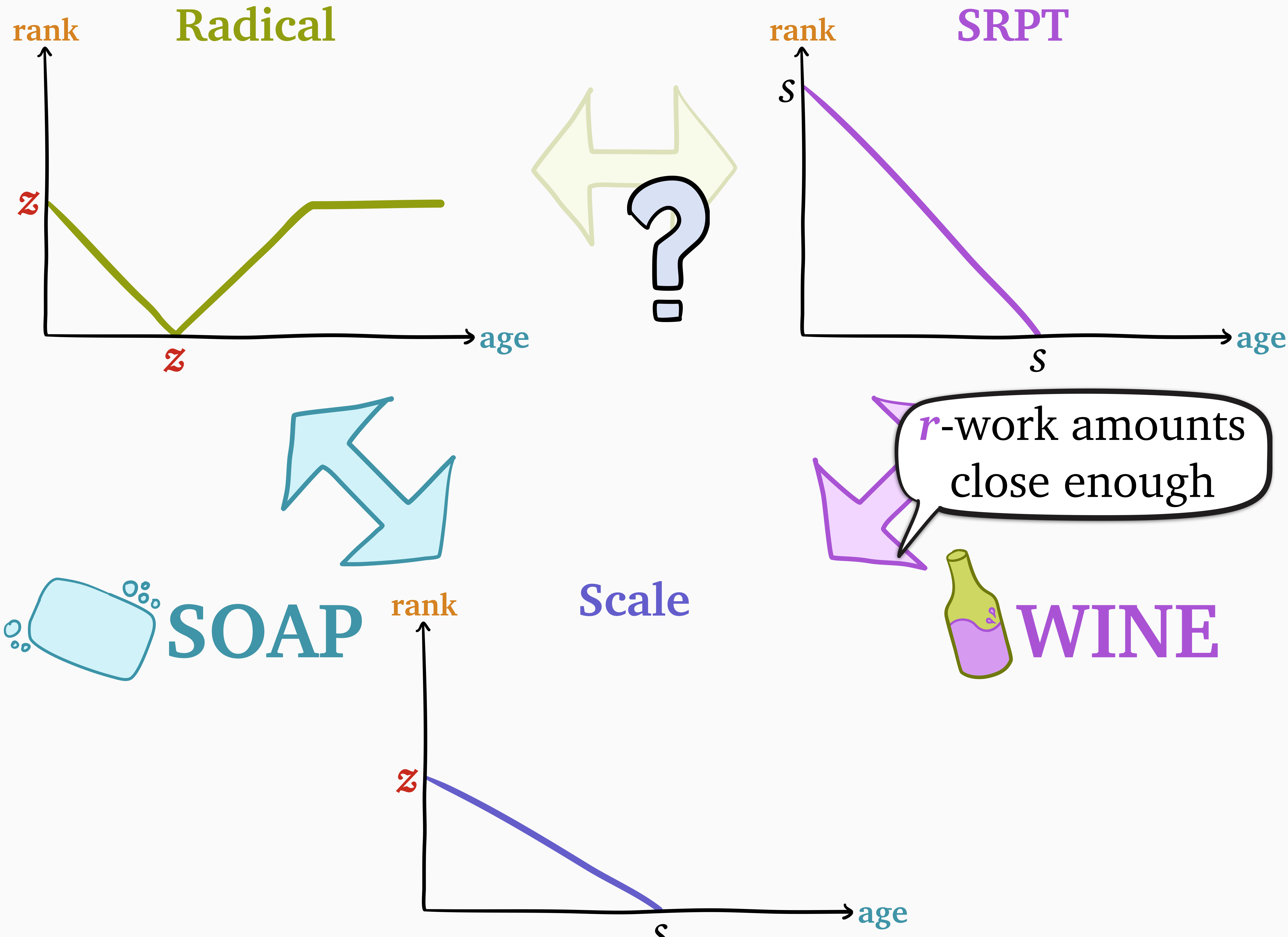
also holds with “noisy scaling”

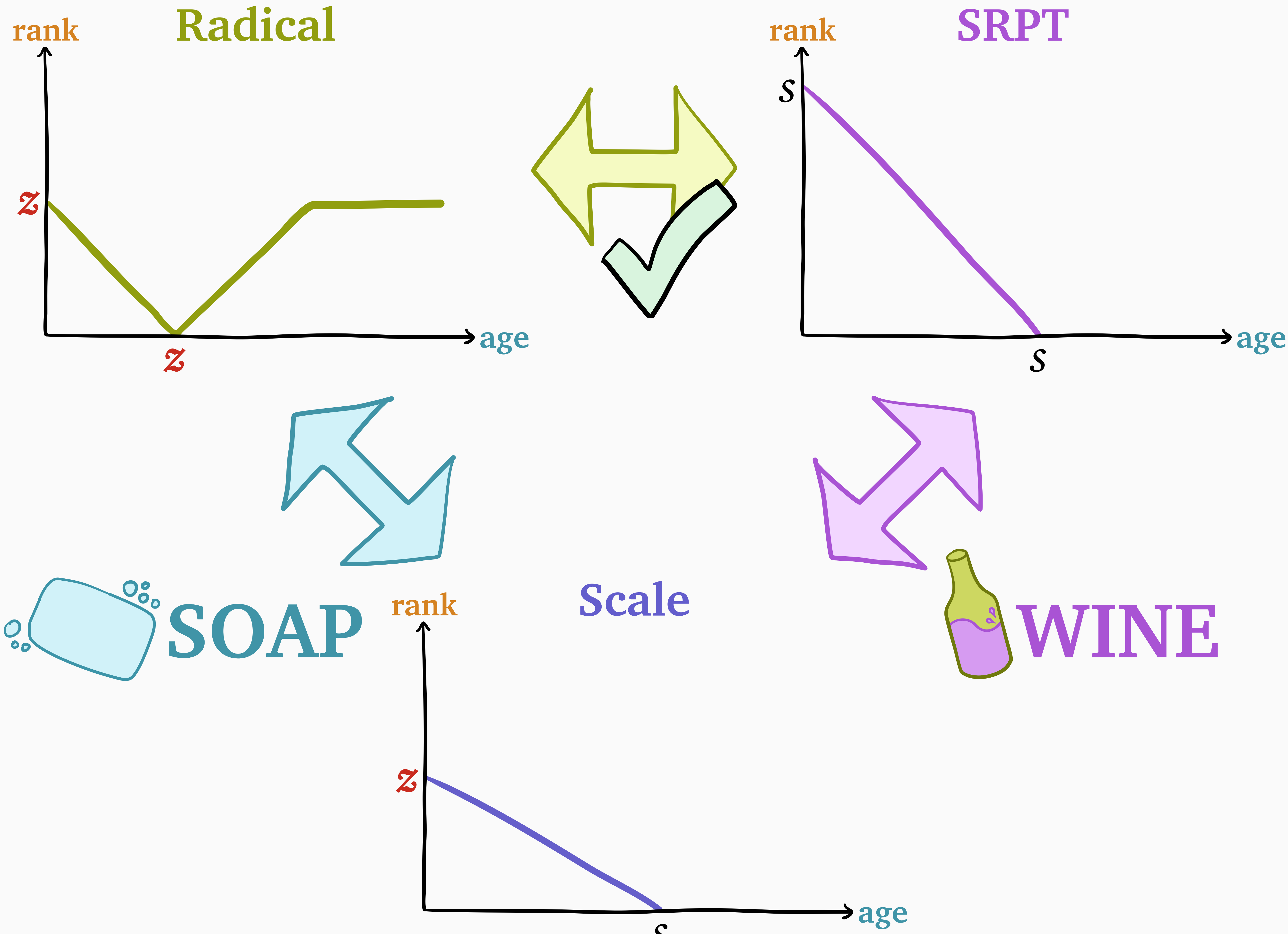
NEW!

Theorem:

$$\mathbf{E}[T_{\text{Scale}}] \le \tfrac{\alpha}{\beta}\mathbf{E}[T_{\text{SRPT}}]$$

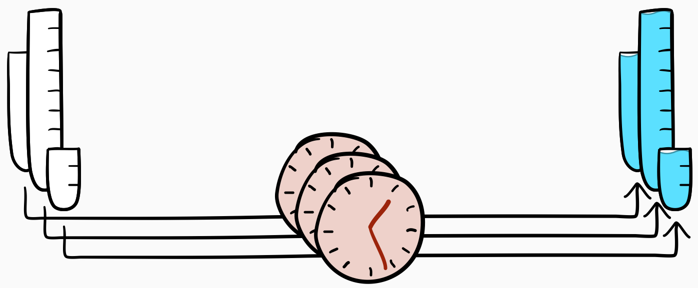

Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
WINE
Scale
rank
z
s
age

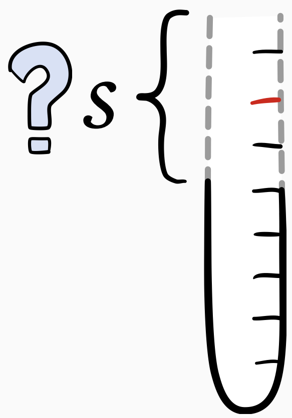

Radical
rank
z
z
age
SRPT
rank
s
s
age
r-work amounts close enough
SOAP
WINE
Scale
rank
z
s
age



Radical
rank
z
z
age
SRPT
rank
s
s
age
SOAP
Scale
rank
z
s
age
WINE

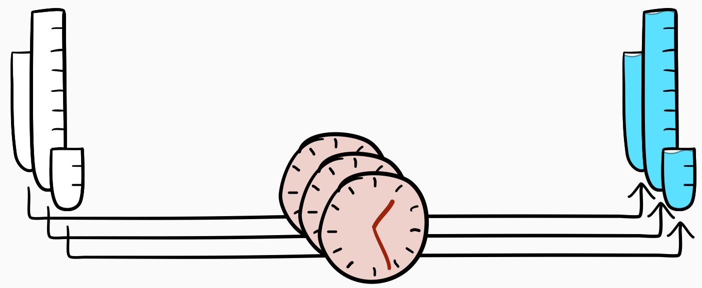
Problem: minimize $\mathbf{E}[T]$
with noisy information

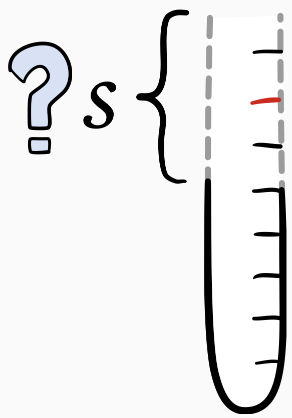




$s \quad z \in [\beta s, \alpha s]$

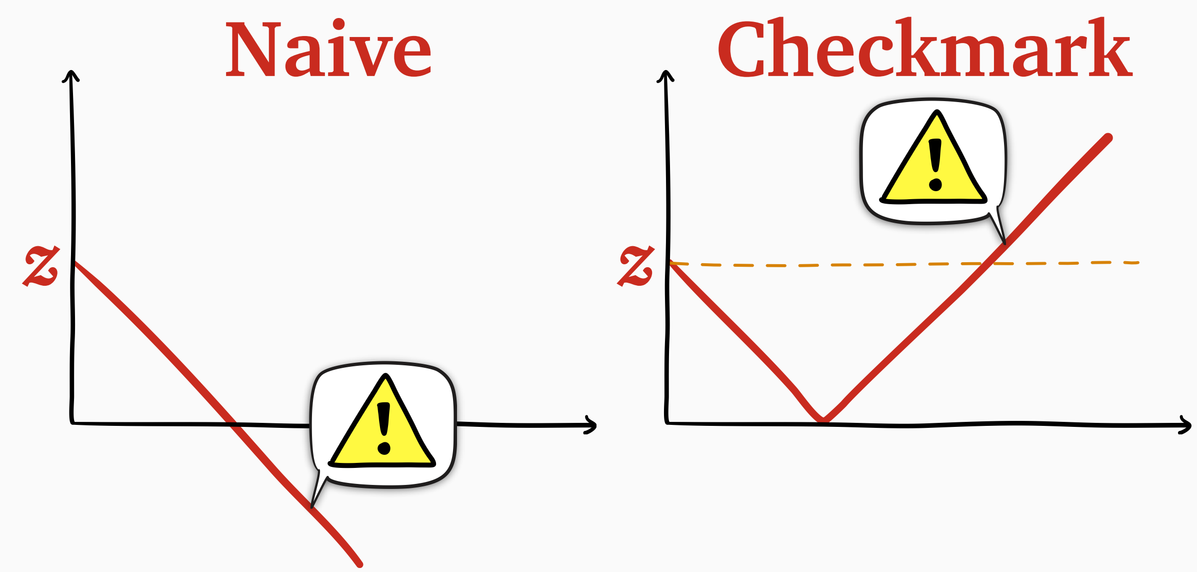
Problem: minimize $\mathbf{E}[T]$ with noisy information



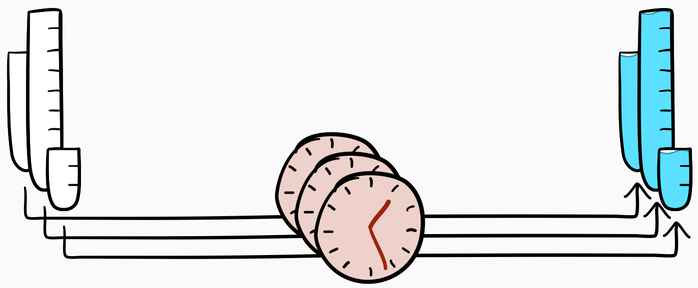


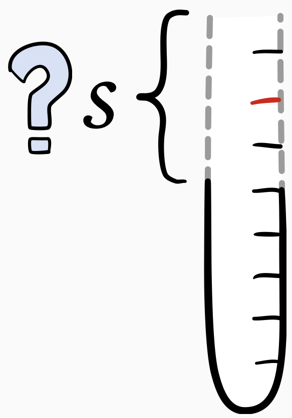
$z \in [\beta s, \alpha s]$

Obstacle: natural **rank** functions perform badly



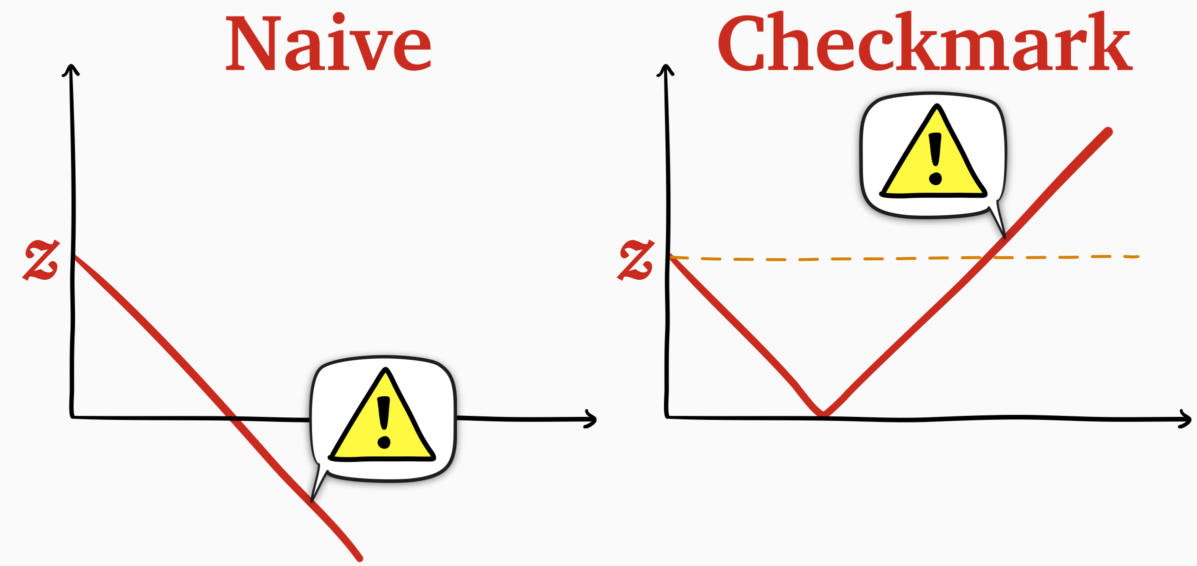
Problem: minimize $\mathbf{E}[T]$ with noisy information



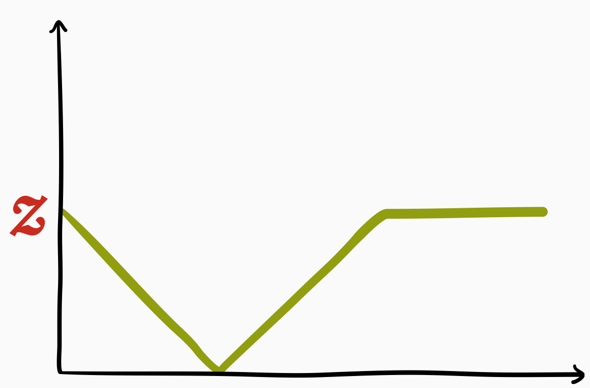


$z \in [\beta s, \alpha s]$

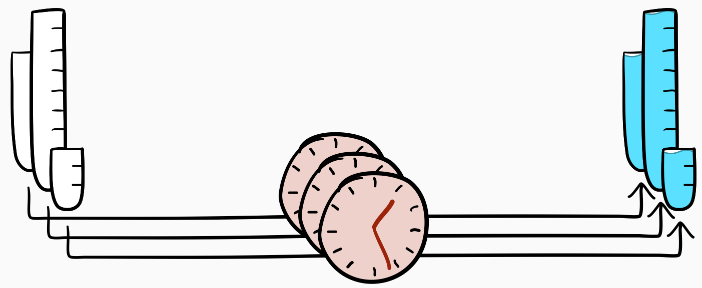
Obstacle: natural rank functions perform badly

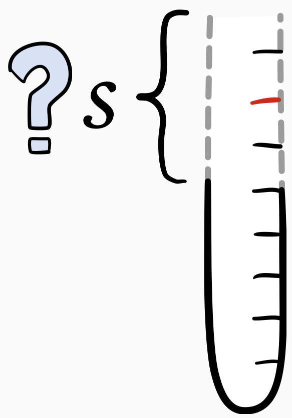


Solution: new policy, Radical, with provably bounded $\mathbf{E}[T]$



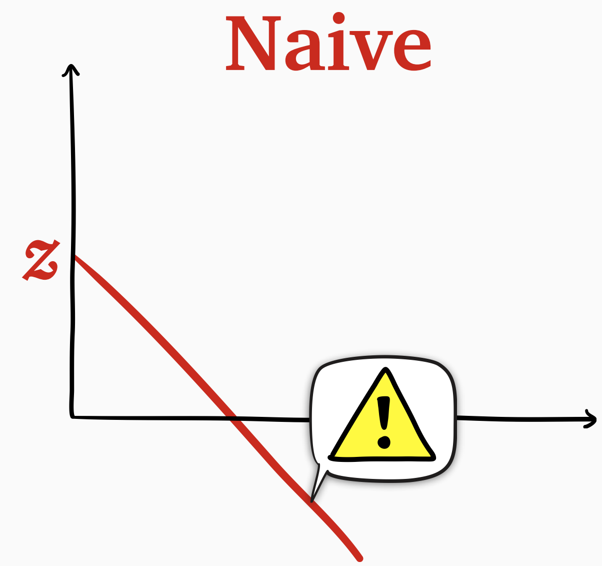
Problem: minimize $\mathbf{E}[T]$ with noisy information

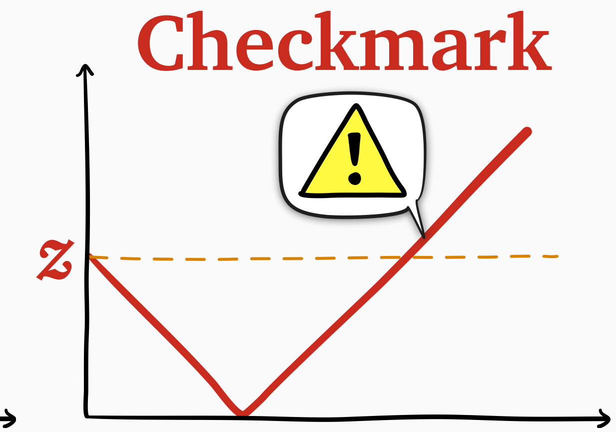




$z \in [\beta s, \alpha s]$

Obstacle: natural **rank** functions perform badly

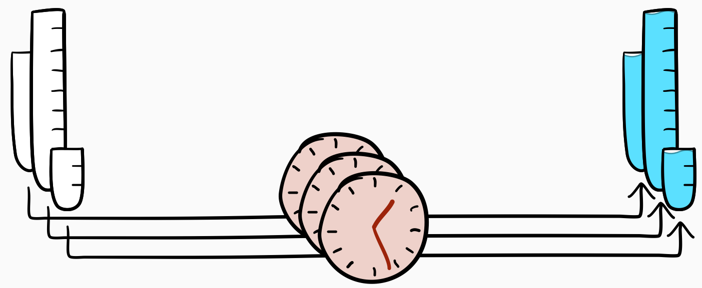


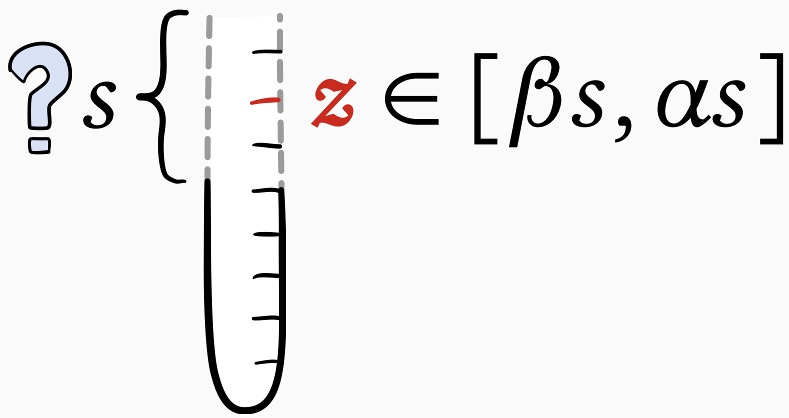


Solution: new policy, **Radical**, with provably bounded $\mathbf{E}[T]$

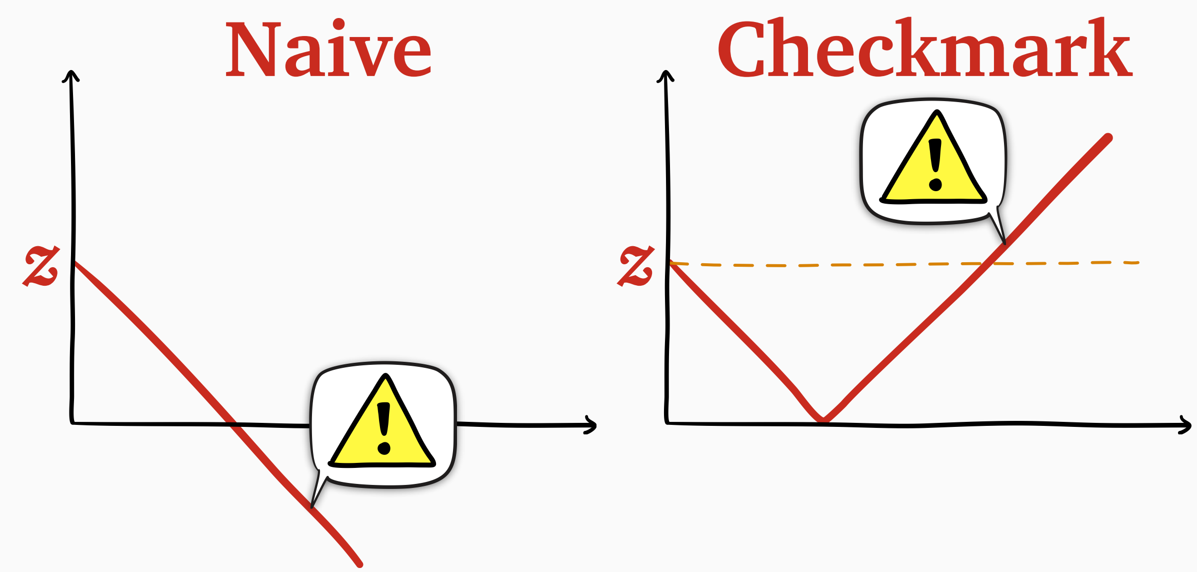
1-consistent,
3.5-graceful

Problem: minimize $\mathbf{E}[T]$ with noisy information

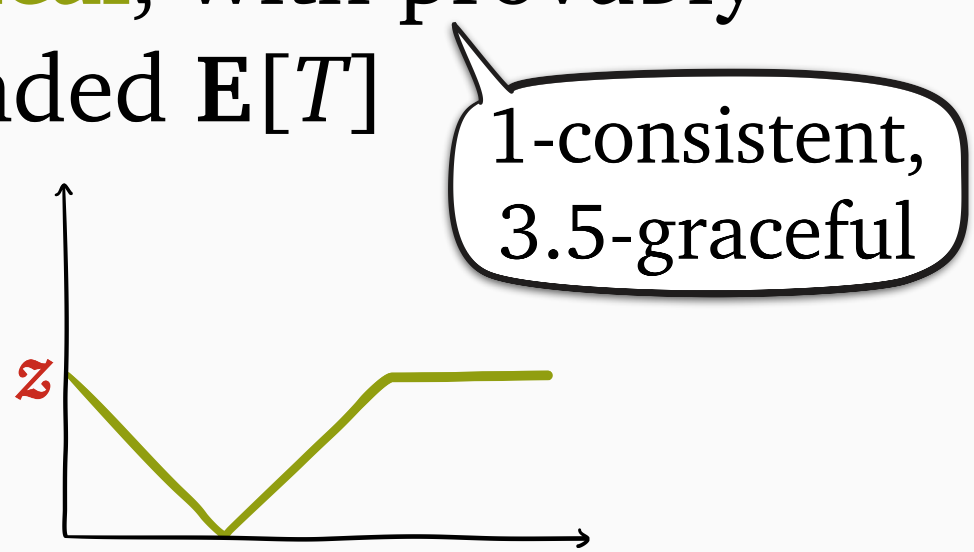




Obstacle: natural rank functions perform badly



Solution: new policy, Radical, with provably bounded $\mathbf{E}[T]$



Method: two new tools from queueing theory

Consistency-robustness tradeoff?

✓ C-consistent: $\dfrac{\mathbf{E}[T_{\mathbf{P}}]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \to C$ as $\alpha, \beta \to 1$

✓ G-graceful: $\dfrac{\mathbf{E}[T_{\mathbf{P}}]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \le G \cdot \dfrac{\alpha}{\beta}$ for all α, β

✗ R-robust: $\dfrac{\mathbf{E}[T_{\mathbf{P}}]}{\mathbf{E}[T_{\mathrm{SRPT}}]} \le R$ for all α, β

Our contribution: *first policy* ***P*** that's consistent and graceful

- $G = 3.5$
- $C = 1$

Consistency-robustness tradeoff?

✓ C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C \quad \text{as } \alpha, \beta \to 1$$

✓ G-graceful: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le G \cdot \frac{\alpha}{\beta} \quad \text{for all } \alpha, \beta$$

✗ R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le R \quad \text{for all } \alpha, \beta$$

Our contribution: *first policy* ***P*** that's consistent and graceful

- $G = 3.5$
- $C = 1$

Consistency-robustness tradeoff?

✓ C-consistent: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \to C \quad \text{as } \alpha, \beta \to 1$$

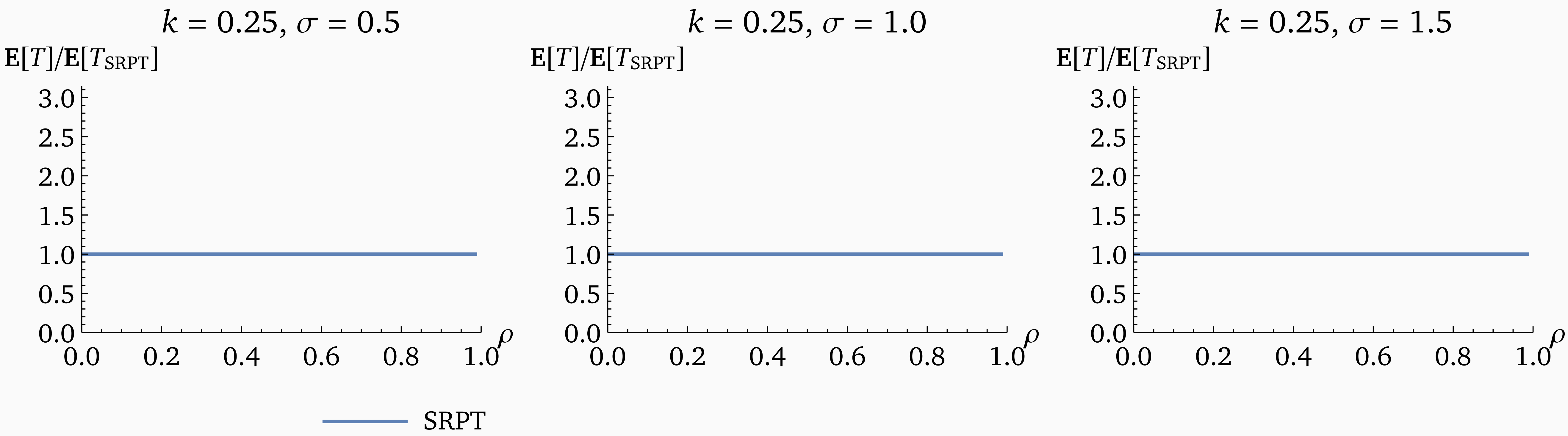
✓ G-graceful: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le G \cdot \frac{\alpha}{\beta} \quad \text{for all } \alpha, \beta$$

✗ R-robust: $$\frac{\mathbf{E}[T_P]}{\mathbf{E}[T_{\text{SRPT}}]} \le R \quad \text{for all } \alpha, \beta$$

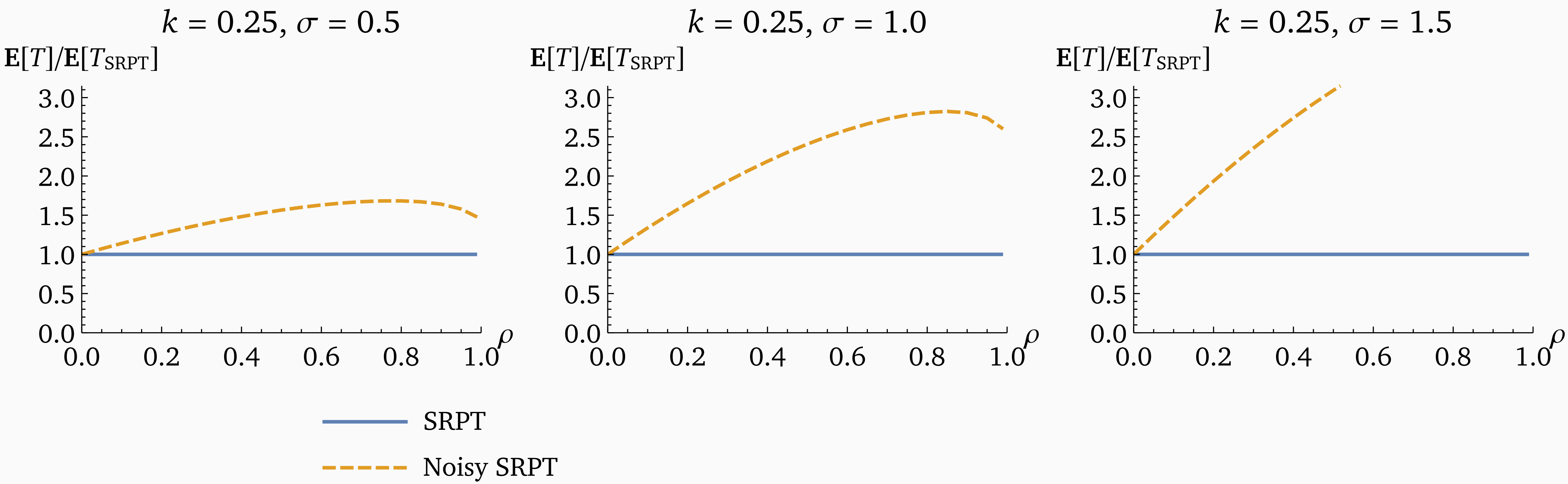
Our contribut ... P

that's consisten ...

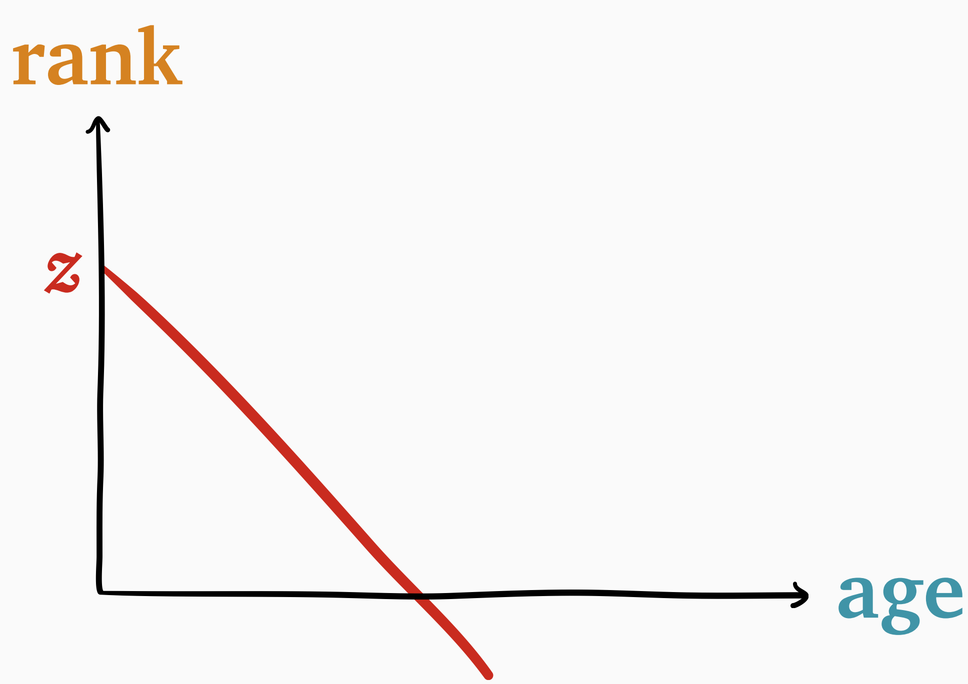
- $G = 3.5$
- $C = 1$

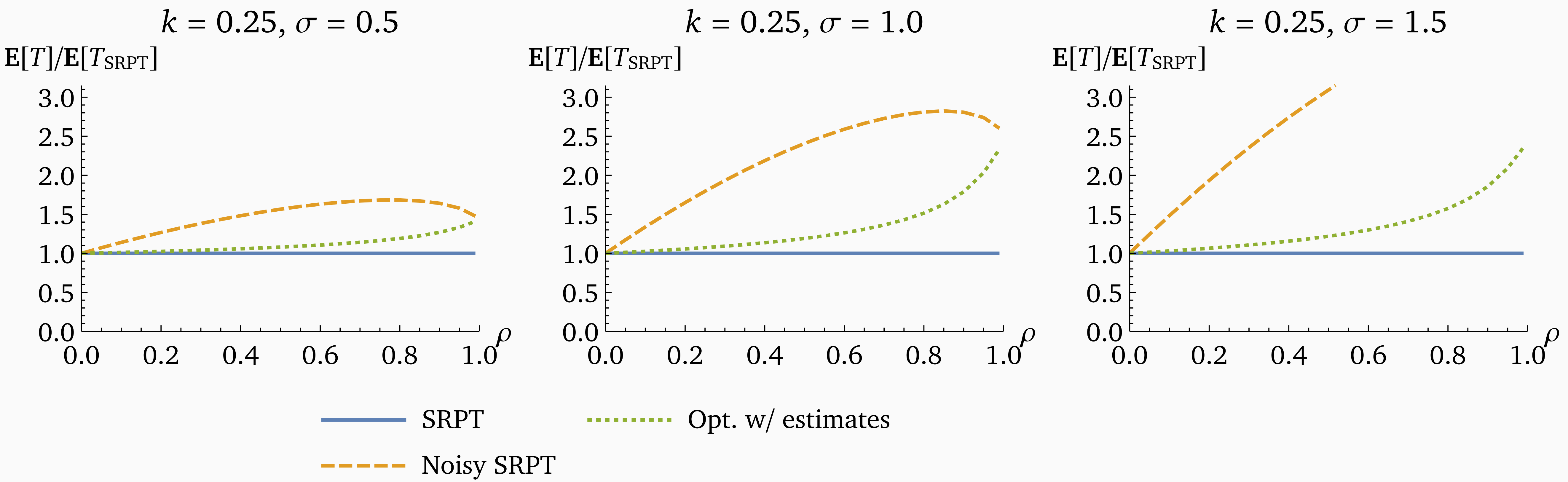


k = 0.25, σ = 0.5
k = 0.25, σ = 1.0
k = 0.25, σ = 1.5
E[T]/E[T_SRPT]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
ρ
SRPT

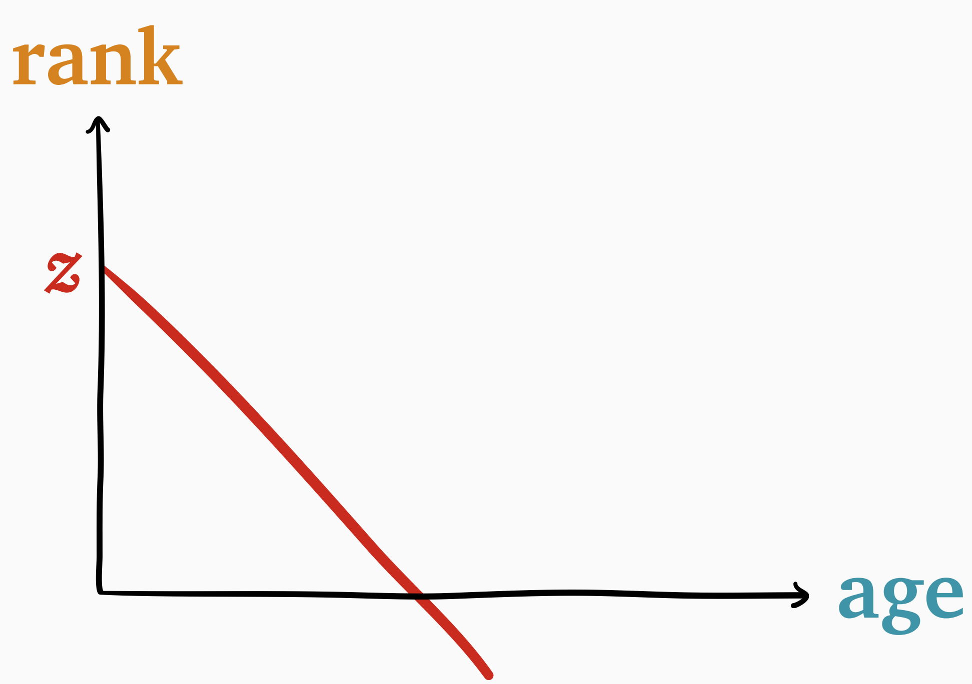


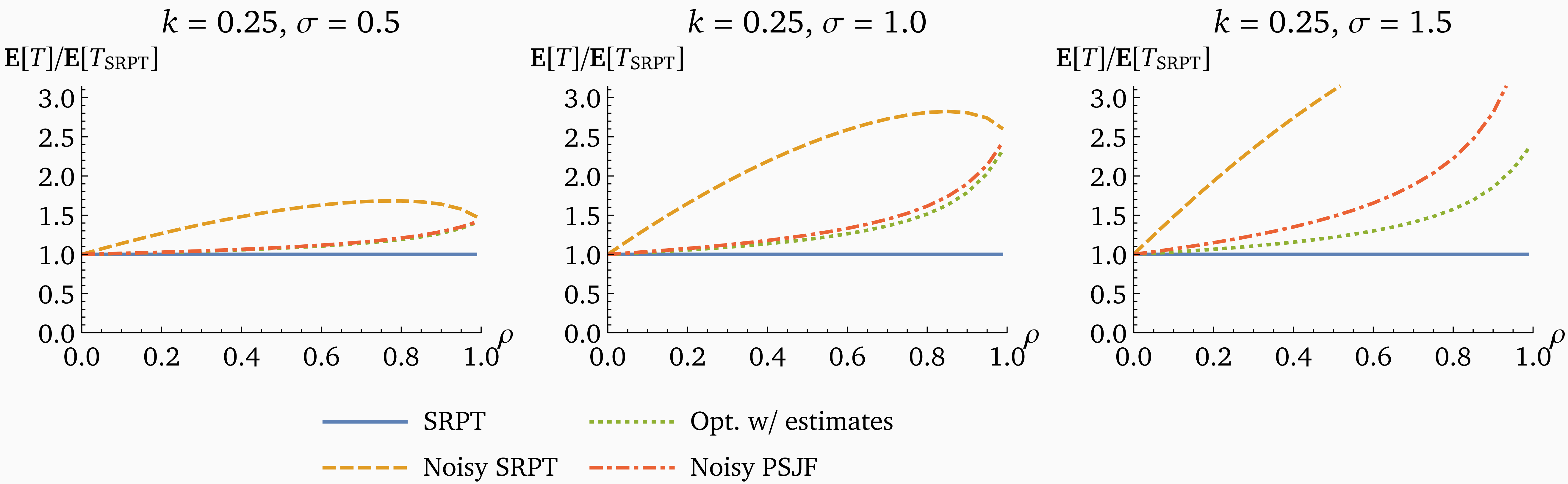
Noisy SRPT (Naive)



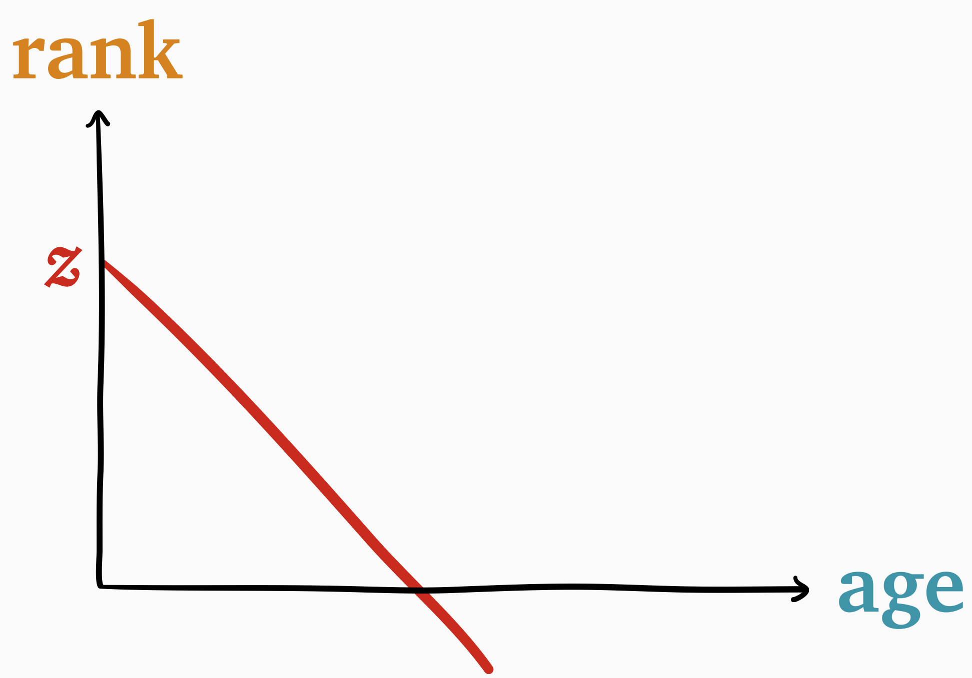


Noisy SRPT (Naive)

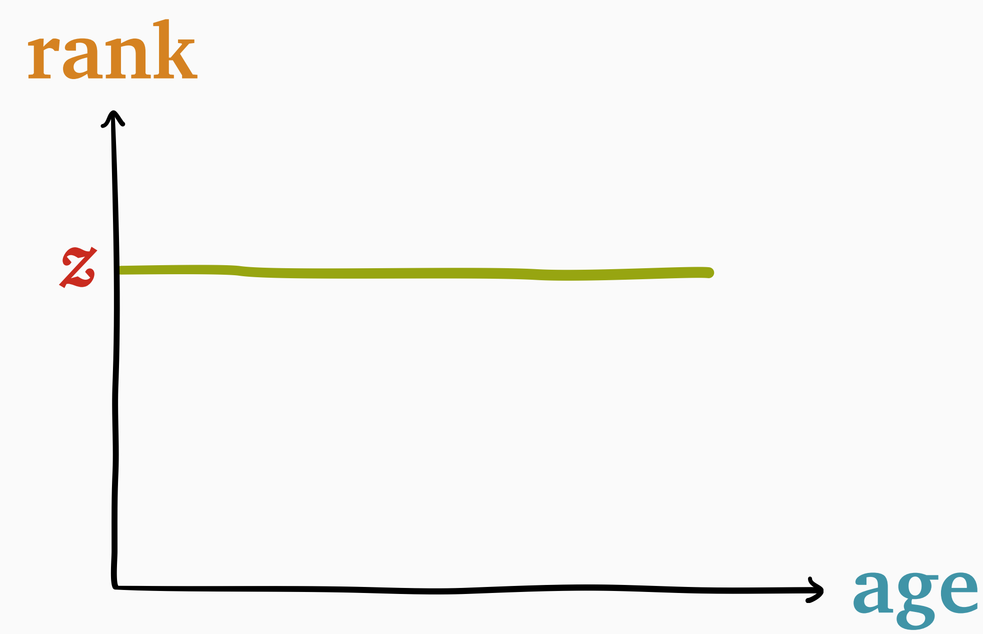


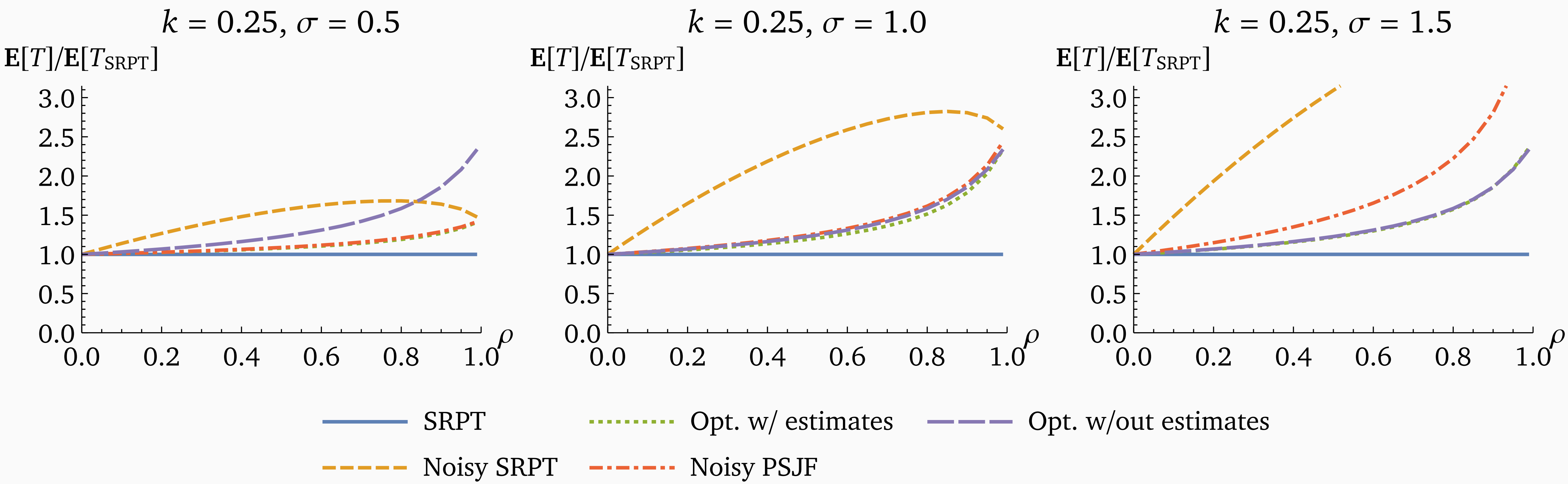


Noisy SRPT (Naive)

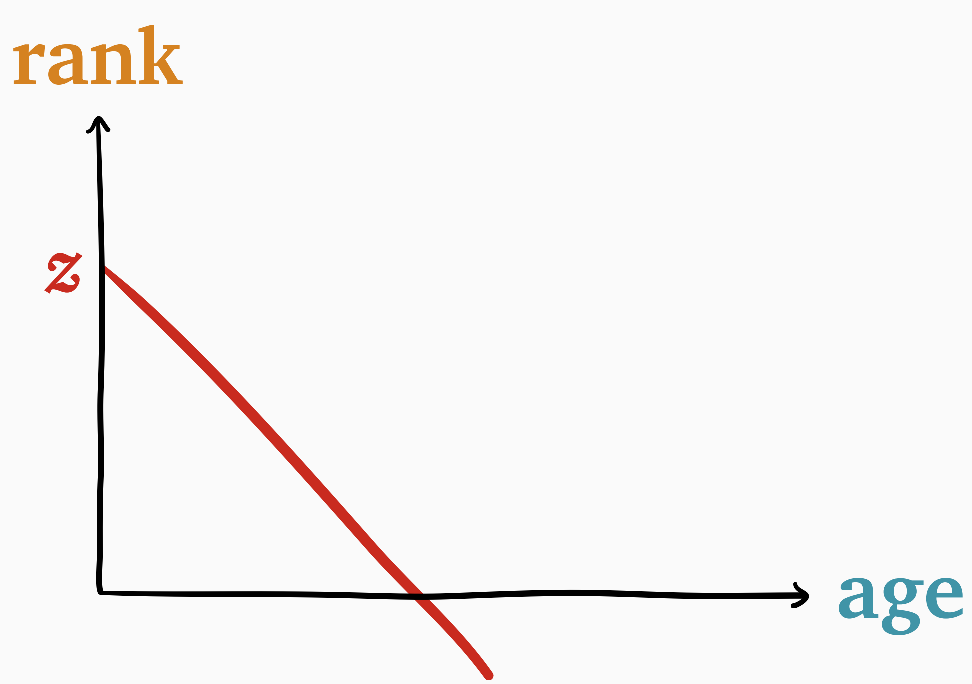


Noisy PSJF (1.5-graceful)

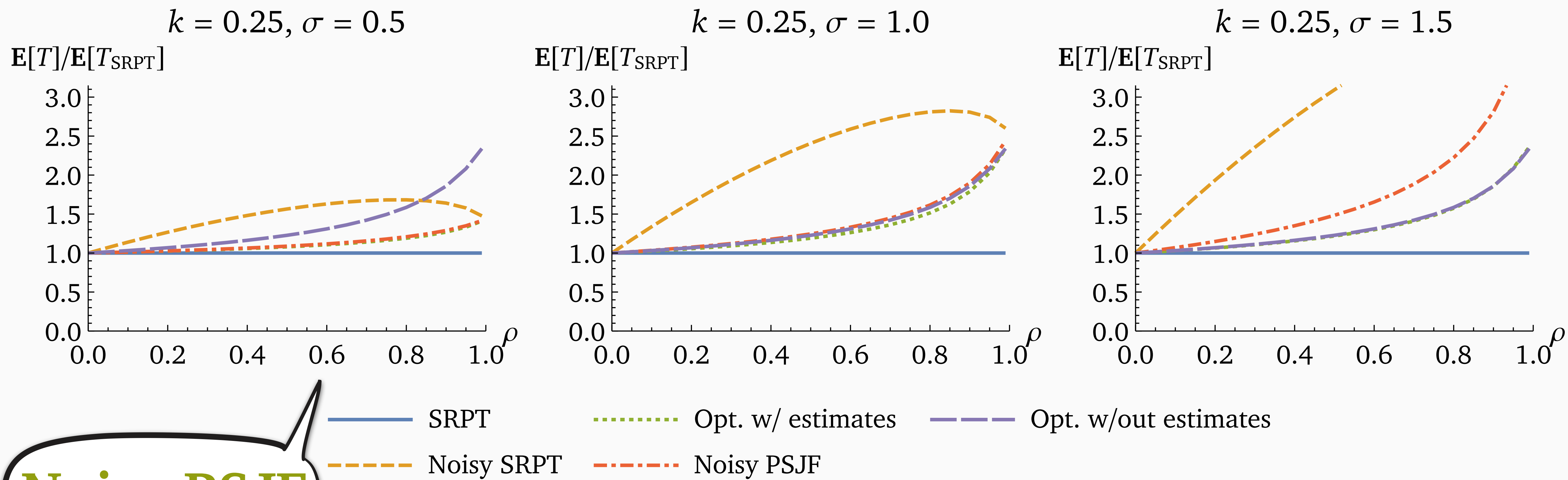




Noisy SRPT (Naive)

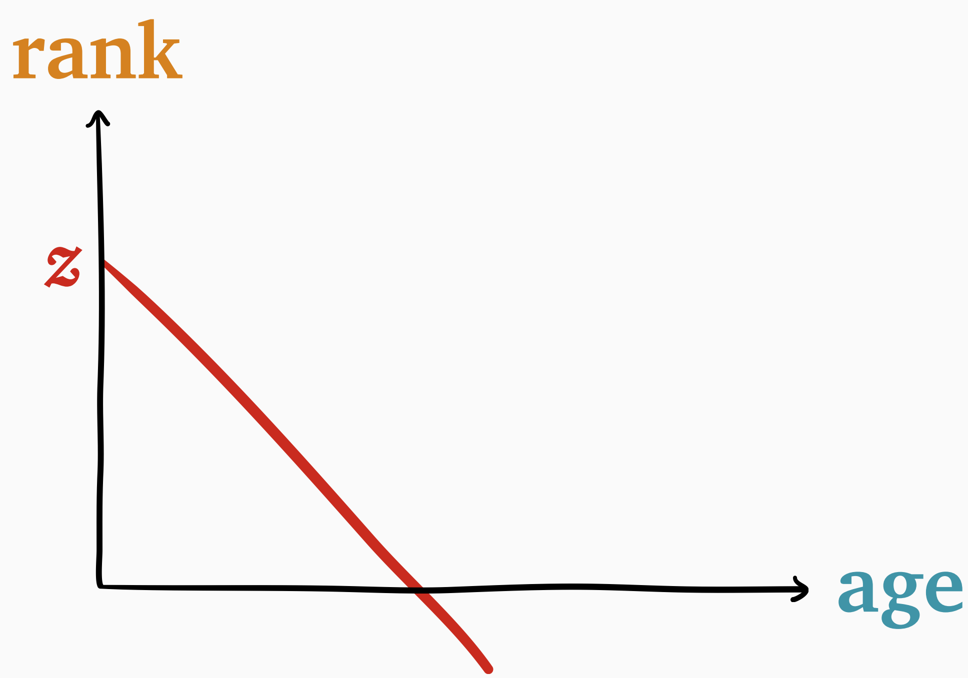


Noisy PSJF (1.5-graceful)

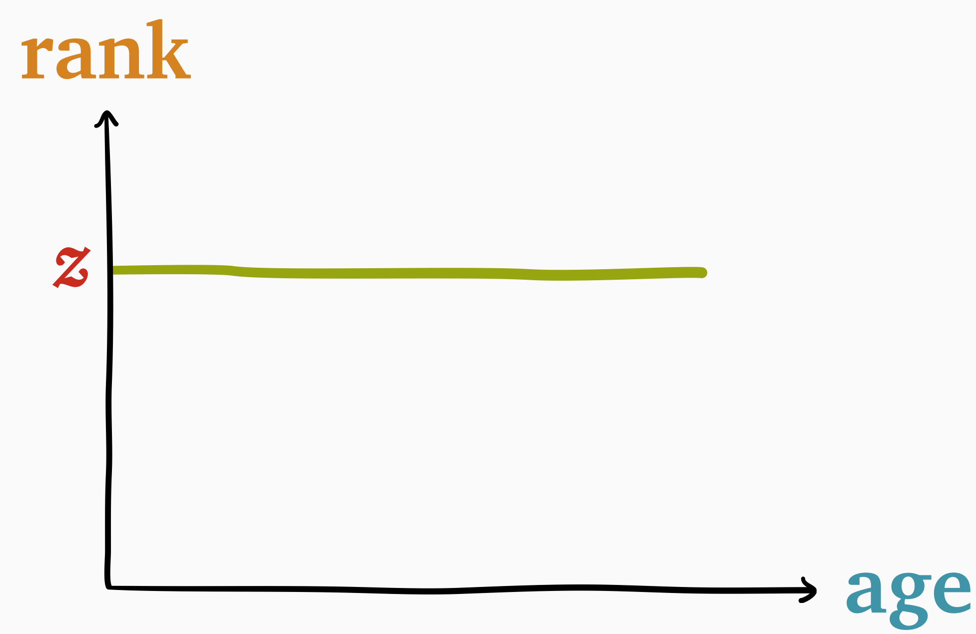


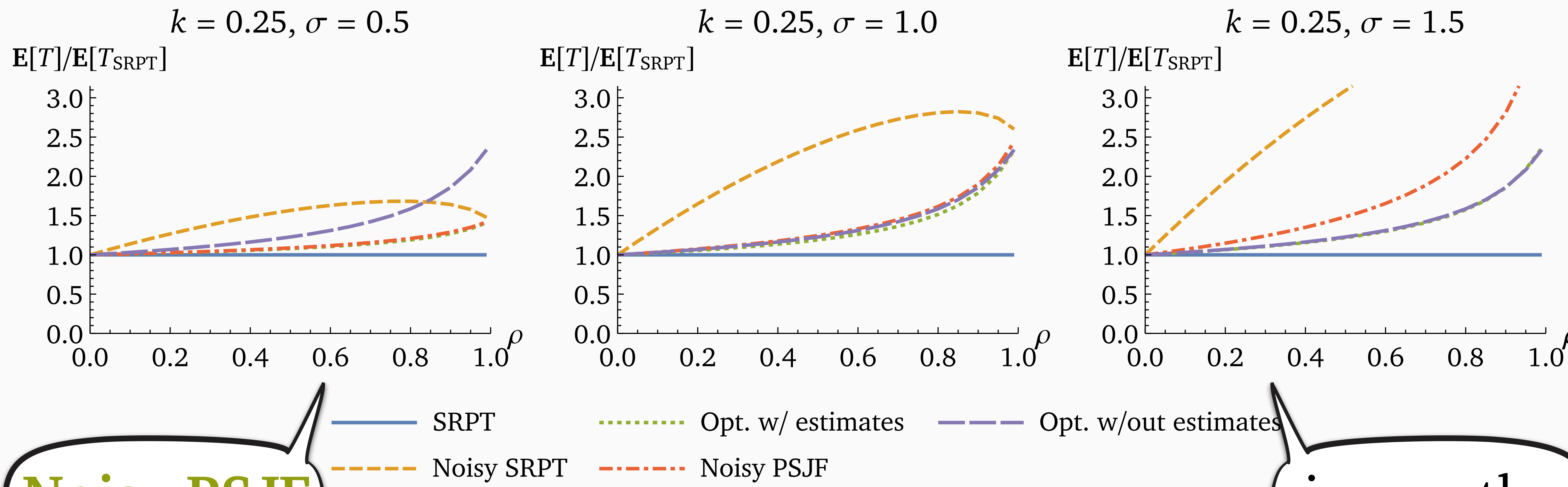
Noisy PSJF suffices

Noisy SRPT (Naive)



Noisy PSJF (1.5-graceful)

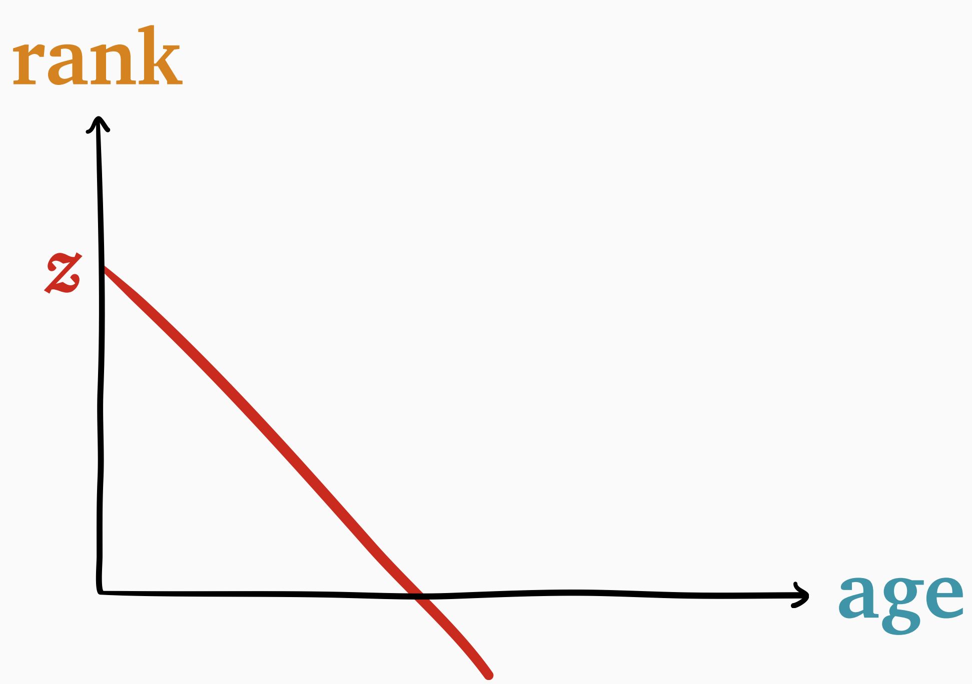




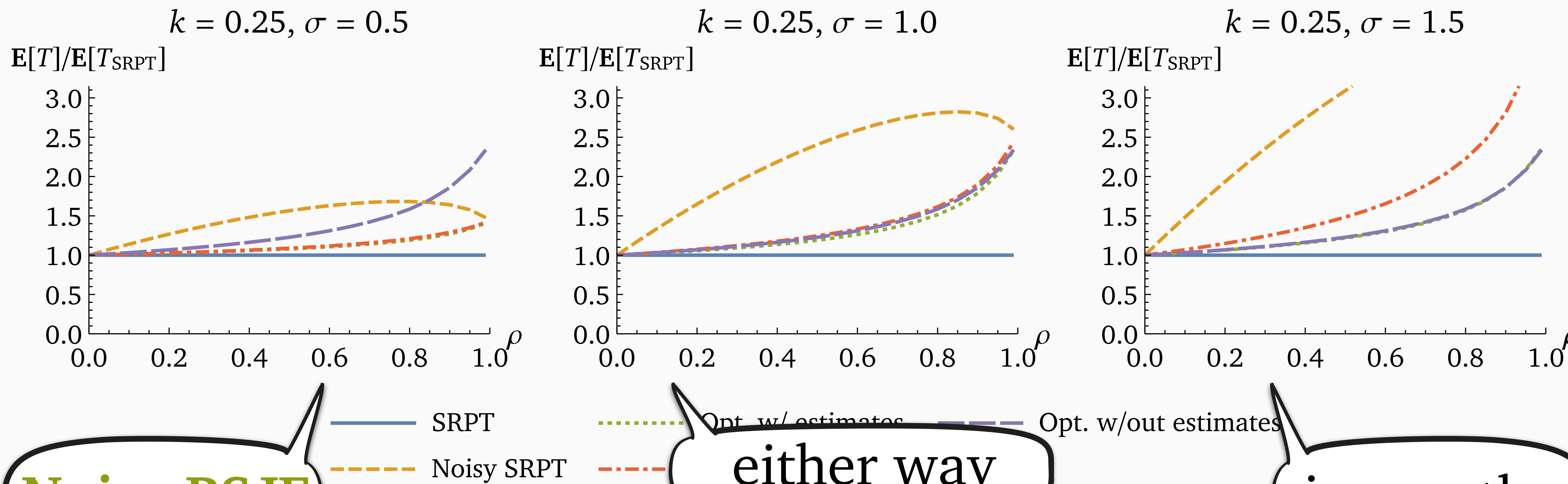
Noisy PSJF suffices

ignore the estimates

Noisy SRPT (Naive)



Noisy PSJF (1.5-graceful)

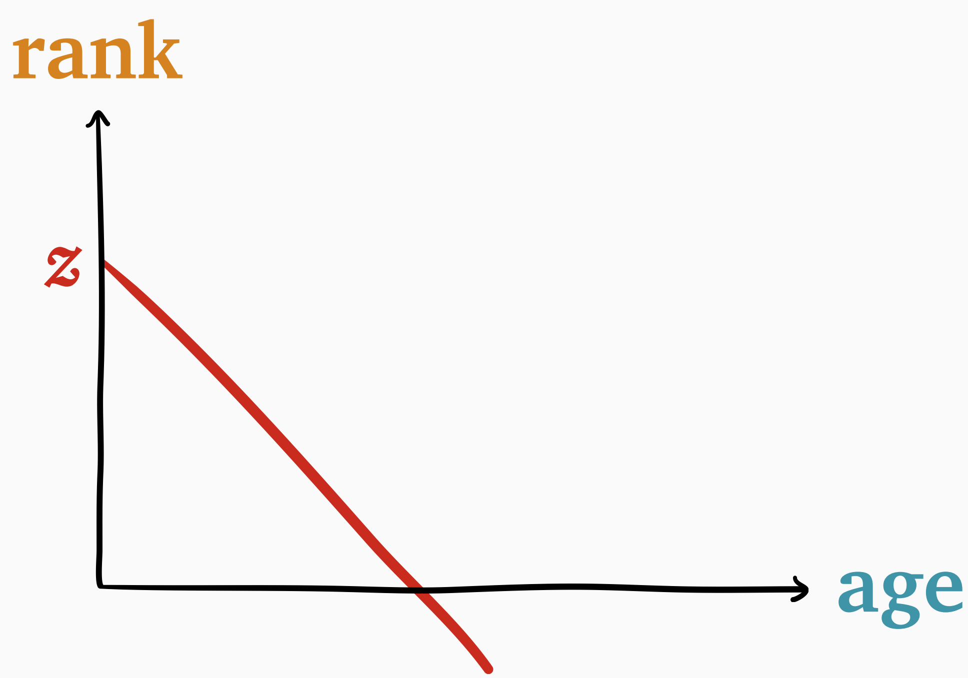


Noisy PSJF suffices

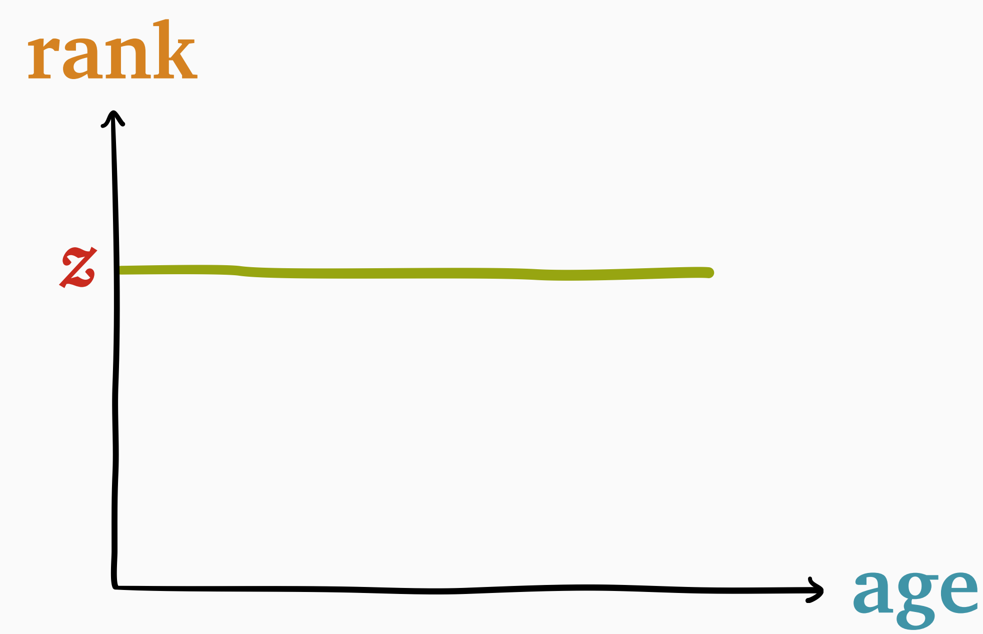
either way

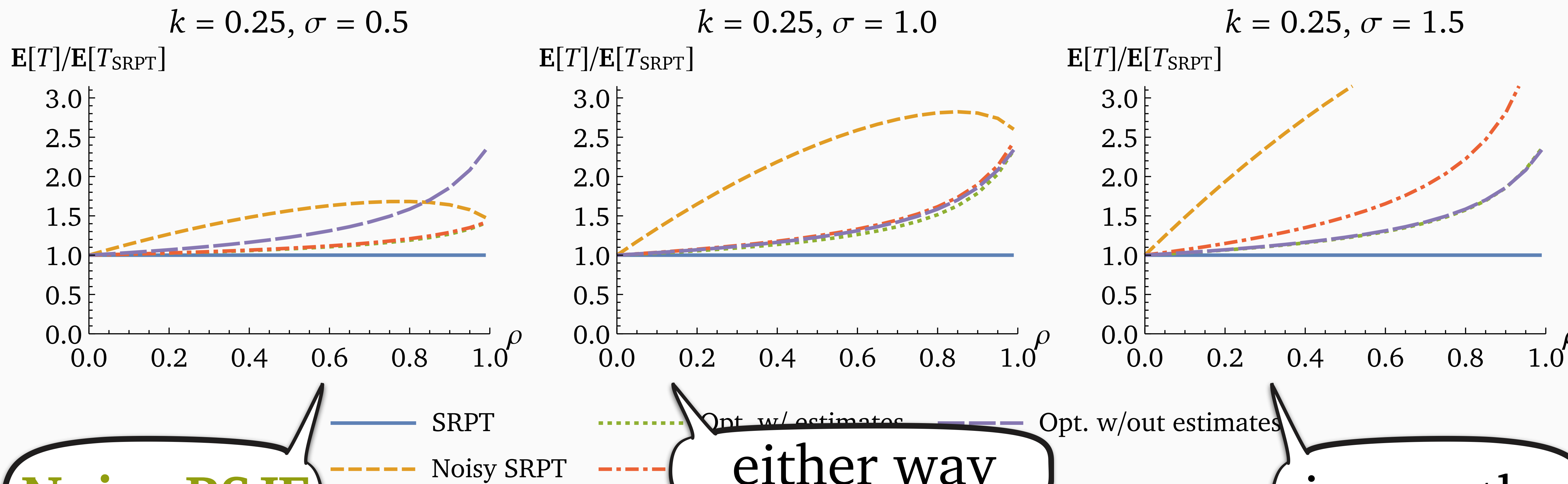
ignore the estimates

Noisy SRPT (Naive)



Noisy PSJF (1.5-graceful)





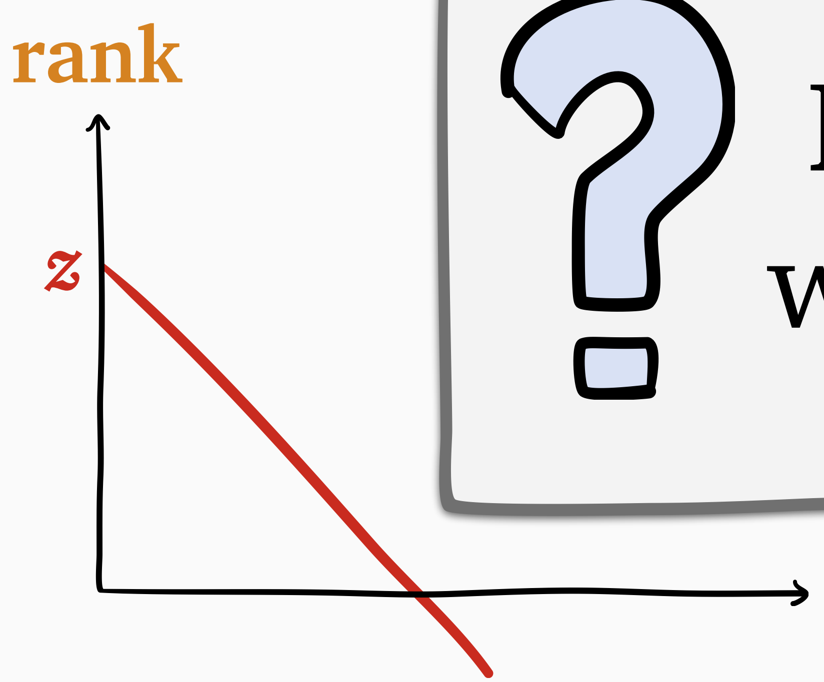
Noisy PSJF suffices

either way

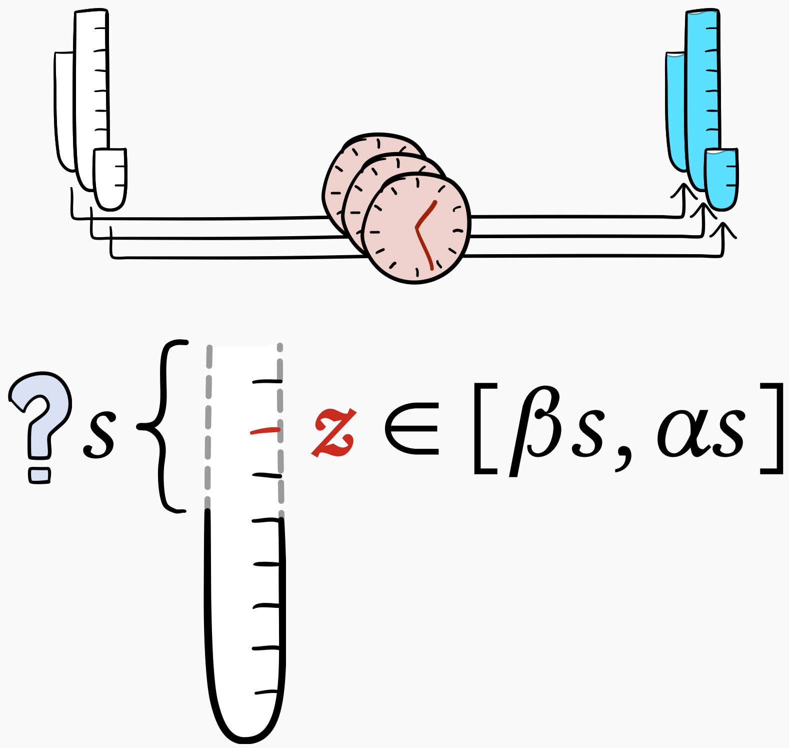
ignore the estimates

Noisy SR … .5-graceful)

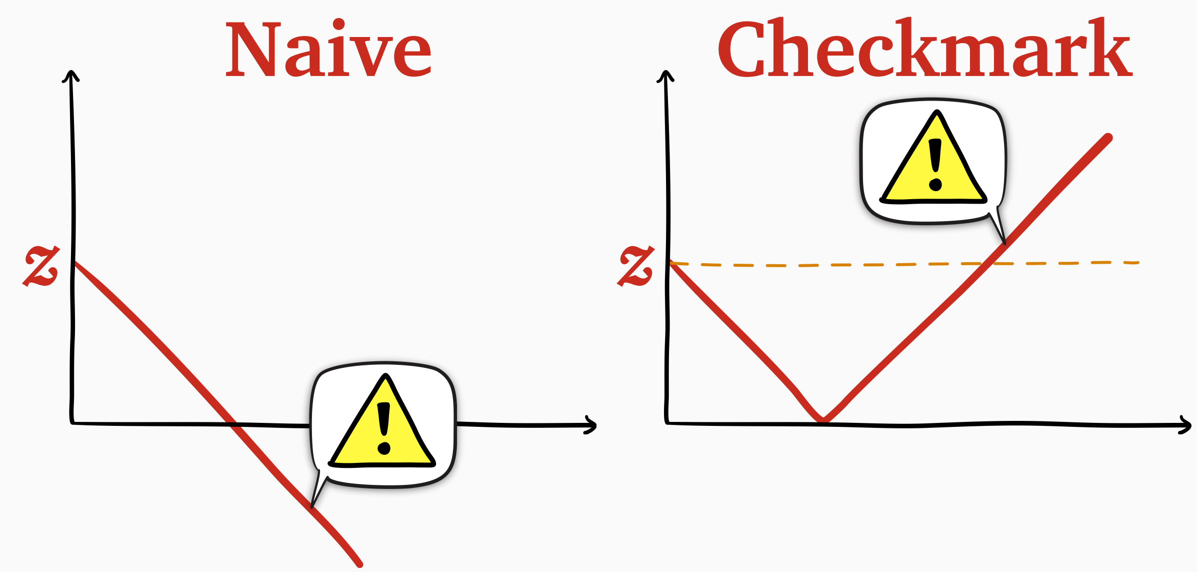
Mix Noisy PSJF with Opt. w/out?



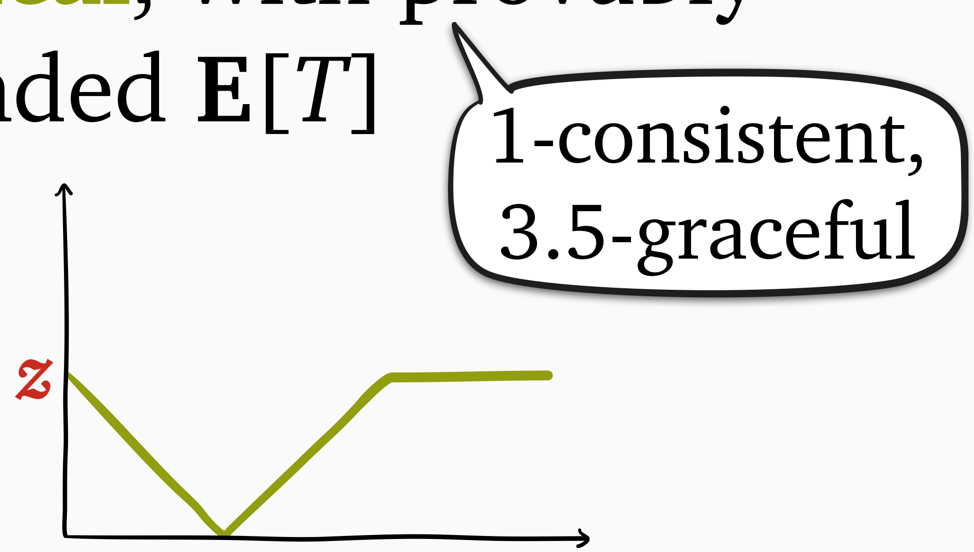
Problem: minimize $\mathbf{E}[T]$ with noisy information



Obstacle: natural rank functions perform badly



Solution: new policy, Radical, with provably bounded $\mathbf{E}[T]$



Method: two new tools from queueing theory

SOAP
& WINE

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathbf{Scale}}(\boldsymbol{r})] \le \mathbf{E}\left[W_{\mathbf{SRPT}}\left(\frac{\alpha}{\beta}\boldsymbol{r}\right)\right]$$

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta}r\right)\right]$$

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta} r\right)\right]$$

Key steps:

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathbf{SRPT}}(\boldsymbol{r})] \le \mathbf{E}[W_{\mathbf{Scale}}(\boldsymbol{r})] \le \mathbf{E}\left[W_{\mathbf{SRPT}}\left(\frac{\alpha}{\beta}\boldsymbol{r}\right)\right]$$

Key steps:

1. **SRPT** minimizes mean $\boldsymbol{r}$-work

Proof sketch

Lemma:

$$\mathbf{E}[W_{\text{SRPT}}(r)] \le \mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}\left[W_{\text{SRPT}}\left(\tfrac{\alpha}{\beta} r\right)\right]$$

Key steps:

1. **SRPT** minimizes mean r-work

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta} r\right)\right]$$

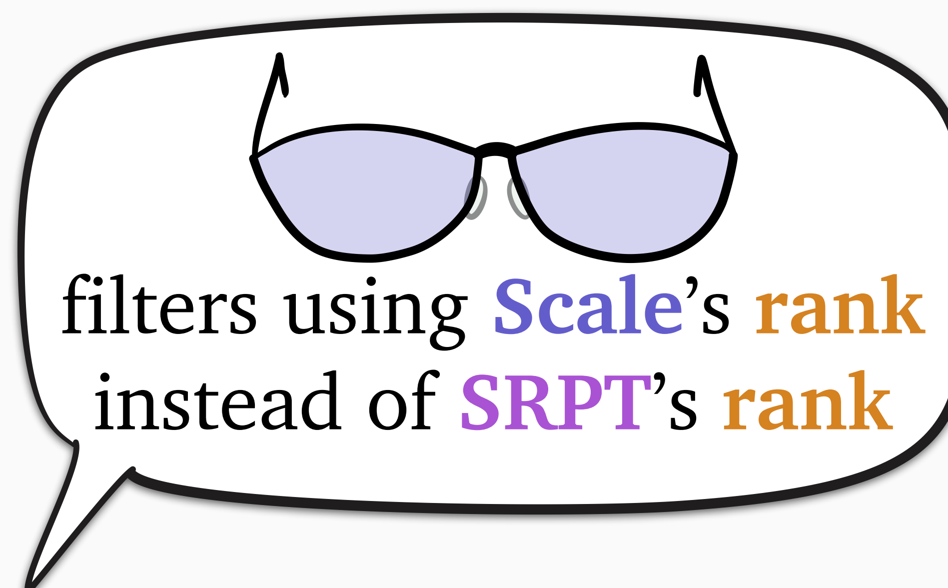
Key steps:

1. **SRPT** minimizes mean r-work
2. **Scale** minimizes mean **noise-scaled**-r-work

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta} r\right)\right]$$



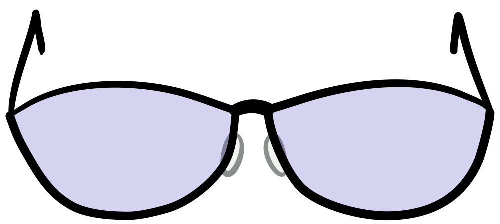
Key steps:

1. **SRPT** minimizes mean r-work
2. **Scale** minimizes mean **noise-scaled**-r-work

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\tfrac{\alpha}{\beta} r\right)\right]$$



filters using **Scale**'s **rank** instead of **SRPT**'s **rank**

Key steps:

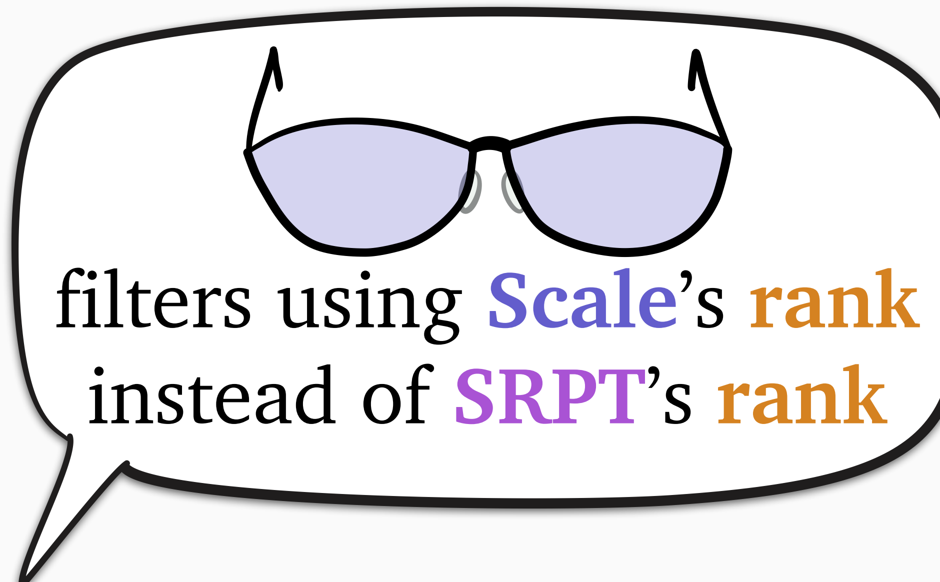
1. **SRPT** minimizes mean r-work
2. **Scale** minimizes mean **noise-scaled**-r-work
3. Under any policy,

Proof sketch

Lemma:

$$\mathbf{E}[W_{\mathrm{SRPT}}(r)] \le \mathbf{E}[W_{\mathrm{Scale}}(r)] \le \mathbf{E}\left[W_{\mathrm{SRPT}}\left(\frac{\alpha}{\beta} r\right)\right]$$

Key steps:



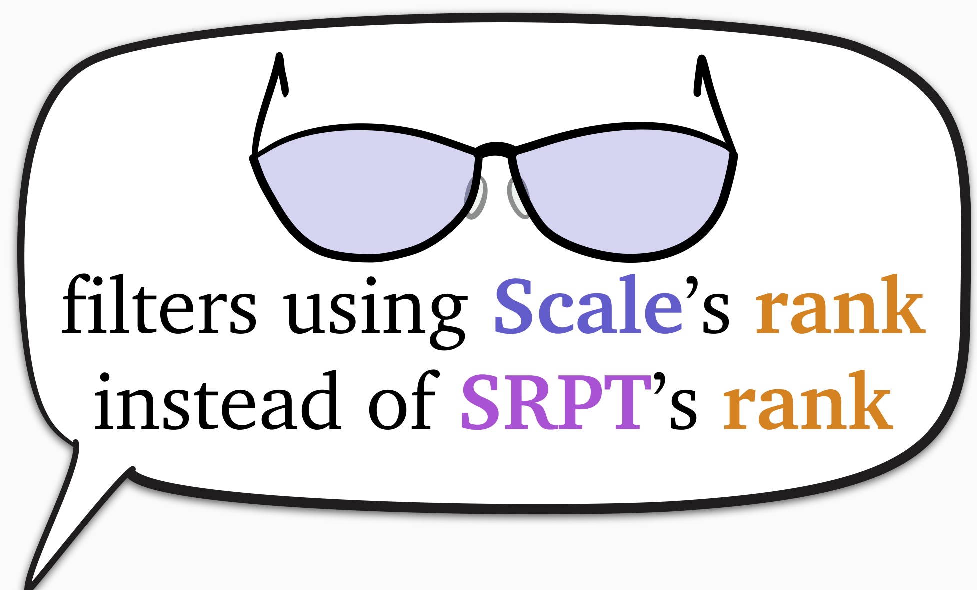
1. SRPT minimizes mean r-work
2. Scale minimizes mean **noise-scaled**-r-work
3. Under any policy,

   r-work ≤ **noise-scaled**-αr-work ≤ $\frac{\alpha}{\beta} r$-work

Proof sketch

Lemma:


$$\mathbf{E}[W_{\text{SRPT}}(r)] \le \mathbf{E}[W_{\text{Scale}}(r)] \le \mathbf{E}\left[W_{\text{SRPT}}\left(\tfrac{\alpha}{\beta} r\right)\right]$$



Key steps:

1. SRPT minimizes mean r-work
2. Scale minimizes mean noise-scaled-r-work
3. Under any policy,

   r-work $\le$ noise-scaled-αr-work $\le \frac{\alpha}{\beta} r$-work