

DEUCE

A Lightweight UI For Structured Editing

Brian Hempel, Justin Lubin, Grace Lu, and Ravi Chugh



THE UNIVERSITY OF
CHICAGO

**Unstructured text
is clumsy**

ñjò-|I[AÍÕÓnLE...@iq6Èûâ8€tù.À, :◇ÿc; :2πU" LZ™Yz±i
ÒJ#|suurü={~'İðÀûÛ~«<>'bN™‡EvPèÆêÛN, Ω≥>n°fTπc"
ª√≤õÕ...+Tã@\$TwhJ<İÚ%(8"İÛFz5f:Ø<iI-^ð9*™s*~'İuj
±[î!ùØ\$3áUà·Î·'Æ◇≥≥'ë\İİ\Ä"dfi<u¥v£ıÿzáo(cõ÷◇·≥
'o¶ç±d, ¥fl3°βH?_i~'√<Æzç§"ÀK*Ã«?<`ö0æN(HRXU^-â^ác
ÜİIzÅ:¶idSYì0πJFíi"÷∞:~"†V»',,, "ôdâuTΩ{<héäÄË);
o,U)è#æ·Jªfi†İÕí2ÜdHÃì2°@'jÀ=·Ú°@*qx]bèuSzÃµxKø
&PkY·VòVçÌ|fÉIi~Ê»ùòúó2i>{jÛdÜβ≤'ΩC;L·}n'I"·T≤

ñjò-|I[AÍÕÓnLE...@iq6Èûâ8€tù.À, :◇ÿc; :2πUˆLZ™Yz±i
ÒJ#|suurü={~'İðÀûÛ~«<>`bN™†EvPèÆêÛN, Ω≥>n°fTπc"
ª√≤õÕ...+Tã@\$TwhJ<İÚ%(8"İÛFz5f:Ø<iI-^ð9*™s*ˆ`İuj
±[î!ùØ\$3áUà·Î·'Æ◇≥≥'ë\İİ\Ä"dfi<u¥v£ıÿzáo(cõ÷◇·≥
'o¶ç±d, ¥fl3°βH?_i~'√<Æzç§"ÀK*Ã«?<`öÖæN(HRXUˆâˆá
ÜİIzÅ:¶idSÿì0πJFíi"÷∞:ˆ"†V»',,, "ôdâuTΩ{<héäÄË);
o,U)è#æ·Jªfi†İÕí2ÜdHÃì2°@'jÀ=·Ú°@*qx]bèuSzÃµxKø
&PkY·VòVçÌ|fÉIi~Ê»ùòúó2i>{jÛdÜß≤'ΩC;L·}n'I"·T≤

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,
Doom'd for a certain term to walk the night,
And for the day confin'd to fast in fires,
Till the foul crimes done in my days of nature
Are burnt and purged away. But that I am forbid
To tell the secrets of my prison-house,
I could a tale unfold whose lightest word
Would harrow up thy soul; freeze thy young blood,

ñjò-|I[AÍÕÓnLE...@iq6Èûâ8€tù.À, :◇ÿc; :2πU" LZ™Yz±i
ÒJ#|suurü={~'İðÀûÛ~«<>`bN™‡EvPèÆêÛN, Ω≥>n°fTπc"
ª√≤õÕ...+Tã@\$TwhJ<İÚ%(8"İÛFz5f:Ø<iI-^ð9*™s*~`İuj
±[î!ùØ\$3áUà·Î·'Æ◇≥≥'ë\İİ\Ä"dfi<u¥v£ıÿzáo(cõ÷◇·≥
'o¶ç±d, ¥fł3°βH?_i~'√<Æzç§"ÀK*Ã«?<`öÖæN(HRXU-â~ác
ÜİIzÅ:¶idSÿì0πJFíi"÷∞:~"†V»',,, "ôdâuTΩ{<héäÄË);
o,U)è#æ·Jªfi†İÕí2ÜdHÃì2°@'jÀ=·Ú°@*qx]bèuSzÃµxKø
&PkY·VòVçİ|fÉIi~Ê»ùòûó2i>{jÛdÜß≤'ΩC;L·}n'I"·T≤

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,
Doom'd for a certain term to walk the night,
And for the day confin'd to fast in fires,
Till the foul crimes done in my days of nature
Are burnt and purged away. But that I am forbid
To tell the secrets of my prison-house,
I could a tale unfold whose lightest word
Would harrow up thy soul; freeze thy young blood,

```
maybeZip : List a -> List b -> Maybe (List (a,b))
maybeZip xs ys = case (xs, ys) of
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ o
                        Nothing -> Nothing
                        Just xys -> Just ((x,y) :: xys)
  ([], [])         -> Just []
  _                 -> Nothing
```



```
ñjò-|I[AÍŌŌnLE...@iq6Èûâ8€tù.À, :ŏÿc; :2πU" LZ™Yz±i  
ŌJ#|suurü={~'İðÀûŰ~«<>`bN™‡EvPèÆêŪN, Ω≥>n°fTπc"  
ª√≤ŏŌ...+Tã@$TwhJ<İŪ%(8"İŪFz5f:Ō<iI-^ð9*™s*~'İuj  
±[î!ùŌ$3áUà·Î·'Œŏ≥≥'ë\İİ\Ä"dfi<u¥v£1ÿZáŏ(cŏ÷ŏ·≥  
'oŏç±d, ¥fŏ3°βH?_i~'√<Œzç$"ÀK*Ã«?<`ŏŌæN(HRXU-â~á  
ŪİIzÅ:ŏidSYìŏπJFíi"÷∞:~"†V»',,, "ôdâuTΩ{<héäÄË);  
o,U)è#æ·Jªfi†İŌí2ŪdHÃì2°@'jÀ=·Ū°@*qX]bèuSzÃµxKø  
&PkY·VŌVçİ|fÉIi~Ê»ùŌúŏ2i>{jŪdŪβ≤'ΩC;L·}n'I"·T≤
```



Not a program

```
GHOST: So art thou to revenge, when thou shalt hear.  
  
HAMLET: What?  
  
GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,
```

```
maybeZip : List a -> List b -> Maybe (List (a,b))  
maybeZip xs ys = case (xs, ys) of  
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ o  
                        Nothing -> Nothing  
                        Just xys -> Just ((x,y) :: xys)  
  ([], [])          -> Just []  
  _                 -> Nothing
```

```
ñjò-|I[AÍŌŌnLE...@iq6Èûâ8€tù.À,:ŏÿc;:2πU"LZ™Yz±i  
ŌJ#|suurü={~'İðÀûŰ~«<>'bN™†EvPèÆêŪN,Ω≥>n°fTπc"  
ª√≤ŏŌ...+Tã@$TwhJ<İŪ%(8"İŪFz5f:Ō<iI-^ð9*™s*~'İuj  
±[î!ùŌ$3áUà·Î·'Œŏ≥≥'ë\İİ\Ä"dfi<u¥v£1ÿZáŏ(cŏ÷ŏ·≥  
'oŏç±d,¥fŏ3°βH?_i~'√<Œzç$"ÀK*Ã«?<`öŌæN(HRXU-â'á  
ŪİIzÅ:ŏidSYìŏπJFíi"÷∞:~"†V»',,,,"ôdâuTΩ{<héäÄË);  
o,U)è#æ·Jªfi†İŌí2ŪdHÃì2°@'jÀ=·Ū°@*qX]bèuSzÃµxKø  
&PKY·VŌVçİ|fÉIi~Ê»ùŌúŏ2i>{jŪdŪβ≤'ΩC;L·}n'I"·T≤
```

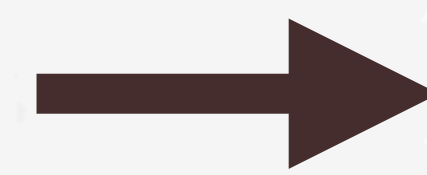


Not a program

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,
Doom'd for a certain term to walk the night,
And for the day confin'd to fast in fires,
Till the foul crimes done in my days of nature
Are burnt and purged away. But that I am forbid
To tell the secrets of my prison-house,
I could a tale unfold whose lightest word
Would harrow up thy soul; freeze thy young blood,



Not a program

```
maybeZip : List a -> List b -> Maybe (List (a,b))  
maybeZip xs ys = case (xs, ys) of  
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ o  
                        Nothing -> Nothing  
                        Just xys -> Just ((x,y) :: xys)  
  ([], [])          -> Just []  
  _                  -> Nothing
```

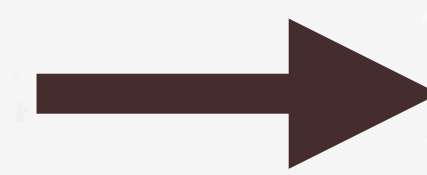


```
ñjò-|I[AÍŌŌnLE...@iq6Èûâ8€tù.À, :ŏÿc;:2πU"LZ™Yz±i  
ŌJ#|suurü={~'İðÀûŰ~«<>`bN™‡EvPèÆêŪN,Ω≥>n°fTπc"  
ª√≤ŏŌ...+Tã@$TwhJ<İŰ%(8"İŪFz5f:Ō<iI-^ð9*™s*~'İuj  
±[î!ùŌ$3áUà·Î·'Œŏ≥≥'ë\İİ\Ä"dfi<u¥vƒ1Ÿzáo(cŏ÷ŏ·≥  
'oŏç±d,¥fŏ3°βH?_i~'√<Œzç$"ÀK*Ã«?<`ŏŌæN(HRXU-â˘á  
ŪİIzÅ:ŏidSŸìŏπJFíi"÷∞:~"†V»',,,,"ôdâuTΩ{<héäÄË);  
o,U)è#æ·Jªfi†İŌí2ŪdHÃì2°@'jÀ=·Ū°@*qX]bèuSzÃµxKø  
&PKY·VŌVçİ|fÉIi~Ê»ùŌúŏ2i>{jŪdŪß≤'ΩC;L·}n'I"·T≤
```



Not a program

```
GHOST: So art thou to revenge, when thou shalt hear.  
  
HAMLET: What?  
  
GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,
```



Not a program

```
maybeZip : List a -> List b -> Maybe (List (a,b))  
maybeZip xs ys = case (xs, ys) of  
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ of  
    Nothing -> Nothing  
    Just xys -> Just ((x,y) :: xys)  
  ([], []) -> Just []  
  _ -> Nothing
```



**Not a program
(hidden syntax error)**

Problem for Beginners

Problem for Beginners

```
-- SYNTAX PROBLEM ----- ././Utils.elm
Arrows are reserved for cases and anonymous functions. Maybe you want > or >=
instead?

46|           Nothing -> Nothing
                        ^
```

Problem for Beginners

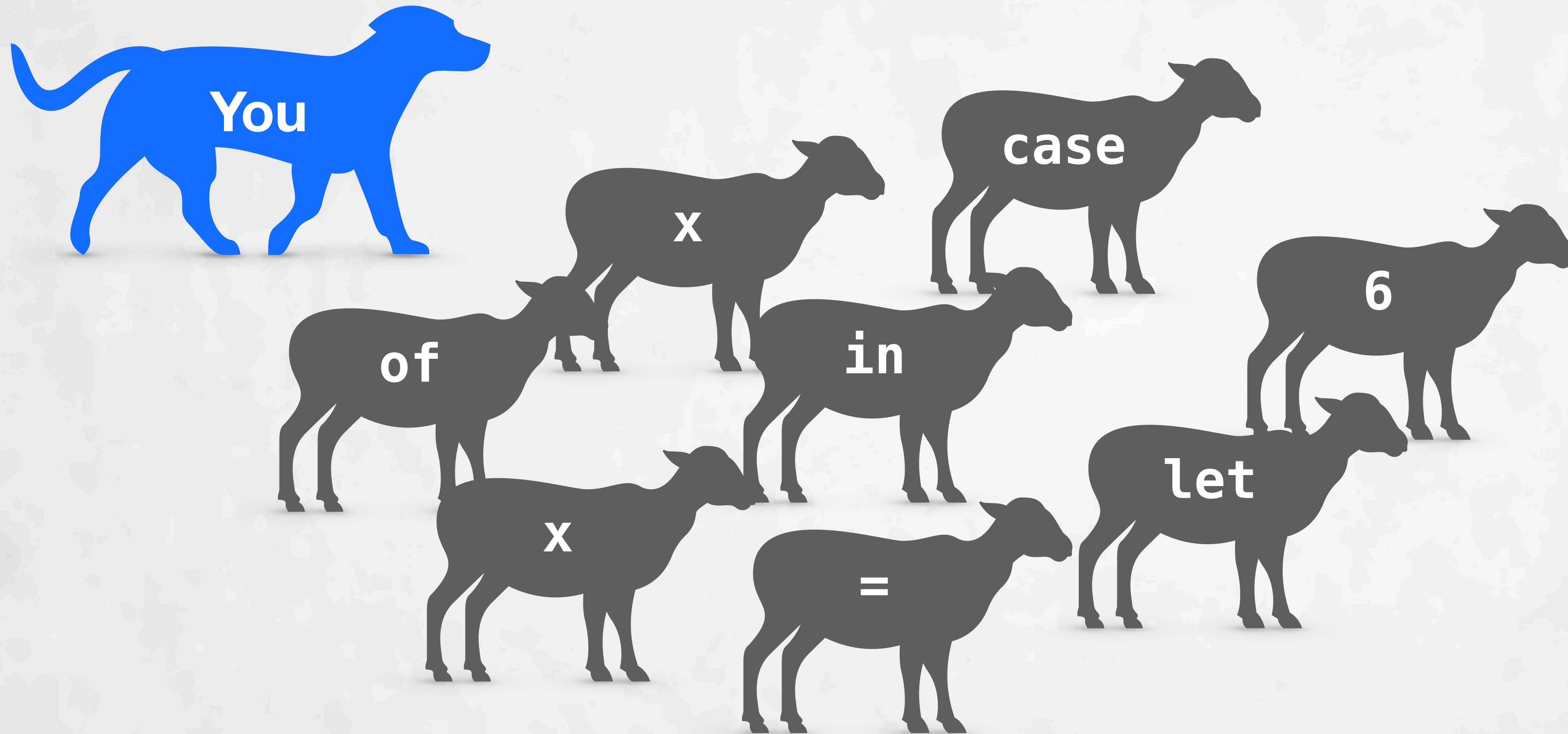
```
-- SYNTAX PROBLEM ----- ././Utils.elm
Arrows are reserved for cases and anonymous functions. Maybe you want > or >=
instead?

46|           Nothing -> Nothing
                        ^
```

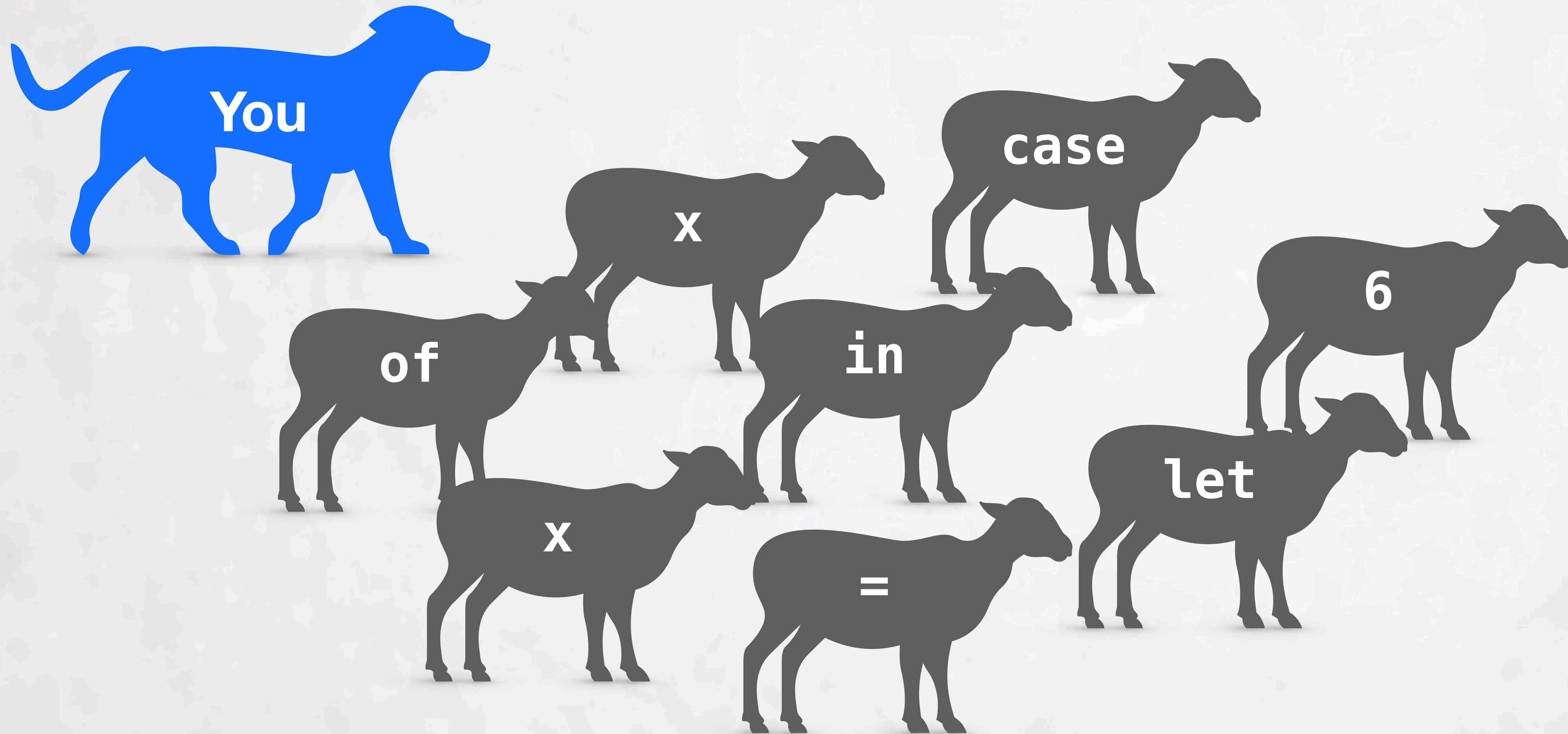
Structure must come from your head!

Problem for Experts

Problem for Experts



Problem for Experts



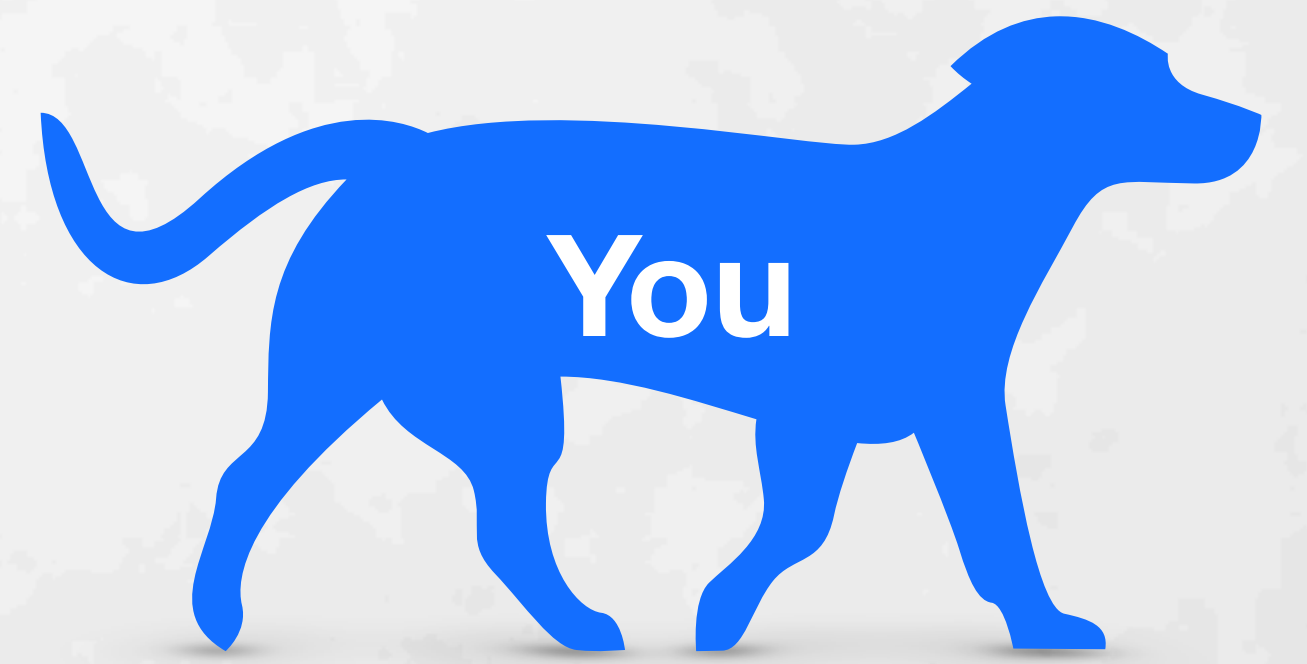
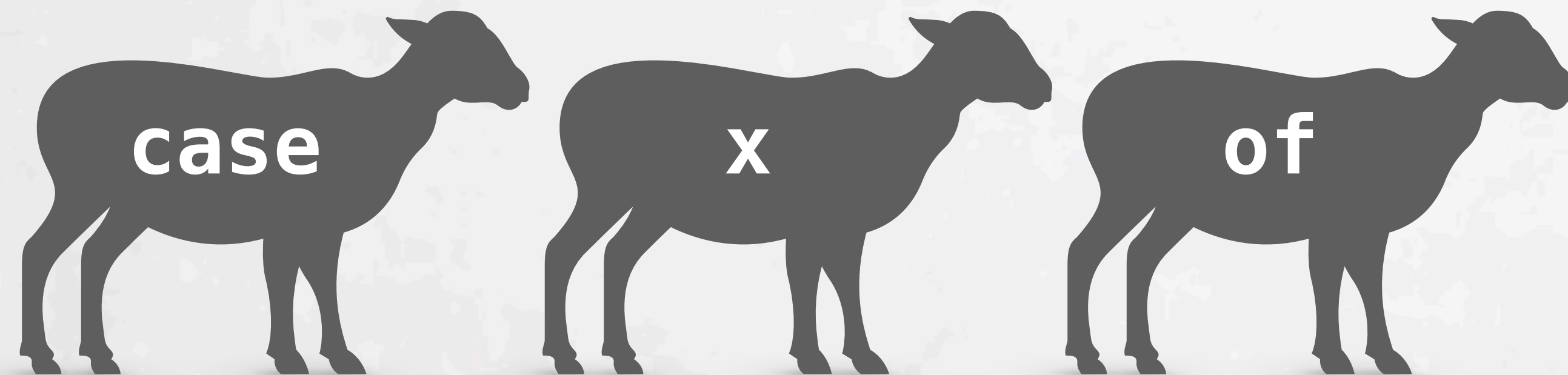
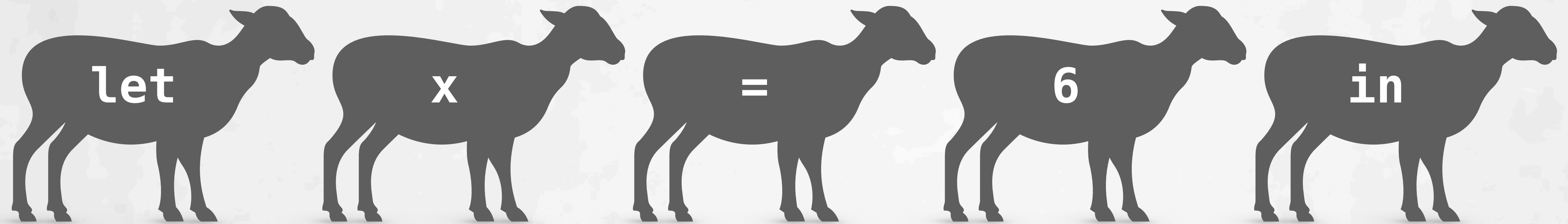
Spend time herding text!

Problem for Experts



Spend time herding text!

Problem for Experts



Spend time herding text!

Your program isn't text.

Your program isn't text.
It's an AST.

Text Changes

Text Changes

≠

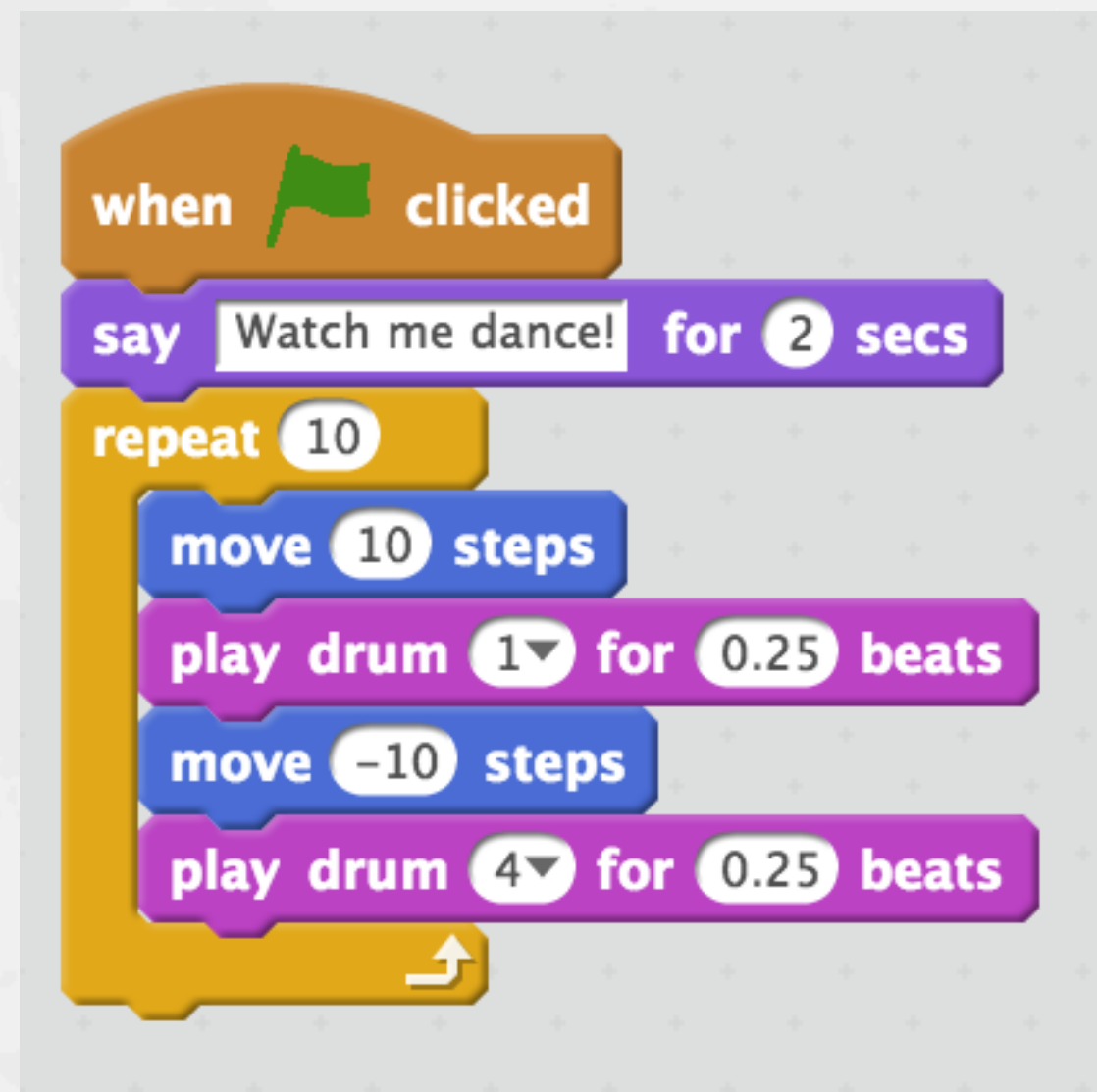
Text Changes

≠

AST Changes

Structured Editors

Structured Editors



Scratch

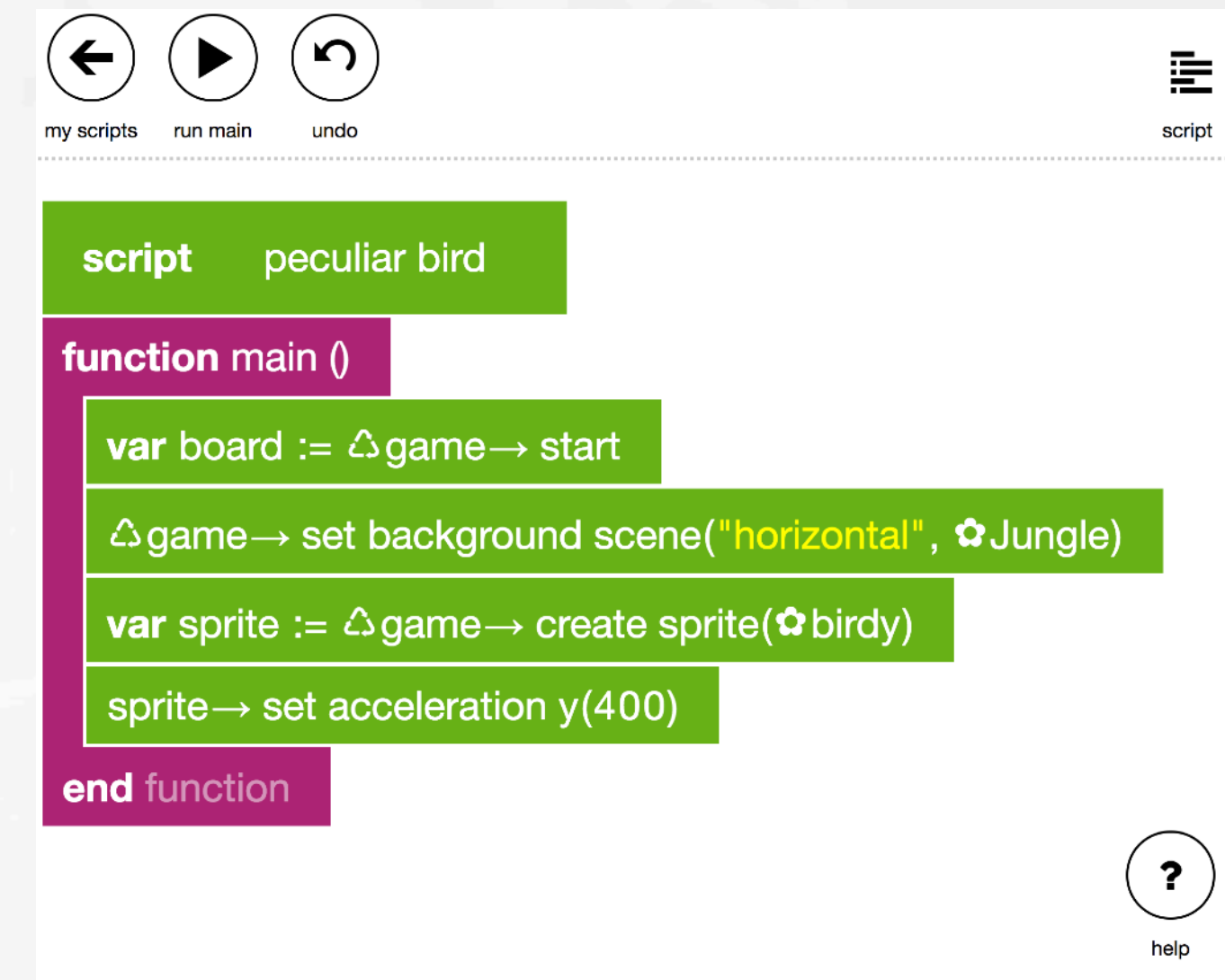
(Maloney et al. 2010; Resnick et al. 2009)

Structured Editors



Scratch

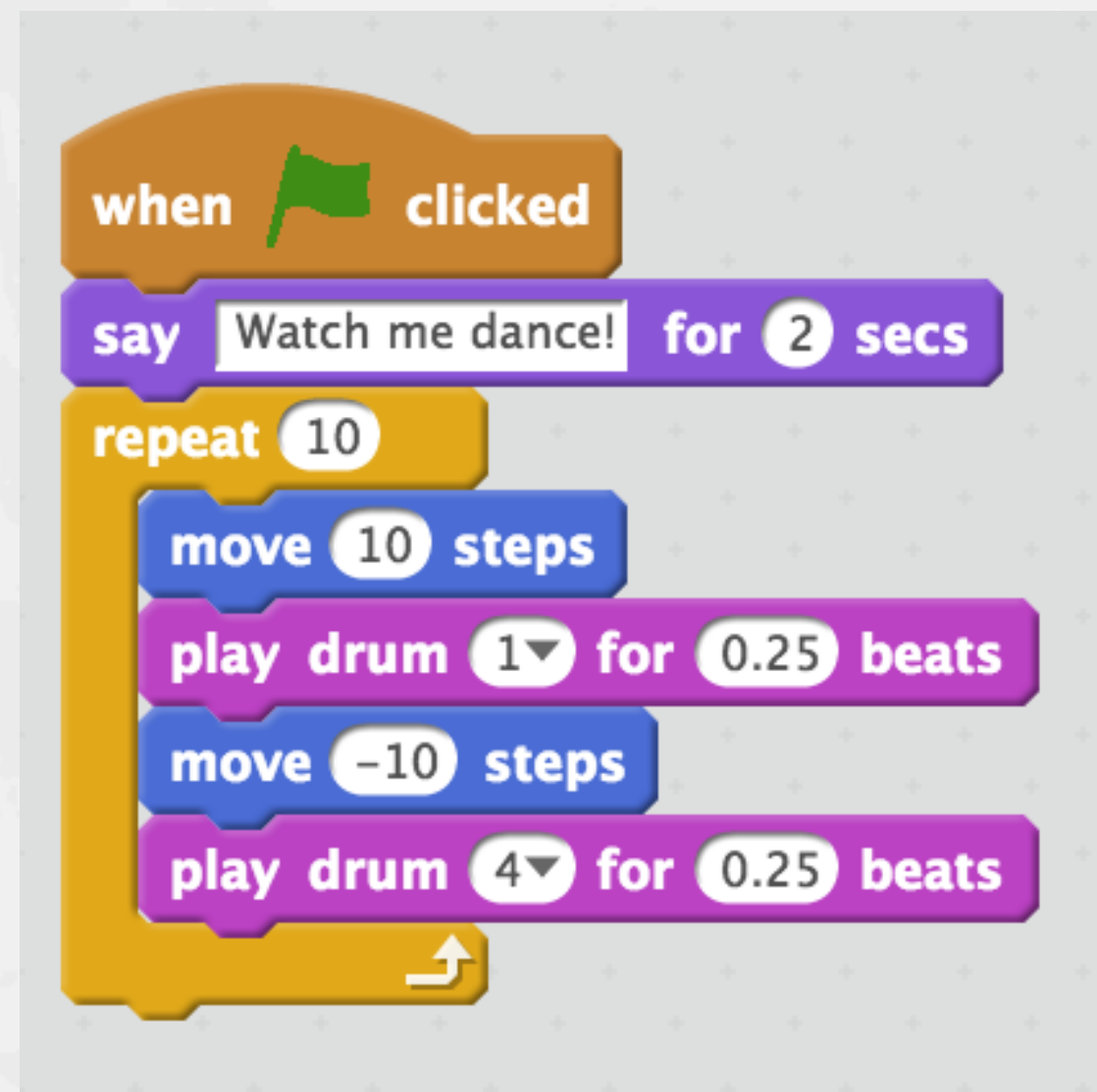
(Maloney et al. 2010; Resnick et al. 2009)



TouchDevelop

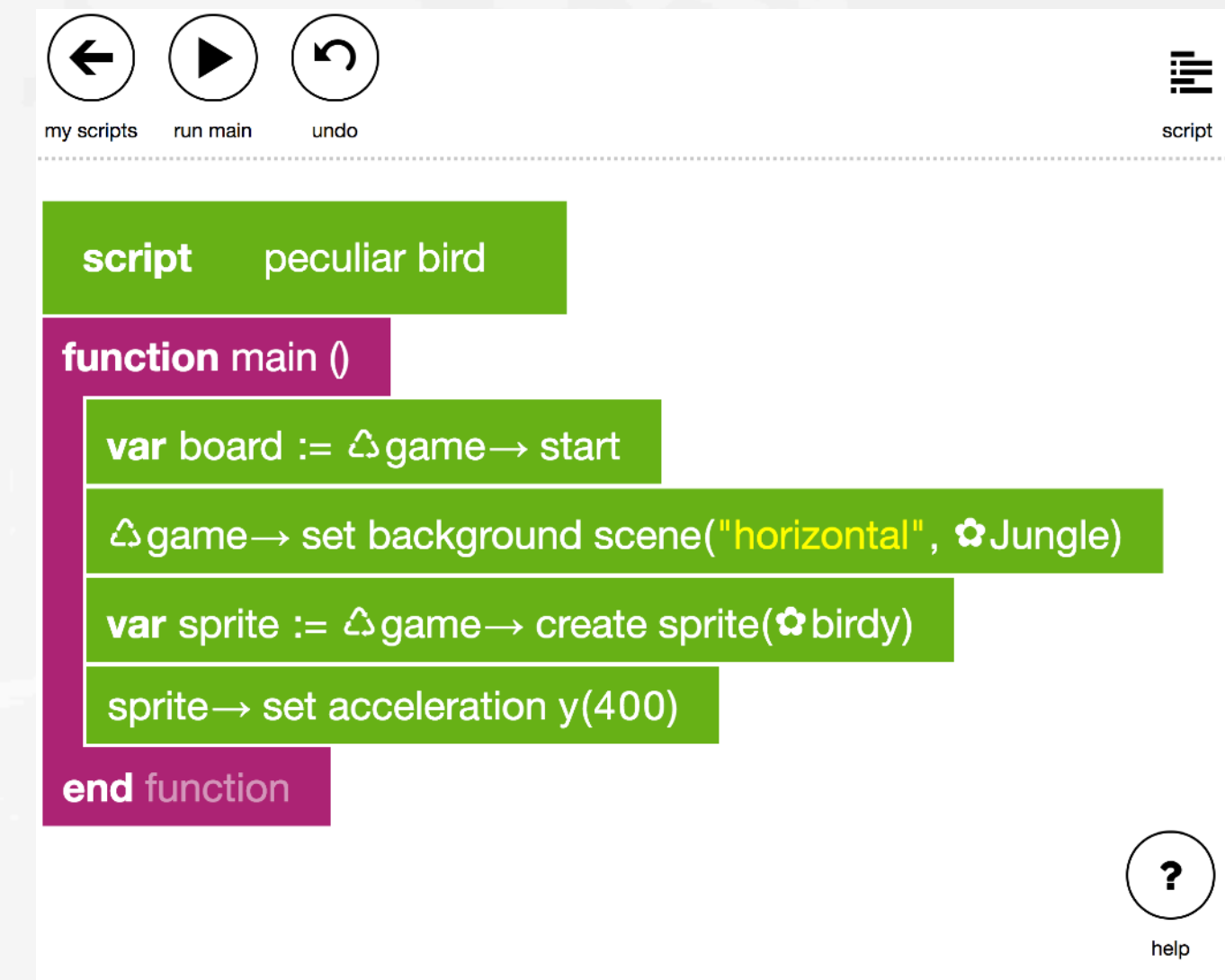
(Tillmann et al. 2012)

Structured Editors



Scratch

(Maloney et al. 2010; Resnick et al. 2009)



TouchDevelop

(Tillmann et al. 2012)

UI challenges; Experts still use plain text

Traditional Refactoring

Traditional Refactoring

Text-Select

Menu

Configure

Traditional Refactoring

Text-Select

Menu

Configure

```
@Override
public ConvexOptimizer getOptimizer() {
    if (optimizer == null) {
        Solver solver = new Solver.Builder().model(this).configure(conf()).build();
        this.optimizer = solver.getOptimizer();
    }
    return optimizer;
}

/**Returns the parameters of the neural network as a flattened row vector
 * @return the parameters of the neural network
 */
@Override
public INDArray params() {
    return paramsFlattened;
}

@Override
public INDArray getParam(String param) {
    return params.get(param);
}
```


Traditional Refactoring

Text-Select

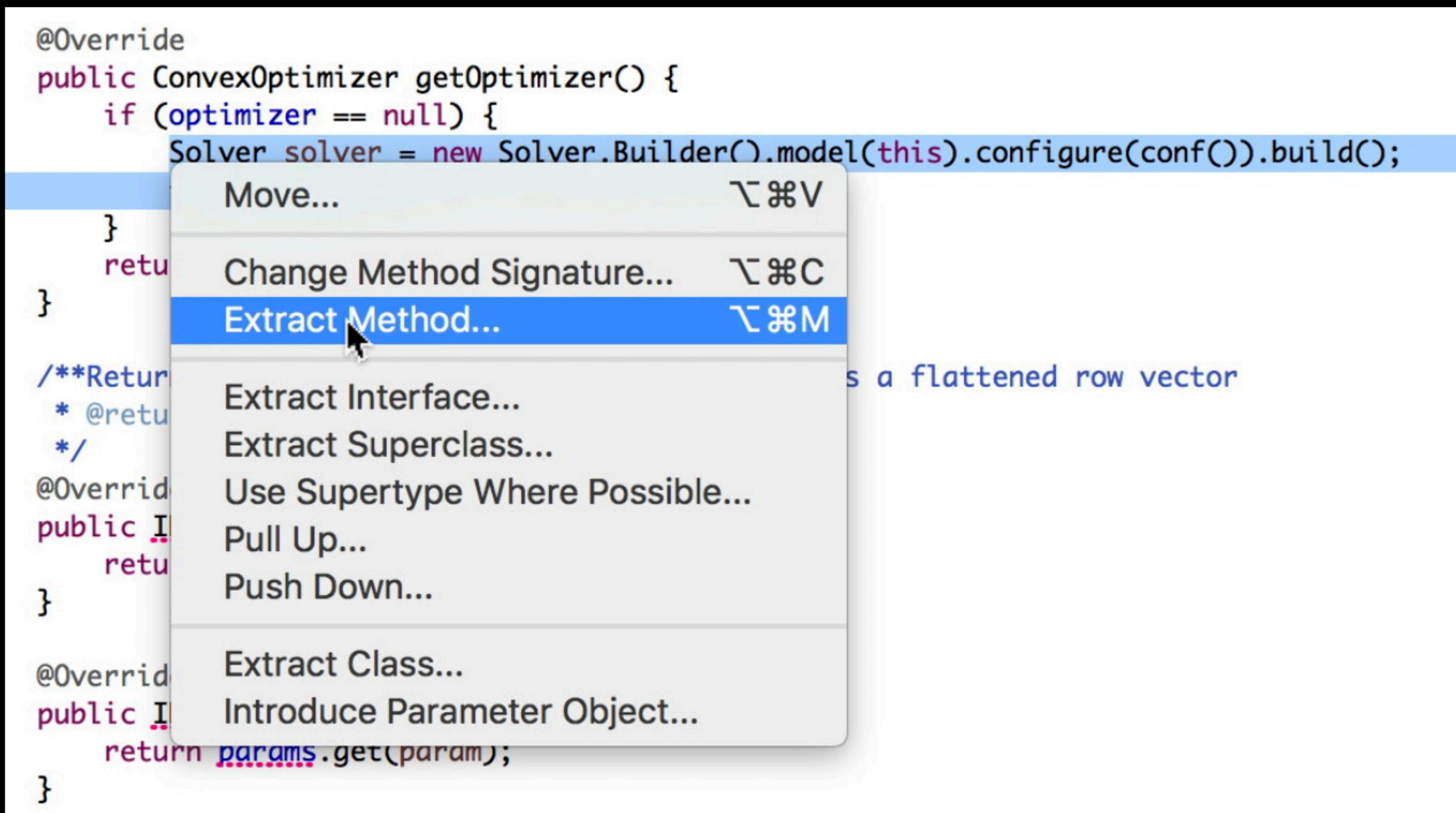
Menu

Configure

```
@Override
public ConvexOptimizer getOptimizer() {
    if (optimizer == null) {
        Solver solver = new Solver.Builder().model(this).configure(conf()).build();
    }
    return solver;
}

/**Return
 * @retu
 */
@Override
public I
return
}

@Override
public I
return params.get(param);
}
```



Move... ⌘⌘V

Change Method Signature... ⌘⌘C

Extract Method... ⌘⌘M

Extract Interface...

Extract Superclass...

Use Supertype Where Possible...

Pull Up...

Push Down...

Extract Class...

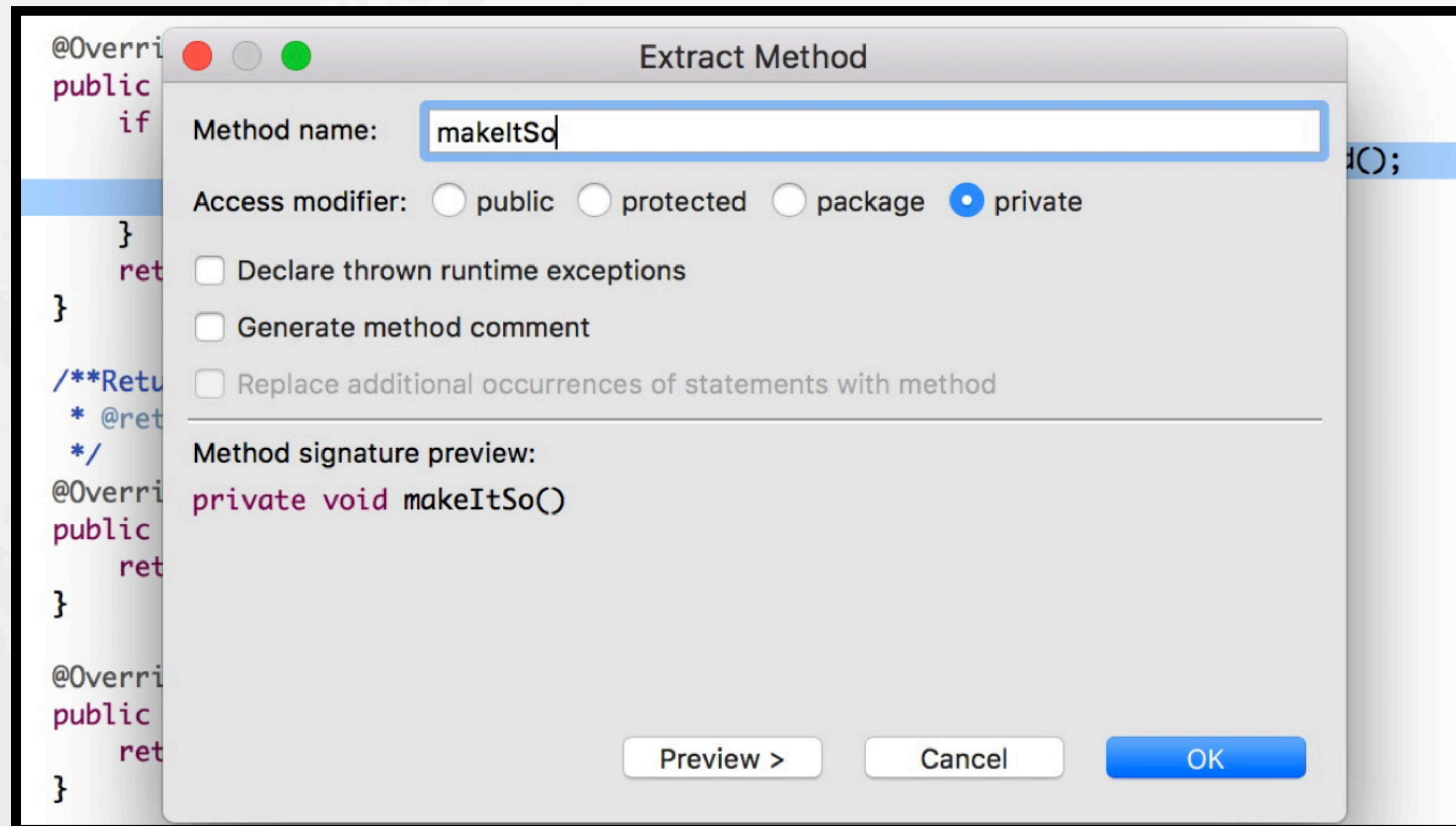
Introduce Parameter Object...

Traditional Refactoring

Text-Select

Menu

Configure



Traditional Refactoring

Text-Select

Menu

Configure

Traditional Refactoring

Text-Select

Menu

Configure

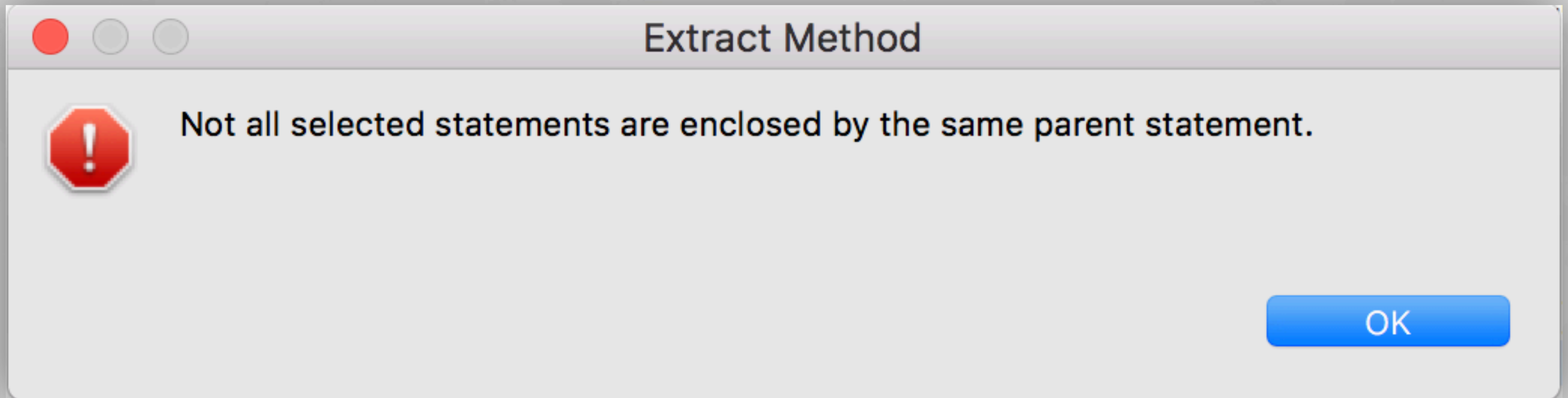
X Awkward

Traditional Refactoring

Text-Select

Menu

Configure



Traditional Refactoring

Text-Select

Menu

Configure

X Awkward

Traditional Refactoring

Text-Select

Menu

Configure

X Awkward

X Multiple
Selections

Traditional Refactoring

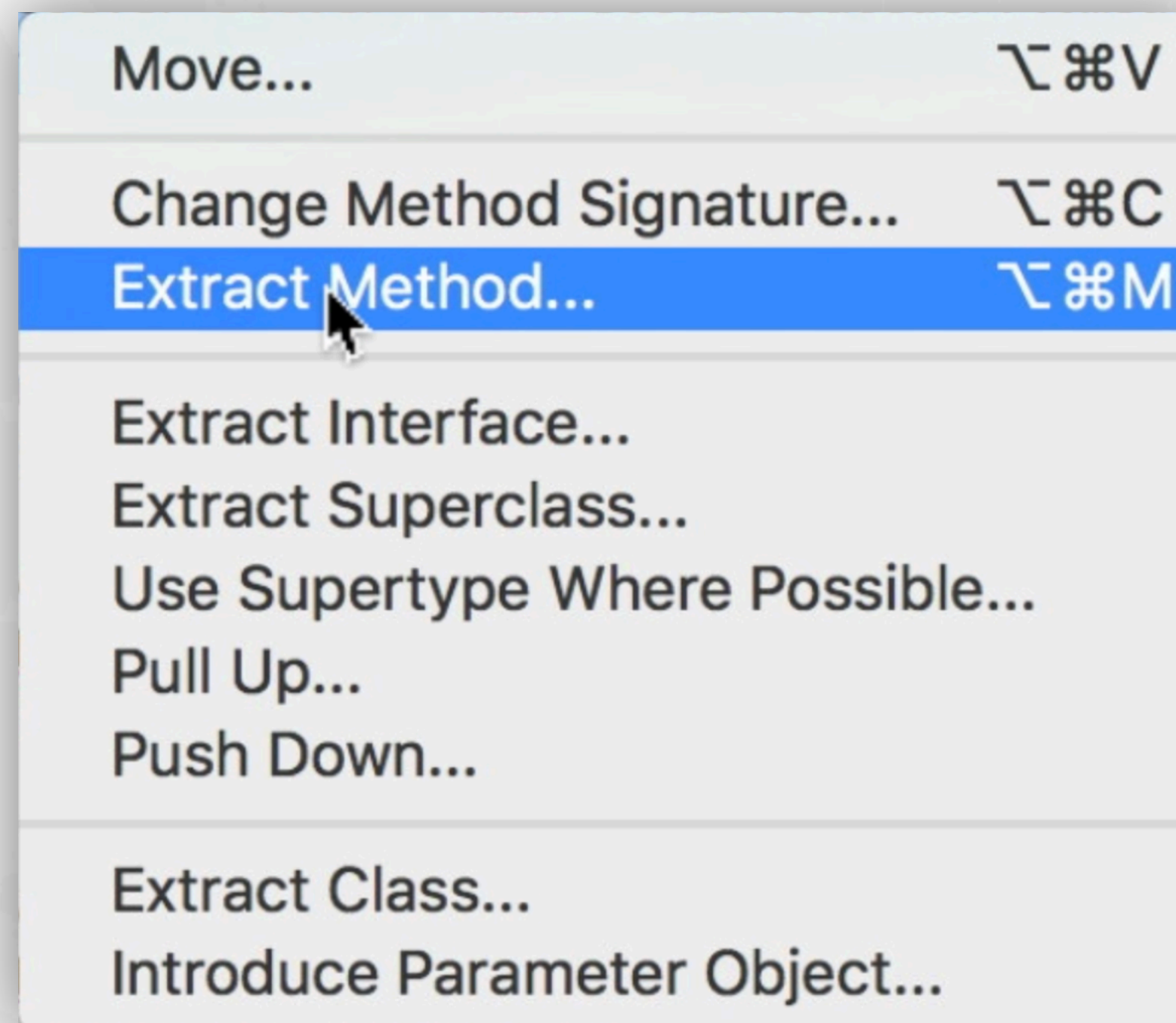
Text-Select

Menu

Configure

X Awkward

X Multiple Selections



Traditional Refactoring

Text-Select

Menu

Configure

X Awkward

X Many
Options

X Multiple
Selections

Text

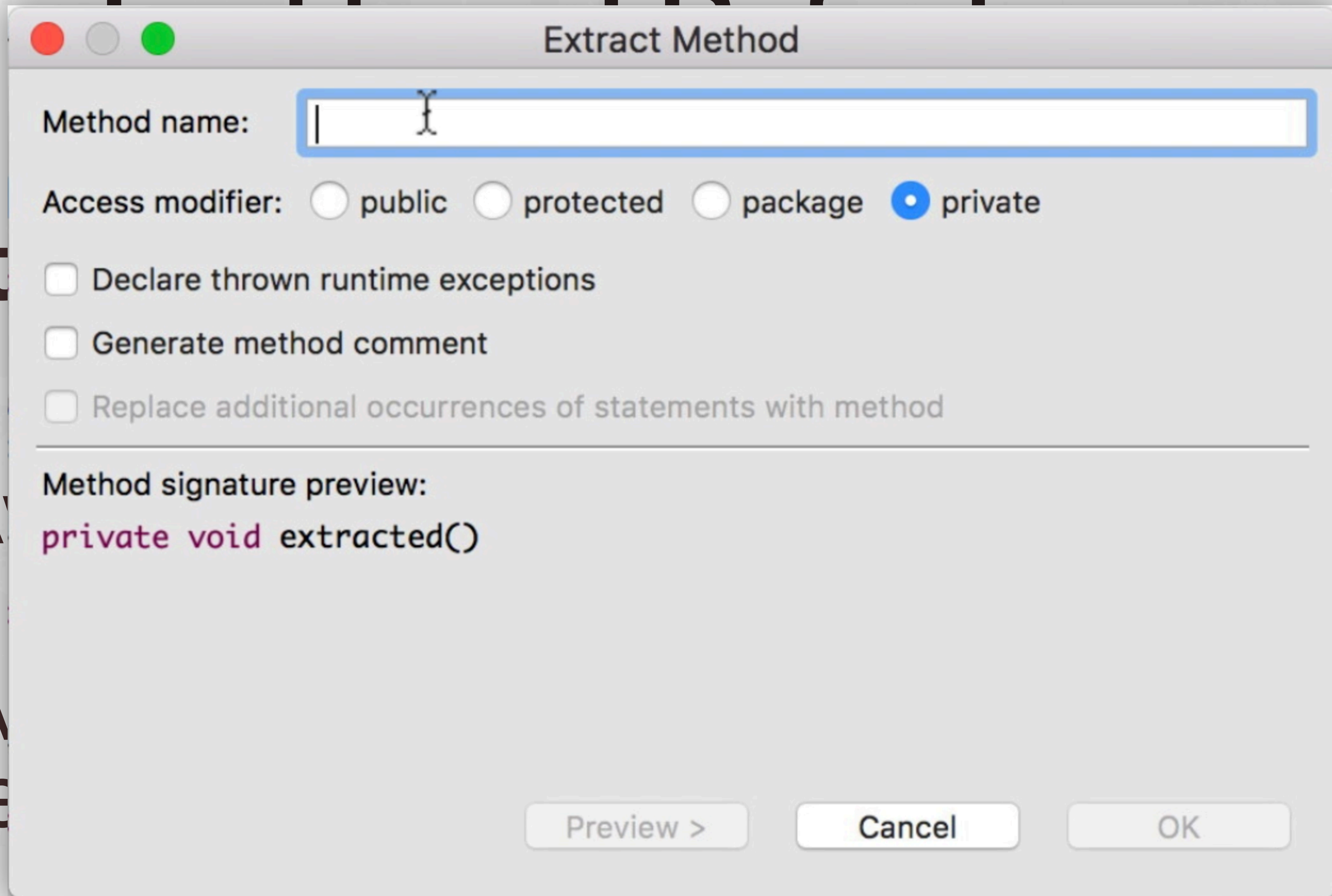
~~X~~

A

~~X~~

M

Se



Figure

Traditional Refactoring

Text-Select

Menu

Configure

X Awkward

**X Many
Options**

X Dialogs

**X Multiple
Selections**

Traditional Refactoring

Text-Select

Menu

Configure

 Awkward

 Many
Options

 Dialogs

 Multiple
Selections

Deuce

Text-Select

Menu

Configure

 Awkward

 Many
Options

 Dialogs

 Multiple
Selections

Deuce

Structure
Select

Menu

Configure

```
(def image1
  (let [width height]
    (let [x y] [50 65]
      (image "lightgrey"))))

(def main
  (draw (concat [image
```

 Many
Options

 Dialogs

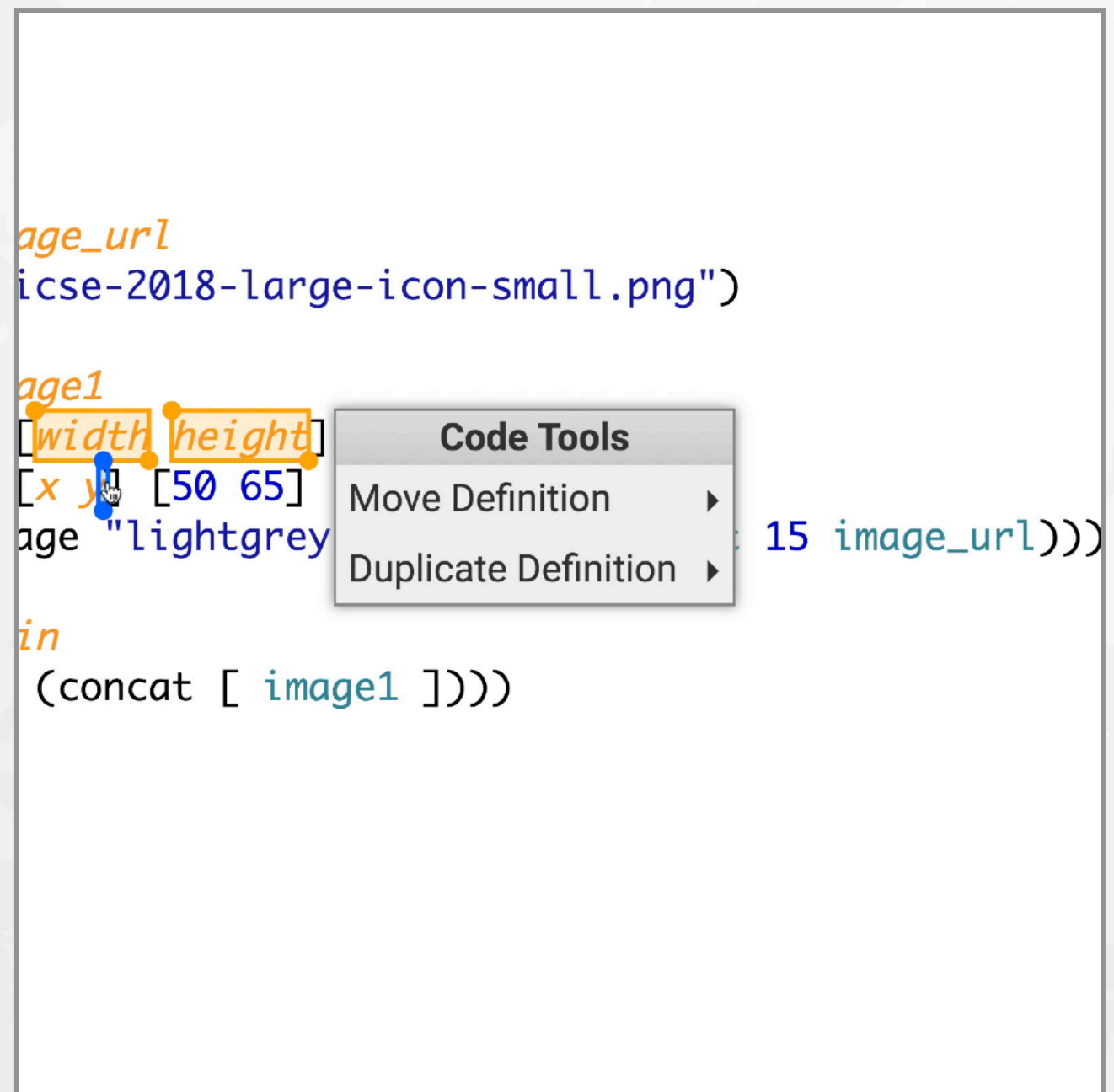
Deuce

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [50 65]  
      (image "lightgrey"  
             15 image_url)))  
)  
  
(def main  
  (draw (concat [ image1 ])))
```

Short Menu

```
image_url  
"icse-2018-large-icon-small.png")  
  
image1  
[width height]  
[x y] [50 65]  
image "lightgrey"  
15 image_url)))  
  
in  
(concat [ image1 ])))
```



Configure

~~Dialogs~~

Deuce

Structure Select

```
(def image1
  (let [width height]
    (let [x y] [50 65]
      (image "lightgrey"
             15 image_url)))

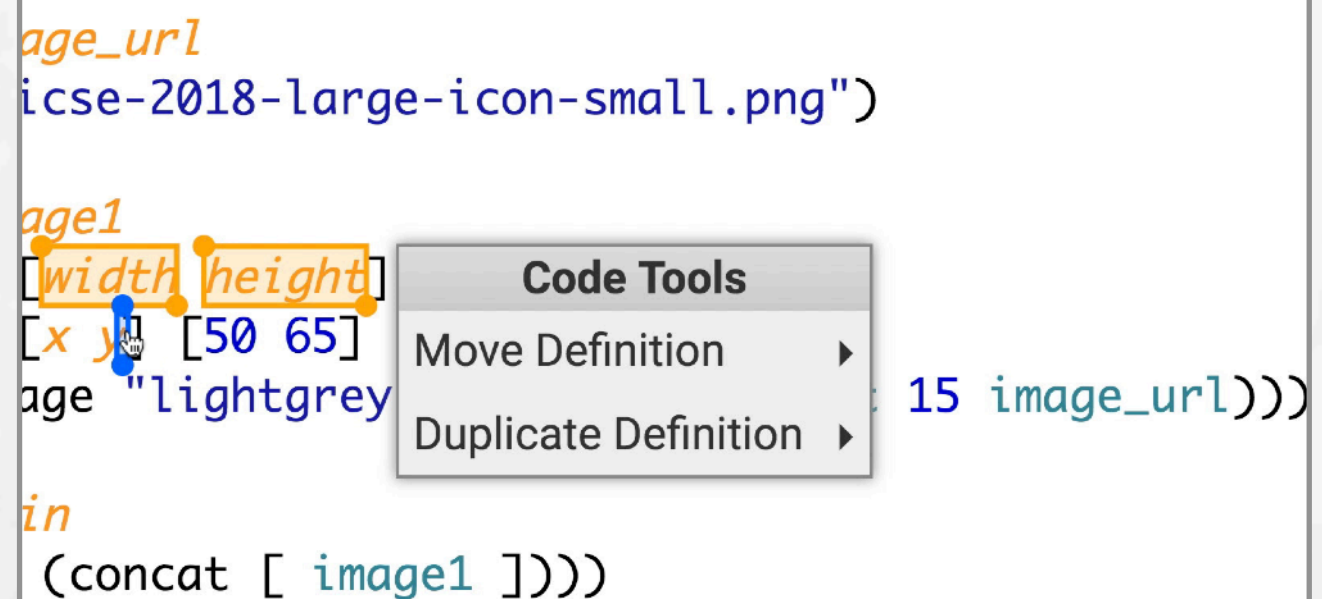
(def main
  (draw (concat [image1 ])))
```

Short Menu

```
image_url
"icse-2018-large-icon-small.png")

image1
[width height]
[x y] [50 65]
image "lightgrey"
15 image_url)))

in
(concat [ image1 ])))
```



Defaults

- ▶ Abstract image1 over its constants
- ▶ Abstract image1 over its named constants

Demo

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, fountain pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a 'Raw Stretchy Sticky' button.

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 54 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool. Below these tools are labels: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, Sticky.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool. Below these tools are the labels 'Raw', 'Stretchy', and 'Sticky'.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, pen tool, and a red star tool.

Raw
Stretchy
Sticky



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool.

Raw
Stretchy
Sticky



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶

```

1 (def image_url
2   "img/icse-2018-large-icon-
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))

```



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Square tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶

Introduce Variable



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square tool icon
- Black circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky tool icon

Current file: *Untitled **

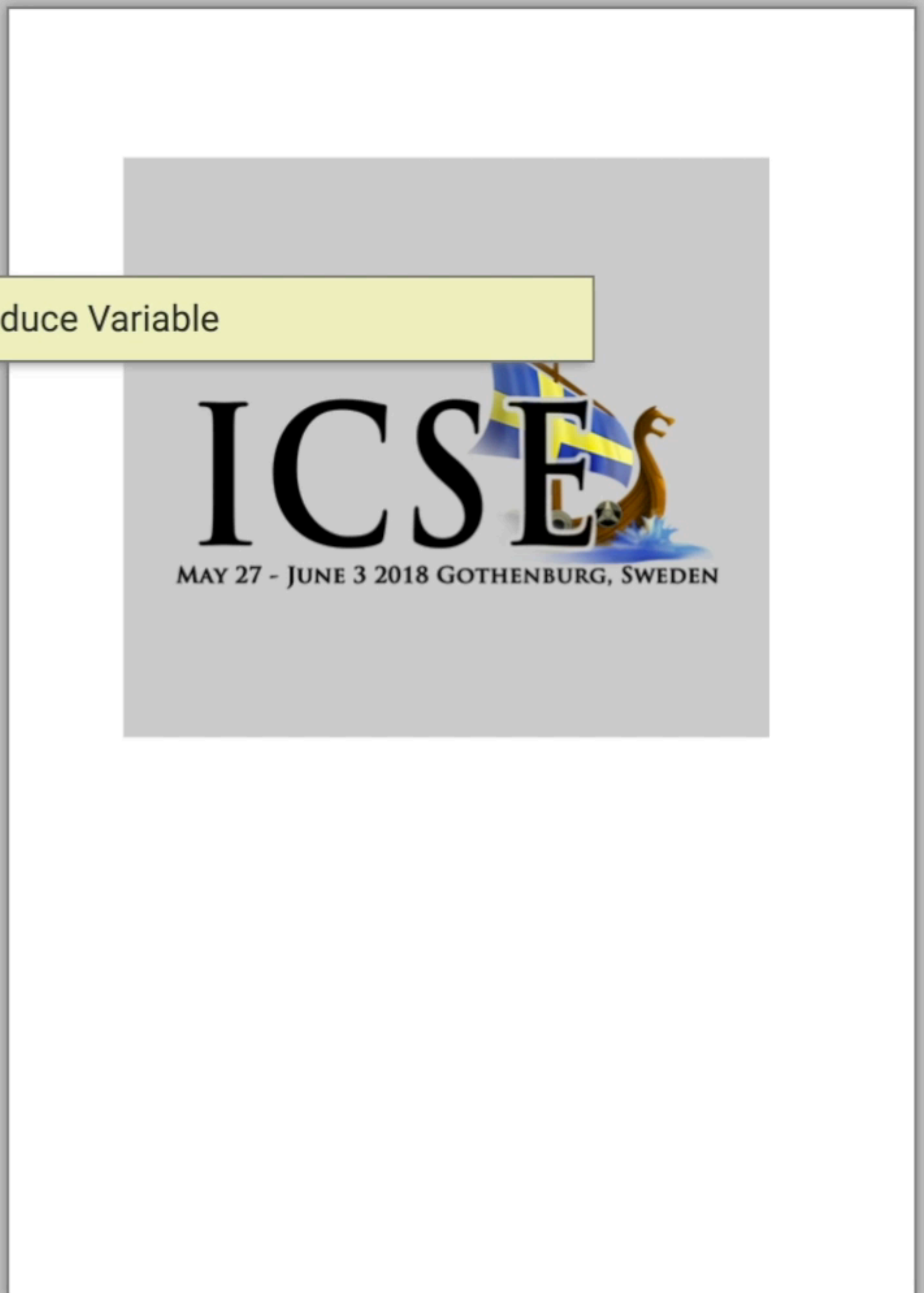
Undo Redo Clean Up Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶

Introduce Variable



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square icon
- Black circle icon
- Black star icon
- Pen tool icon
- Red star icon
- Raw Stretchy Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool.

Raw
Stretchy
Sticky



Current file: *Untitled **

Undo Redo Clean Up Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small."
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶



Navigation icons: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, red star.

Raw Stretchy Sticky



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```

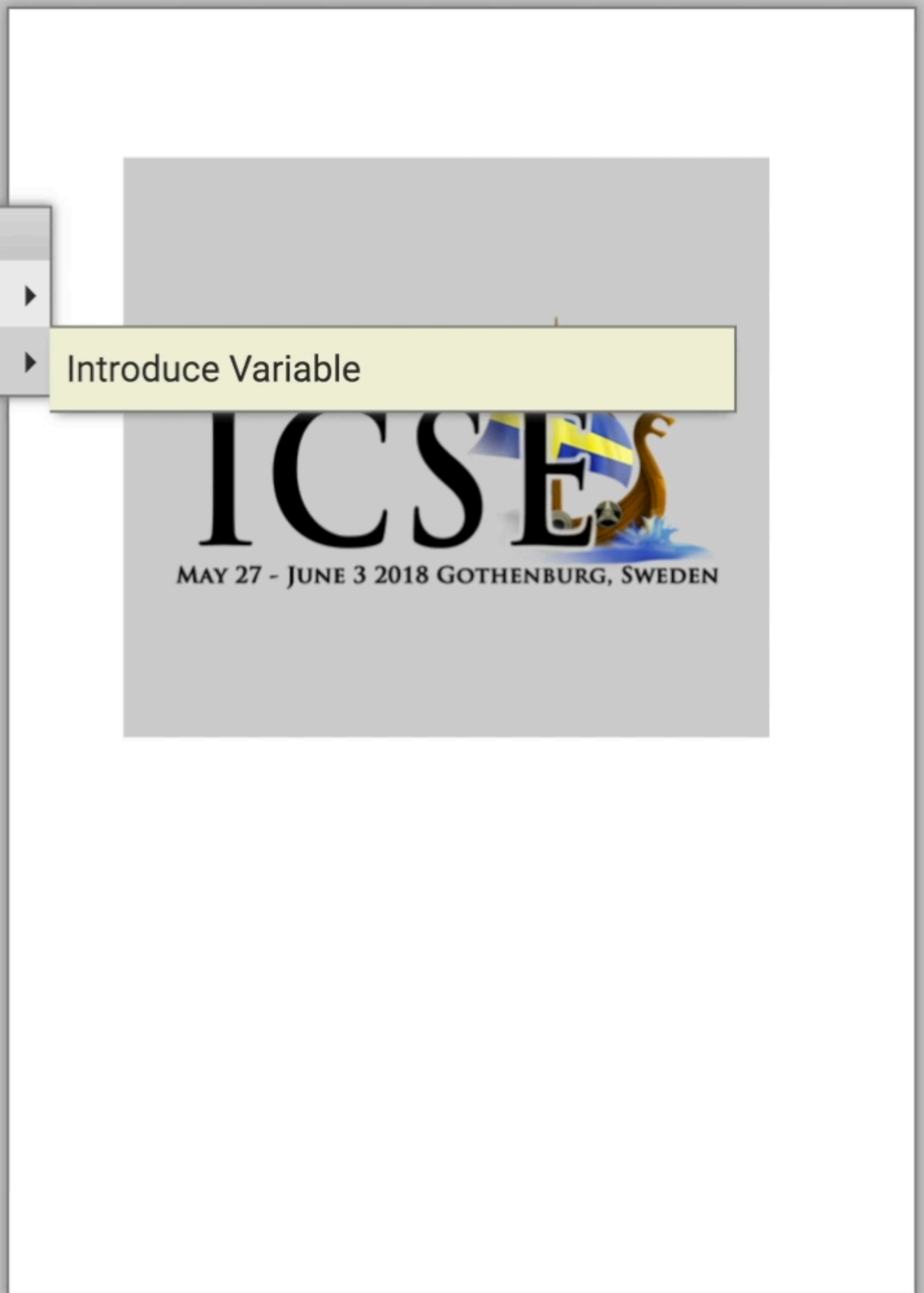
1 (def image_url
2   "img/icse-2018-large-icon-small."
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))

```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶

Introduce Variable



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Rectangle tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky

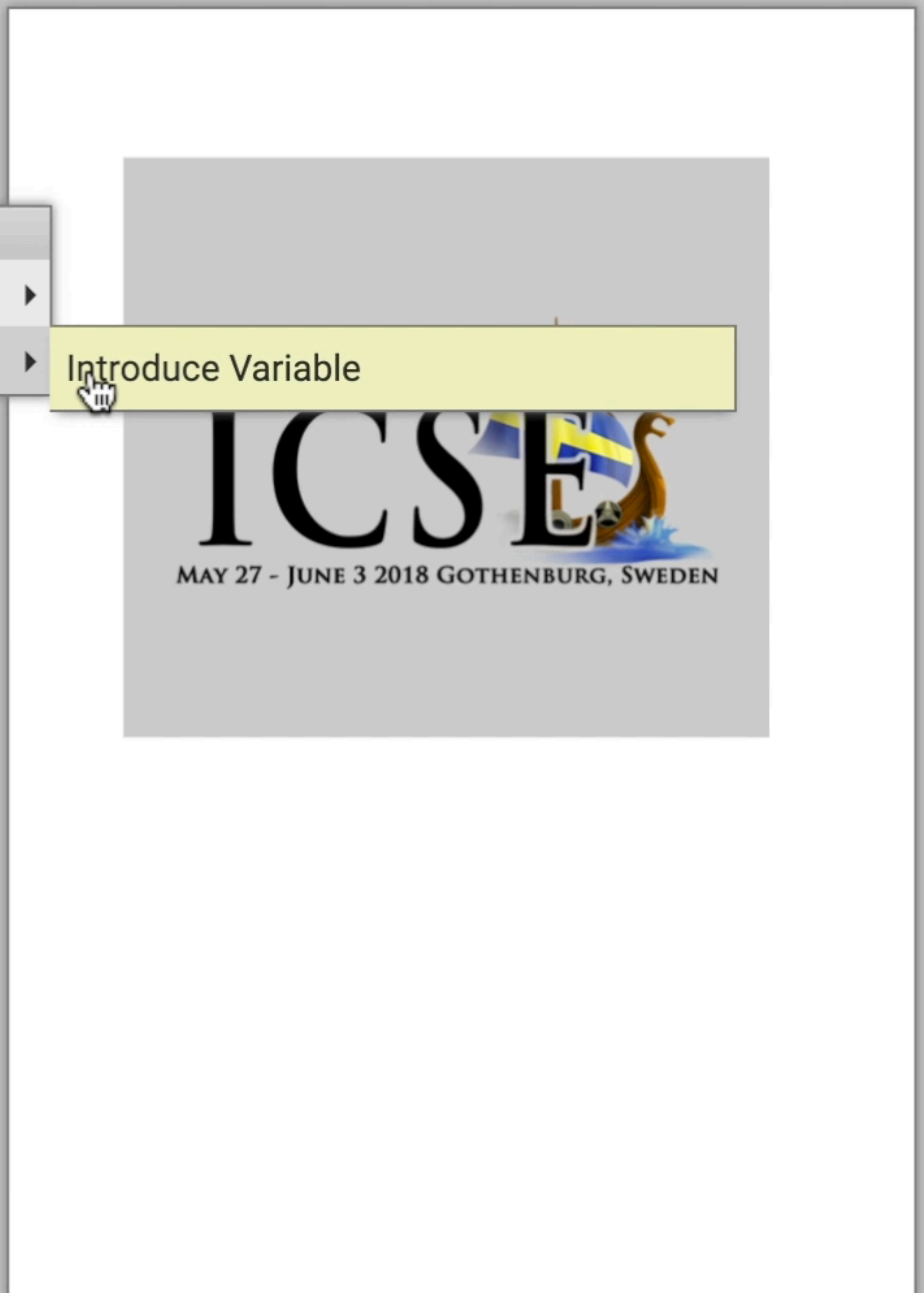
Current file: *Untitled **

Undo Redo Clean Up Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶ **Introduce Variable**



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square tool icon
- Black circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle, star), and drawing tools (pen, eraser). A dropdown menu at the bottom shows 'Raw', 'Stretchy', and 'Sticky' options.



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```

1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))

```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable ▶



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Square tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options



Current file: *Untitled **

Undo Redo Clean Up

Code Tools

- Create Function from Arguments ▶
- Introduce Variables ▶
- Make Equal with Single Variable ▶
- Make Equal by Copying ▶
- Swap Expressions ▶

```

1 (def image_url
2   "img/icse-2018-large-
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))

```

Run ▶



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Square tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options



Current file: *Untitled **

Undo Redo Clean Up

```
1 (def image_url
2   "img/icse-2018-large-
3
4 (def image1
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variables ▶
- Make Equal with Single Variable ▶
- Make Equal by Copying ▶
- Swap Expressions ▶

Run ▶

Introduce Variables



Navigation and drawing tools: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a star icon. Below these is a menu with options: Raw, Stretchy, Sticky.

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Raw
Stretchy
Sticky

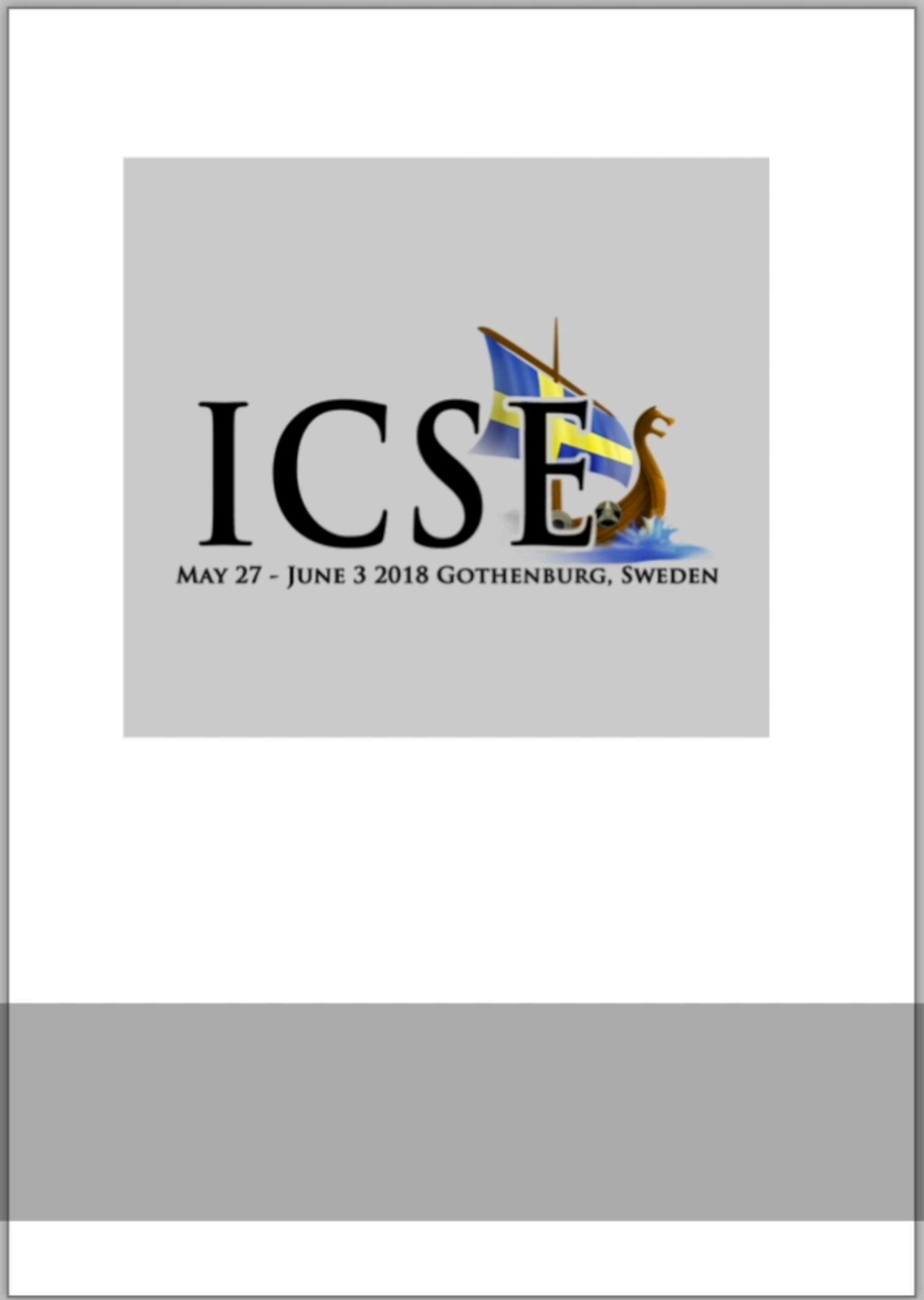
Current file: *Untitled **

Undo Redo Clean Up Run ▶

```
1 (def image1 (image "img/icon-small.png"))
2
3
4 (def image2
5   (let [width 283
6         height 254
7         [x y] [50 65]]
8     (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Code Tools

- Move Definition ▶
- Duplicate Definition ▶



Navigation and tool icons:

- Mouse cursor
- Text tool (T)
- Line tool
- Black square
- Black circle
- Star
- Pen tool
- Red star
- Raw Stretchy Sticky



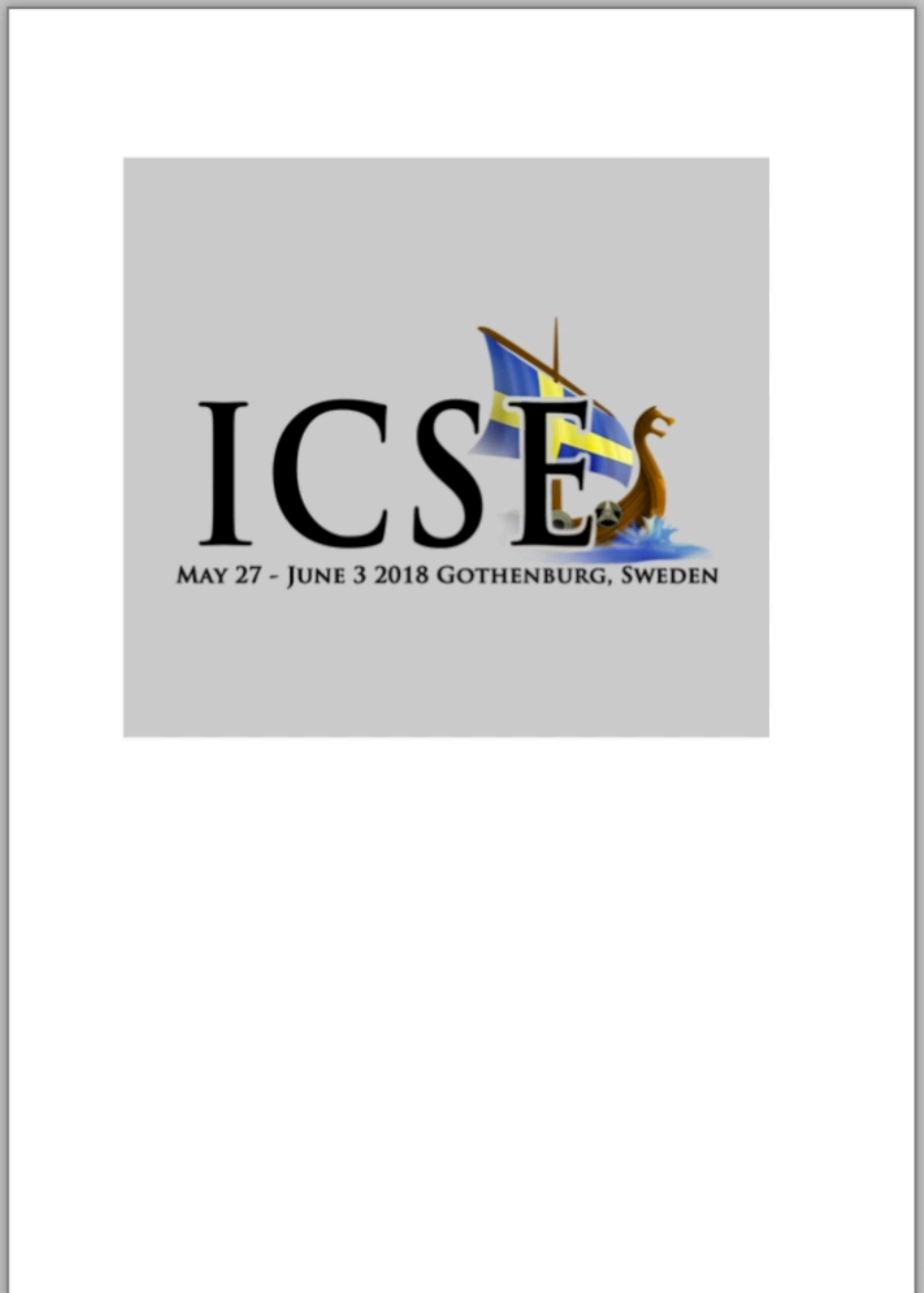
Current file: Untitled

```
1 (def image-url "img/icse.png")
2
3
4 (def image
5   (let [width 283
6         height 254
7         [x y] [[50 65]
8                (image "lightgrey" x y width height 15 image_url)]))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Definition ▶
- Create Function from Arguments ▶
- Rename *width* ▶
- Inline Definition ▶

Run ▶



Navigation and tool icons:

- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square icon
- Black circle icon
- Black star icon
- Pen tool icon
- Red star icon
- Raw Stretchy Sticky button

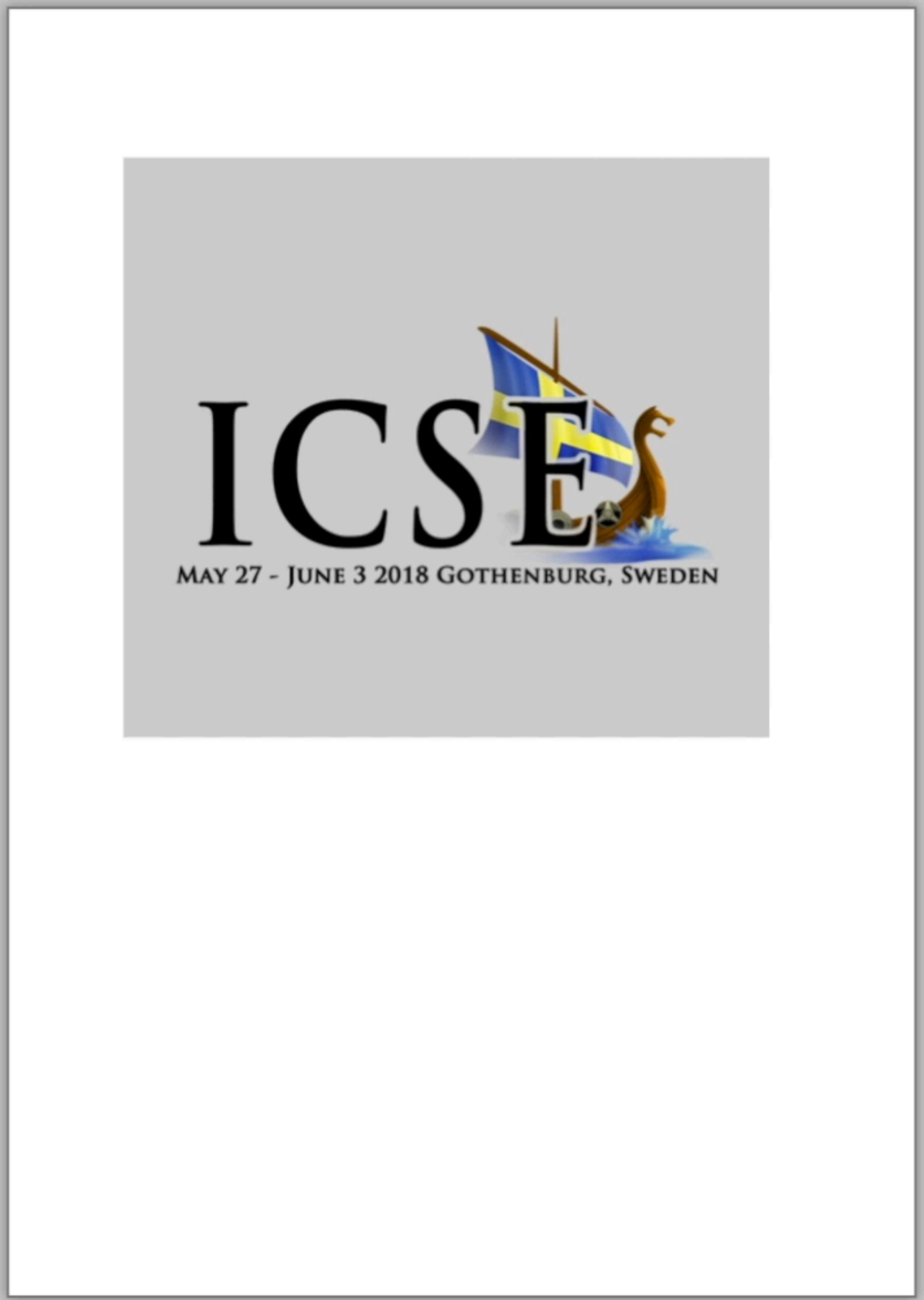
Current file: *Untitled **

Undo Redo Clean Up Run ▶

```
1 (def image-url "img/icon-small.png")
2
3
4 (def image
5   (let [width 283
6         height 254
7         [x y] [50 65]]
8     (image "lightgrey" x y width height 15 image-url)))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Code Tools

- Move Definition ▶
- Duplicate Definition ▶



Navigation and tool icons:

- Mouse cursor
- Text tool (T)
- Line tool
- Black square
- Black circle
- Black star
- Pen tool
- Red star
- Raw
- Stretchy
- Sticky

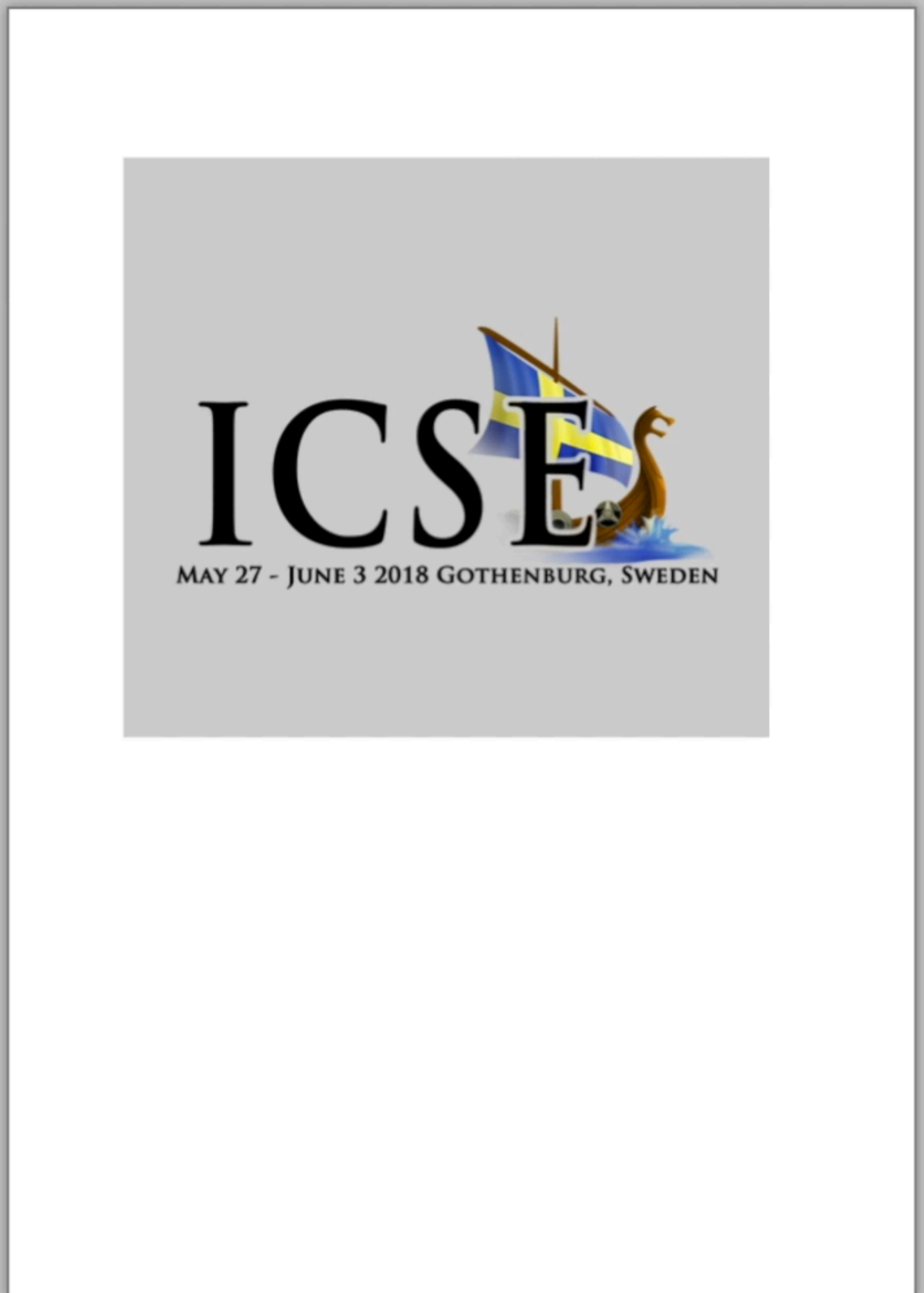
Current file: *Untitled **

Undo Redo Clean Up Run ▶

```
1 (def imag
2   "img/ic
3
4 (def imag
5   (let [width height] [283 254]
6     (let [x y] [50 65]
7       (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Code Tools

- Move Definition ▶ Move width
- Duplicate Definition ▶



Navigation icons: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, red star.

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let [width height] [283 254]
6     (let [x y] [50 65]
7       (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, fill tools (square, circle), star tool, fountain pen tool, and a red star tool. Below these is a menu with options: Raw, Stretchy, and Sticky.

Current file: *Untitled **

Undo Redo Clean Up Run ▶

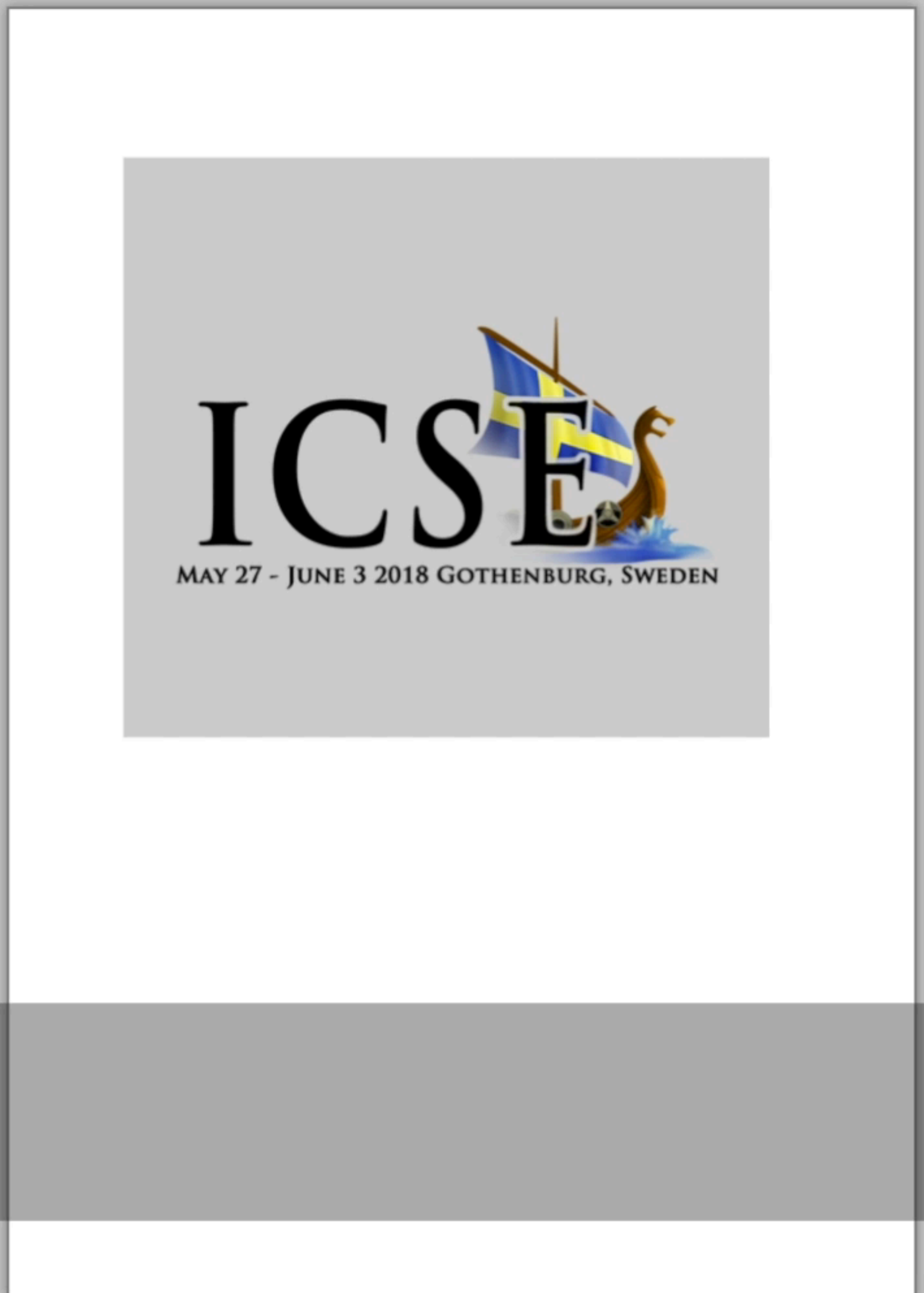
Code Tools

- Move Definition ▶
- Duplicate Definition ▶

```

1 (def image1
2   "img/ics-se-small.png")
3
4 (def image2
5   (let [width height] [283 254]
6     (let [x y] [50 65]
7       (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))

```



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Square tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options



Current file: *Untitled **

Undo Redo Clean Up Run ▶

Code Tools
Move Definition ▶ Move width and height
Duplicate Definition ▶

```
1 (def image_1  
2   "img/icse"   
3  
4 (def image_2  
5   (let [x y width height] [50 65 283 254]  
6     (image "lightgrey" x y width height 15 image_url)))  
7  
8 (def main  
9   (draw (concat [ image1 ])))
```



Navigation icons: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, red star tool.

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let [x y width height] [50 65 283 254]
6     (image "lightgrey" x y width height 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool. Below these tools are three checkboxes labeled 'Raw', 'Stretchy', and 'Sticky'.

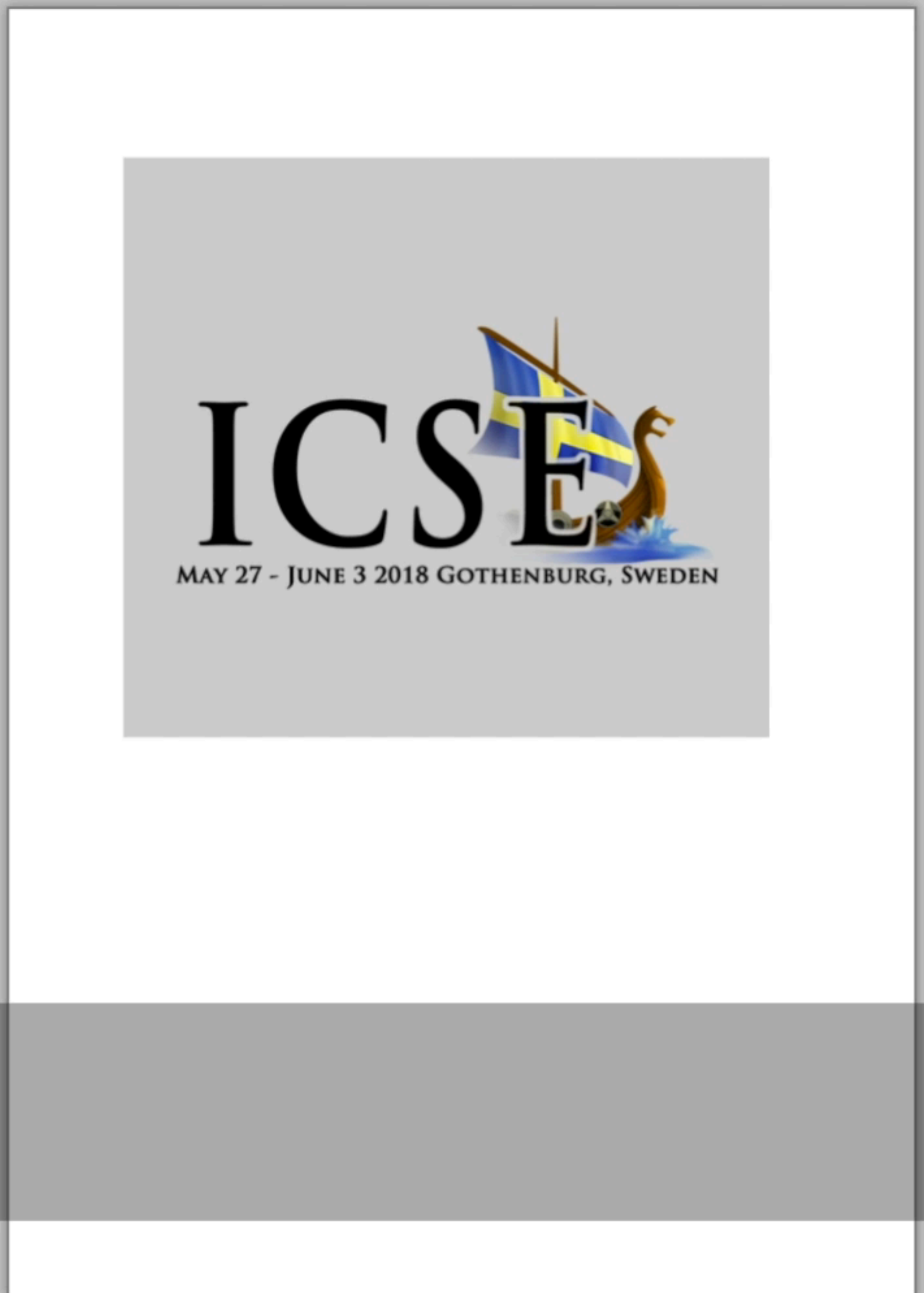
Current file: Untitled *

```
1 (concat [image "small.png"])
2
3
4 (def image1
5   (let [x y width height] [50 65 283 254]
6     (image "lightgrey" x y width height 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Definition ▶
- Inline Definition ▶
- Make Single Line ▶

Run ▶



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square icon
- Black circle icon
- Star icon
- Fountain pen icon
- Red star icon
- Raw Stretchy Sticky

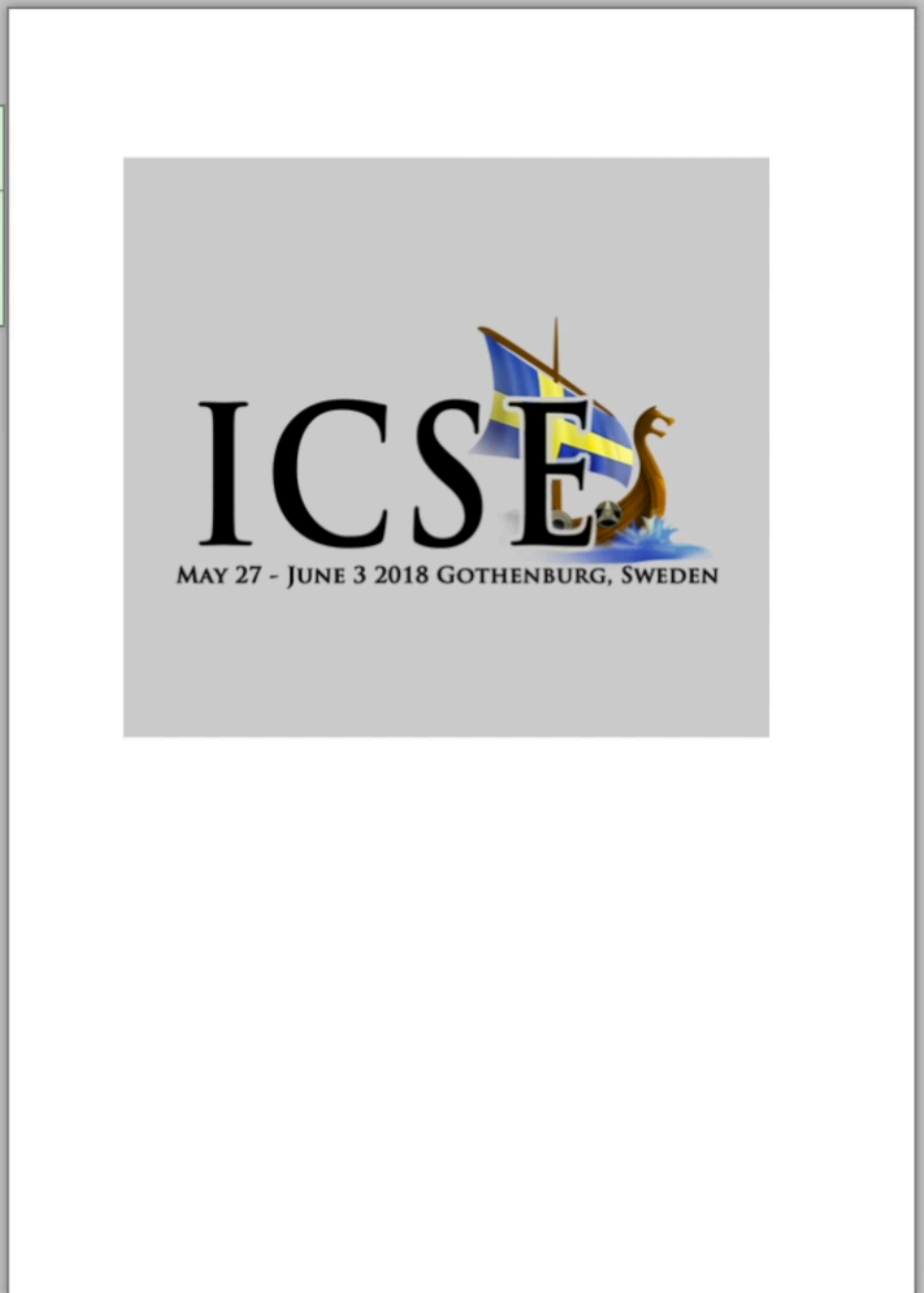


Current file: Untitled *

Code Tools

- Undo
- Create Function from Definition ▶ Abstract image1 over its constants
- Inline Definition ▶ Abstract image1 over its named
- Make Single Line ▶ constants

```
1 (def image1  
2   (let [x y width height] [50 65 283 254]  
3     (image "lightgrey" x y width height 15 image_url)))  
4  
5 (def main  
6   (draw (concat [ image1 ])))  
7  
8  
9
```



Navigation and tool icons: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, and a red star icon.

Raw
Stretchy
Sticky

Current file: Untitled *

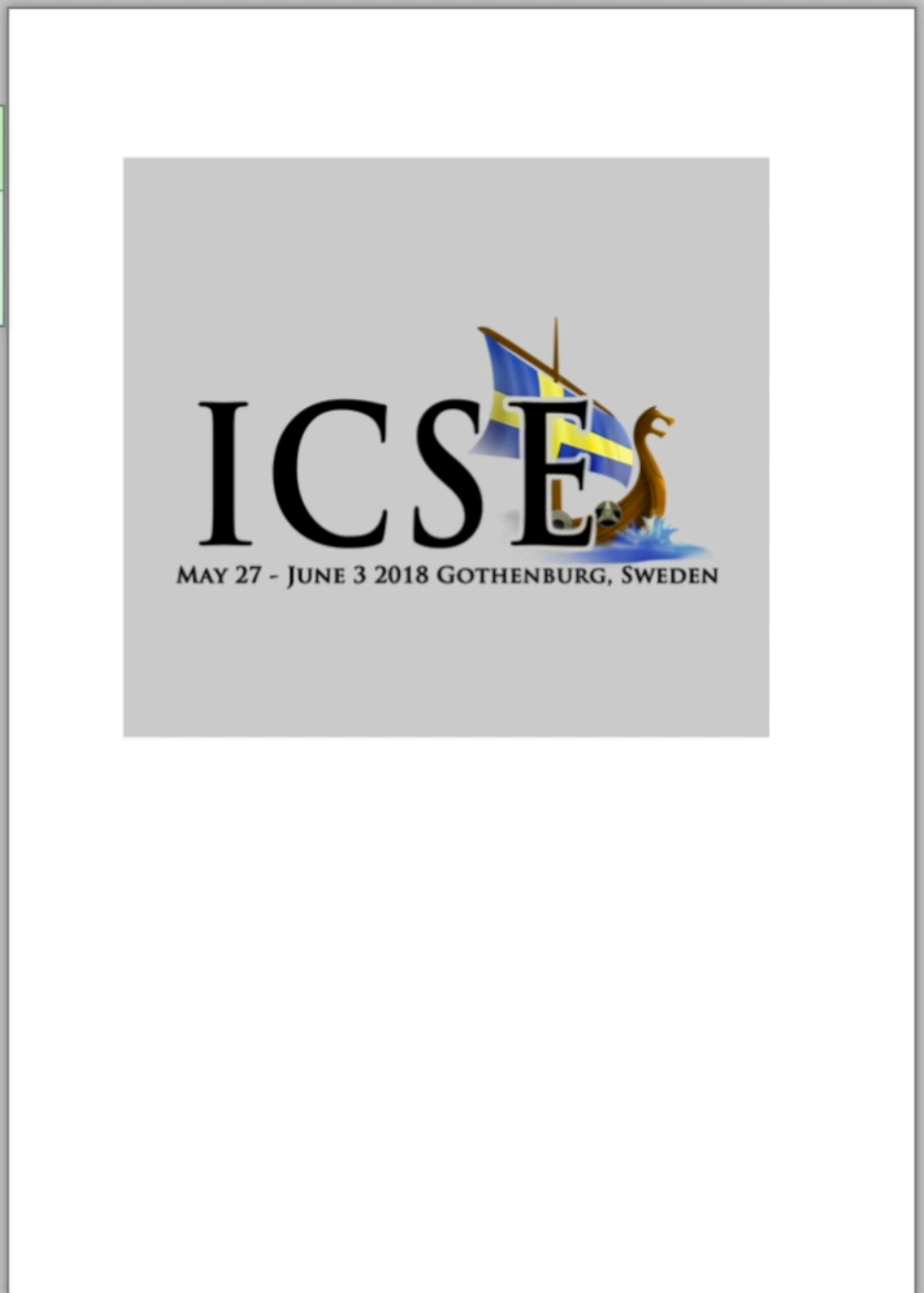
Code Tools

- Undo
- Create Function from Definition ▶
- Inline Definition ▶
- Make Single Line ▶

Abstract image1 over its constants

Abstract image1 over its named constants

```
1 (def image1 (\(x y width height bgColor padding)  
2   (image bgColor x y width height padding image_url)))  
3  
4 (def main  
5   (draw (concat [ (image1 50 65 283 254 "lightgrey" 15) ]
```



Navigation and tool icons:

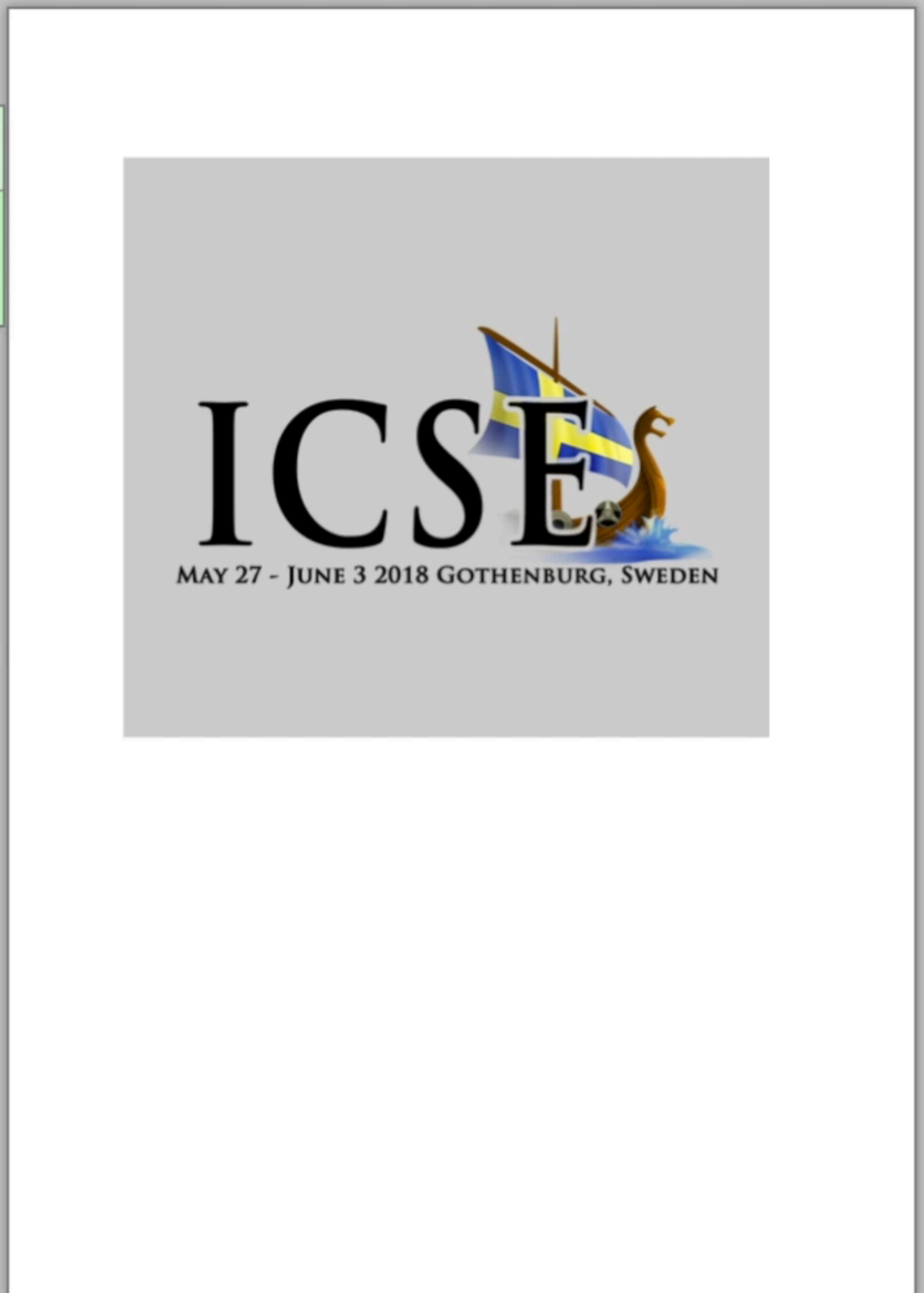
- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Black square icon
- Black circle icon
- Black star icon
- Pen tool icon
- Red star icon
- Raw Stretchy Sticky options

Current file: *Untitled **

Code Tools

- Undo
- Create Function from Definition ▶ Abstract image1 over its constants
- Inline Definition ▶ Abstract image1 over its named constants
- Make Single Line ▶

```
1 (def image1 (\(x y width height)
2   (image "lightgrey" x y width height 15 image_url)))
3
4 (def main
5   (draw (concat [ (image1 50 65 283 254) ])))
```



Navigation and tool icons:

- Mouse cursor
- Text tool (T)
- Line tool
- Black square
- Black circle
- Black star
- Pen tool
- Red star
- Raw
- Stretchy
- Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1 (\(x y width height)
5   (image "lightgrey" x y width height 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65 283 254) ]))))
```

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean U

```
1 (def image_url
2   "img/icse-2018-large"
3
4   (def image1 (\(x y width height)
5     (image "lightgrey" x y width height 15 image_url)))
6
7   (def main
8     (draw (concat [ (image1 50 65 283 254) ]))))
```

Code Tools

- Remove Arguments
- Swap Variable Names and Usages
- Swap Variable Usages
- Inline Definitions

Run ▶



- Mouse cursor
- Text tool (T)
- Line tool
- Black square
- Black circle
- Star
- Fountain pen
- Red star
- Raw
- Stretchy
- Sticky

Current file: *Untitled **

Undo Redo Clean U

```
1 (def image_url
2   "img/icse-2018-large"
3
4 (def image1 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))
```

Code Tools

- Remove Arguments
- Swap Variable Names and Usages
- Swap Variable Usages
- Inline Definitions

Remove Arguments

Run



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool. Below these tools are labels: Raw, Stretchy, Sticky.

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, and a red star. Below these icons are the labels: Raw, Stretchy, and Sticky.

Current file: Untitled

Code Tools

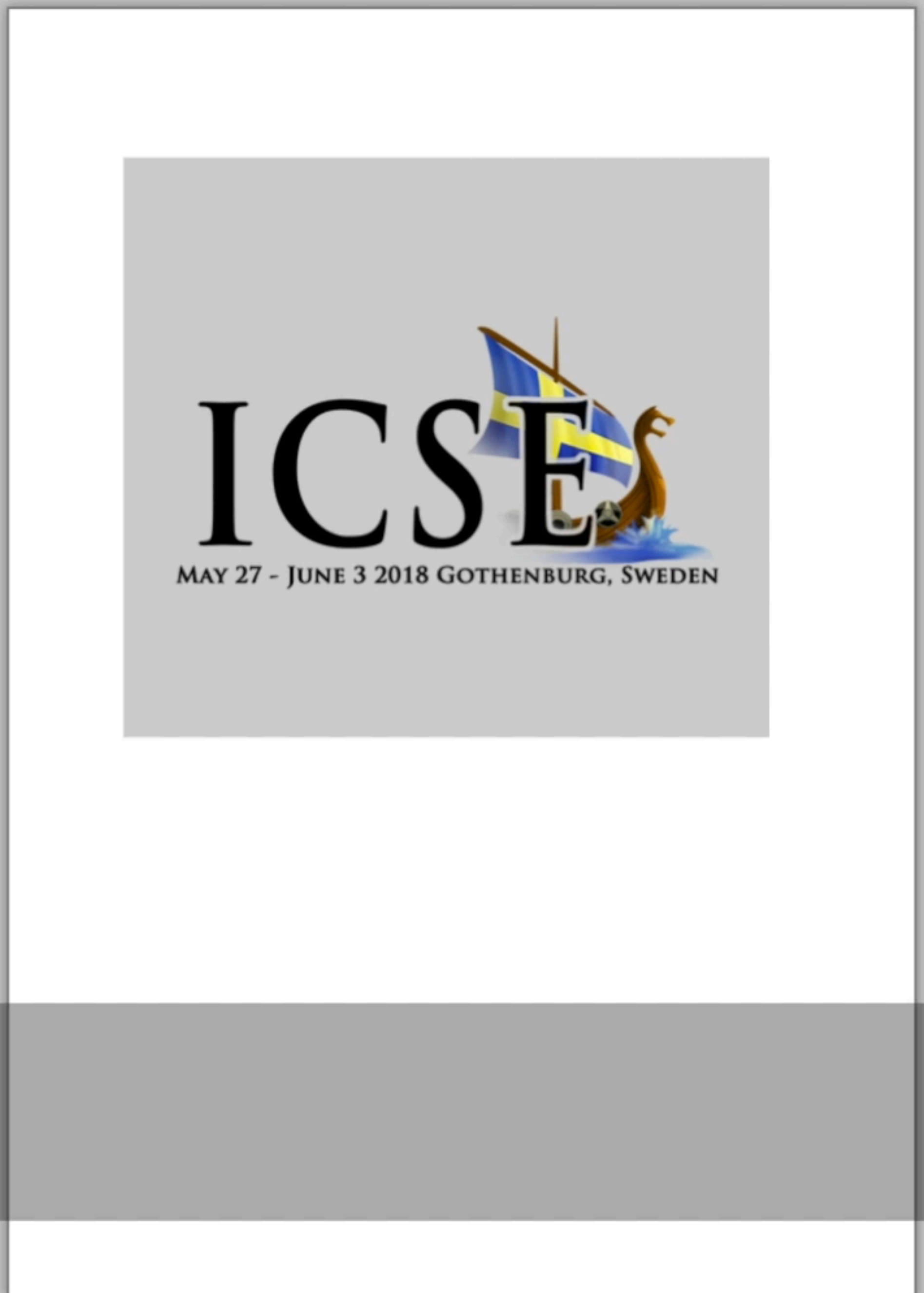
- Create Function from Definition ▶
- Rename *image1* ▶
- Inline Definition ▶

Run ▶

```

1 (def image1 "img/
2   "png")
3
4 (def image1 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))

```



- Mouse cursor icon
- Text tool icon (T)
- Line tool icon
- Square tool icon
- Circle tool icon
- Star tool icon
- Pen tool icon
- Red star tool icon
- Raw Stretchy Sticky options



Current file: Untitled

Undo Run

```
1 (def image1  
2   "img/  
3  
4 (def image1 (\(x y)  
5   (image "lightgrey" x y 283 254 15 image_url)))  
6  
7 (def main  
8   (draw (concat [ (image1 50 65) ])))
```

Code Tools

- Create Function from Definition ▶
- Rename *image1* ▶
- Inline Definition ▶

Rename *image1* to...



Navigation and drawing tools: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, red star, and a menu with 'Raw', 'Stretchy', and 'Sticky' options.

Current file: Untitled

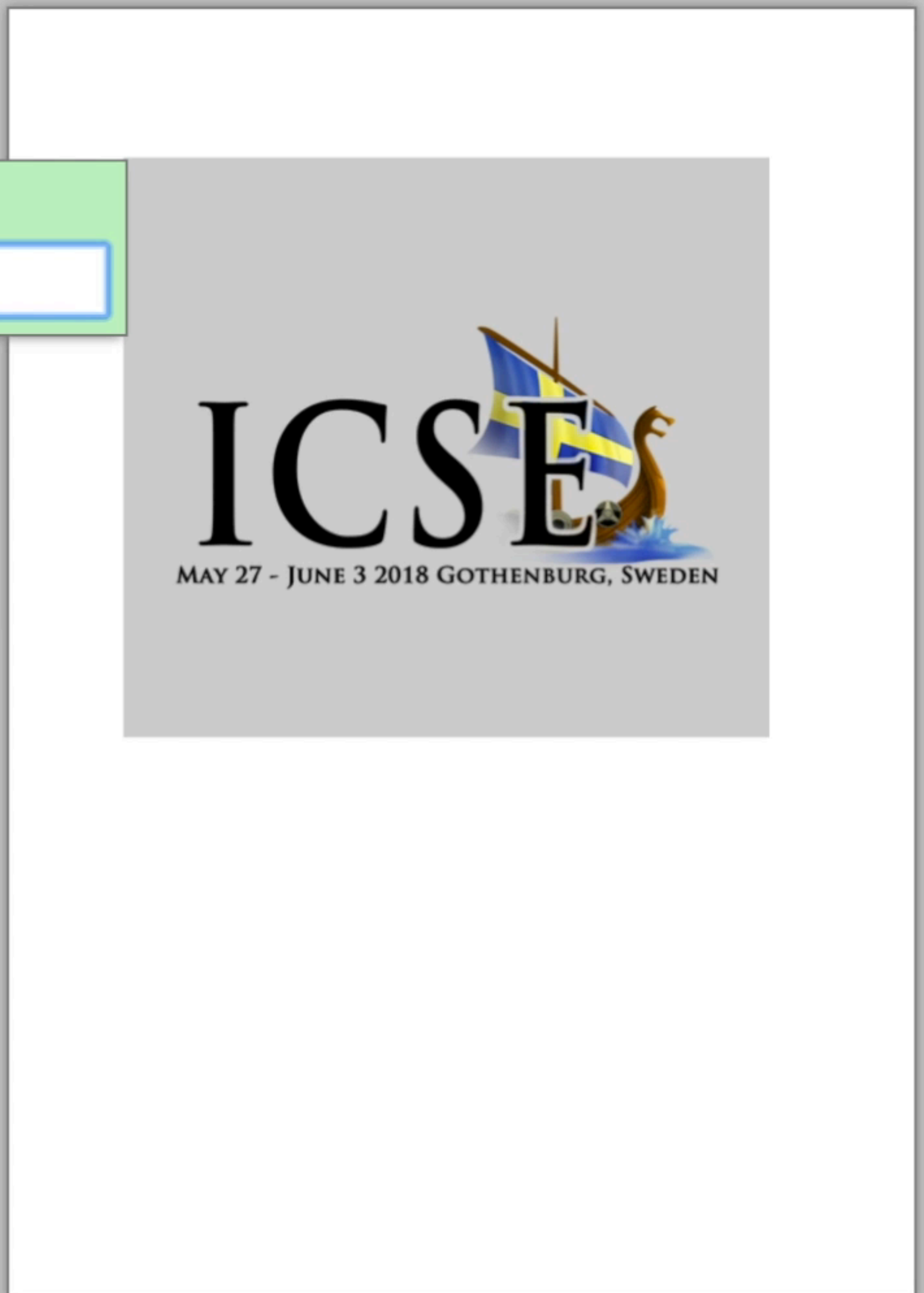
Undo Code Tools Run ▶

1 (def image1
2 "img/
3
4 (def image1 (\(x y)
5 (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8 (draw (concat [(image1 50 65)])))

Code Tools

- Create Function from Definition ▶
- Rename *image1* ▶
- Inline Definition ▶

Rename *image1* to *icse2018*



Navigation icons: mouse cursor, text tool (T), line tool, square, circle, star, fountain pen, red star.

Raw
Stretchy
Sticky

Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def icse2018 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (icse2018 50 65) ])))
```



Navigation and drawing tools including: mouse cursor, text tool (T), line tool, square tool, circle tool, star tool, fountain pen tool, and a red star tool.

Raw
Stretchy
Sticky



Current file: *Untitled **

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def icse2018 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (icse2018 50 65) ])))
```

Raw
Stretchy
Sticky

Deuce

**Structure
Select**

**Short
Menu**

Defaults

Deuce

Structure Select

Short Menu

Defaults

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
        [x y] [100 100]  
        (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

```
(def image1  
  (let [width height]  
        [x y] [100 100]  
        (image "lightgrey"
```

Deuce

Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

```
(def image1  
  (let [width height]  
    (let [x y] [100 100]  
      (image "lightgrey"
```

Short Menu

Code Tools

- Create Function from Definition ▶
- Inline Definition ▶
- Make Single Line ▶

Code Tools

- Move Definition ▶
- Duplicate Definition ▶

Defaults

Deuce

Structure Select

Short Menu

Defaults

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

```
(def image1  
  (let [width height]  
    (let [x y] [100 100]  
      (image "lightgrey"
```

| Code Tools | |
|---------------------------------|--------------------------------------|
| Create Function from Definition | ▶ Abstract image1 over its constants |
| Inline Definition | ▶ Abstract image1 over its named |
| Make Single Line | ▶ constants |

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

Deuce more effective
than **Traditional**?

Deuce more effective
than **Traditional**?

Deuce preferred
to **Traditional**?

Deuce

Deuce

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"
```

Short Menu

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

Defaults

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Short Menu

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

Defaults

Traditional

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Short Menu

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

Defaults

Traditional “Text-Select Mode”

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Short Menu

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

Defaults

Traditional “Text-Select Mode”

Text Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Short Menu

| Code Tools | |
|----------------------|-------------------------|
| Move Definition | ▶ Move width and height |
| Duplicate Definition | ▶ |

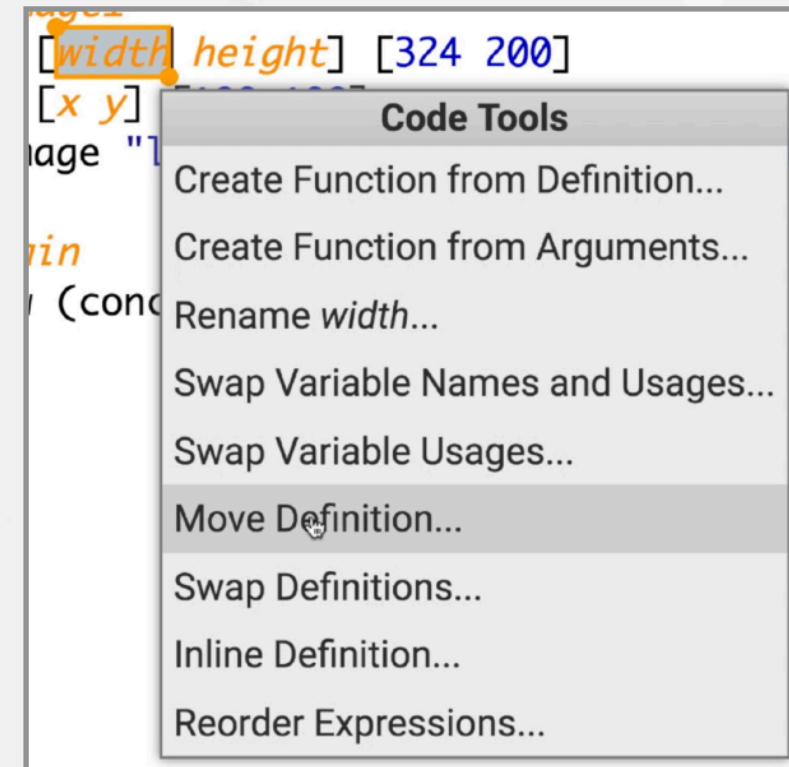
Defaults

Traditional “Text-Select Mode”

Text Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey")
```

Right-Click Menu

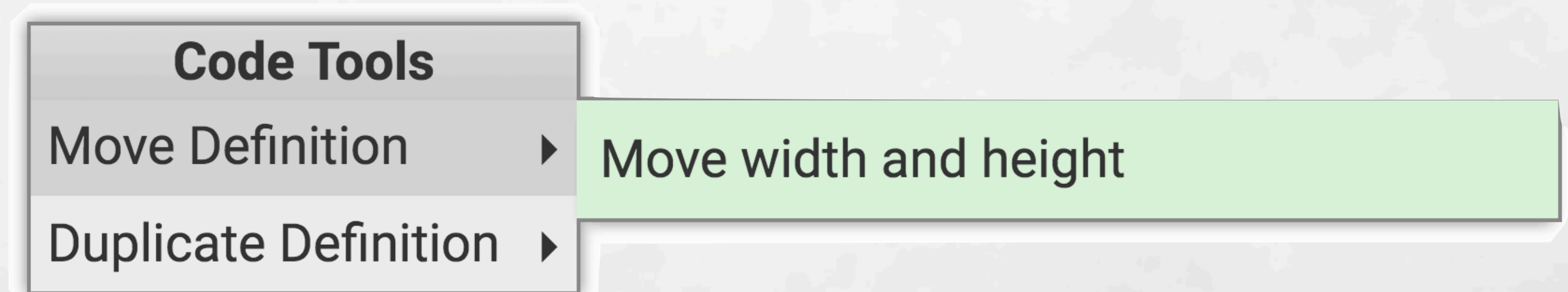


Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey")
```

Short Menu



Defaults

Traditional “Text-Select Mode”

Text Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Right-Click Menu

- Create Function from Definition...
- Create Function from Arguments...
- Rename width...
- Swap Variable Names and Usages...
- Swap Variable Usages...
- Move Definition...
- Swap Definitions...
- Inline Definition...
- Reorder Expressions...

Select Arguments

Move Definition

Requirements

- Select one or more variable definitions and one target position (i.e. whitespace)

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
      (image "lightgrey"))
```

Short Menu

Code Tools

- Move Definition ▶ Move width and height
- Duplicate Definition ▶

Defaults

Traditional “Text-Select Mode”

Text Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
    (image "lightgrey")
```

Right-Click Menu

- Create Function from Definition...
- Create Function from Arguments...
- Rename width...
- Swap Variable Names and Usages...
- Swap Variable Usages...
- Move Definition...
- Swap Definitions...
- Inline Definition...
- Reorder Expressions...

Select Arguments

Move Definition

Requirements

- Select one or more variable definitions and one target position (i.e. whitespace)

Defaults

Move Definition

Requirements

- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

Code Updates

Move width and height

Deuce “Box-Select Mode”

Structure Select

```
(def image1  
  (let [width height]  
    (let [x y] [100 100])  
    (image "lightgrey")
```

Short Menu

Code Tools

- Move Definition ▶ Move width and height
- Duplicate Definition ▶


Defaults

Tutorial

Tutorial

Head-to-Head Tasks (2x; once per mode)

```
1
2 (def rect1
3   (let [x 20
4         y 20
5         height 250
6         width 80]
7     (let fill "black"
8       (rect fill x y height width))))
9
10 (def main
11   (draw [rect1]))
12
13 ; The final program should look something like:
14 ;
15 ;
16 ; (def rect1
17 ;   (let [fill x y width height] ["black" 20 20 80 250]
18 ;     (rect fill x y width height)))
19 ;
20 ; (def main
21 ;   (draw [rect1]))
22
```




Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.

Goals:

- The programmer intended the rectangle to be 80 pixels tall and 20 pixels wide, but the `height` and `width` arguments to `rect` are in the wrong order. Swap them.
- Rearrange the five variable definitions into a single tuple definition. The order of variables should match the order of arguments to `rect`.

```
1
2 (def connectedCircles
3   (let [startX 50
4         endY 50
5         startY 70
6         endX 150]
7     [(circle "gray" startX startY 30)
8      (circle "gray" endX endY 30)
9      (line "gray" 10 startX startY endX endY)
10    ]))
11
12 (def main
13   (draw connectedCircles))
14
15 ; The final program should look something like:
16 ;
17 ;
18 ; (def connectedCircles (\startX startY endX endY)
19 ;   [(circle "gray" startX startY 30)
20 ;    (circle "gray" endX endY 30)
21 ;    (line "gray" 10 startX startY endX endY)
22 ;    ])
23 ;
24 ; (def main
25 ;   (draw (connectedCircles 50 70 150 50)))
26
```




Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.

Goals:

- Turn `connectedCircles` into a function that takes `startX`, `startY`, `endX`, and `endY` arguments, and draws two gray circles at `(startX, startY)` and `(endX, endY)`, connected by a line.

```
1
2 (def rect1
3   (let [fill x y w h] ["red" 30 30 50 70]
4     (rect fill x y w h)))
5
6 (def rect2
7   (let [fill x y w h] ["green" 109 53 50 70]
8     (rect fill x y w h)))
9
10 (def rect3
11   (let [fill x y w h] ["blue" 192 35 50 70]
12     (rect fill x y w h)))
13
14 (def main
15   (draw [rect1 rect2 rect3]))
16
17 ; The final program should look something like:
18 ;
19 ;
20 ; (def rect_50_70 (\fill x y
21 ;   (let [w h] [50 70]
22 ;     (rect fill x y w h))))
23 ;
24 ; (def rect1
25 ;   (rect_50_70 "red" 30 30))
26 ;
27 ; (def rect2
28 ;   (rect_50_70 "green" 109 53))
```




Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.

Goals:

- The three rectangle definitions share a lot of identical code. Create a function `rect_50_70` that generates a 50 x 70 rectangle given color and position arguments, and define `rect1`, `rect2`, and `rect3` in terms of `rect_50_70`.

```
1
2 (def ring (\
3   (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray")
4     (circle fill 150 150 (* 30 i))))
5
6 (def target (\startIndex endIndex)
7   (map ring (reverse (range startIndex endIndex))))
8
9 (def main
10   (draw (target 1 4)))
11
12 ; The final program should look something like:
13 ;
14 ;
15 ; (def target (\numRings cx cy num)
16 ;   (let ring (\
17 ;     (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray")
18 ;       (circle fill cx cy (* num i))))
19 ;     (map ring (reverse (range 1 numRings)))))
20 ;
21 ; (def main
22 ;   (draw (target 4 150 150 30)))
23
```







Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.

Goals:



- Remove the `startIndex` argument; its value should always be 1.
- Rename `endIndex` to `numRings`.
- Move the `ring` function inside the `target` definition.
- Add the center position and ring width as arguments to `target`.

Tutorial

Head-to-Head Tasks (2x; once per mode)



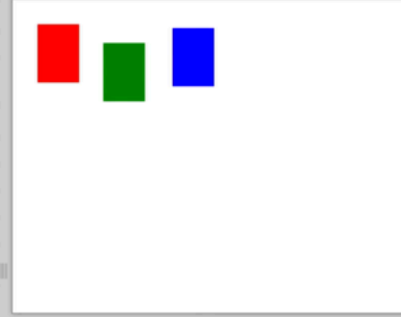

| | |
|---|---|
| <pre>1 2 (def rect1 3 (let [x 20 4 y 20 5 height 250 6 width 80] 7 (let fill "black" 8 (rect-fill x y height width)))) 9 10 (def main 11 (draw [rect1])) 12 13 ; The final program should look something like: 14 15 : 16 : (def rect1 17 : (let [fill x y width height] ["black" 20 20 80 250] 18 : (rect-fill x y width height))) 19 : 20 : (def main 21 : (draw [rect1])) 22 :</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">The programmer intended the rectangle to be 800 pixels tall and 80 pixels wide, but the <code>height</code> and <code>width</code> arguments to <code>rect-fill</code> are in the wrong order. Swap them.Rearrange the five variable definitions into a single tuple definition. The order of variables should match the order of arguments to <code>rect-fill</code>. |
| <pre>1 2 (def connectedCircles 3 (let [startX 50 4 endY 50 5 startY 70 6 endX 150] 7 [(circle "gray" startX startY 30) 8 (circle "gray" endX endY 30) 9 (line "gray" 10 startX startY endX endY)])) 10 11 12 (def main 13 (draw connectedCircles)) 14 15 ; The final program should look something like: 16 17 : (def connectedCircles (lambda (startX startY endX endY) 18 : [(circle "gray" startX startY 30) 19 : (circle "gray" endX endY 30) 20 : (line "gray" 10 startX startY endX endY)])) 21 : 22 : 23 : (def main 24 : (draw (connectedCircles 50 70 150 50))) 25 : 26 :</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Turn <code>connectedCircles</code> into a function that takes <code>startX</code>, <code>startY</code>, <code>endX</code>, and <code>endY</code> arguments, and draws two gray circles at <code>(startX, startY)</code> and <code>(endX, endY)</code>, connected by a line. |
| <pre>1 2 (def rect1 3 (let [fill x y w h] ["red" 30 30 50 70] 4 (rect-fill x y w h))) 5 6 (def rect2 7 (let [fill x y w h] ["green" 100 53 50 70] 8 (rect-fill x y w h))) 9 10 (def rect3 11 (let [fill x y w h] ["blue" 192 35 50 70] 12 (rect-fill x y w h))) 13 14 (def main 15 (draw [rect1 rect2 rect3])) 16 17 ; The final program should look something like: 18 19 : (def rect1 (lambda (fill x y w h) (rect-fill x y w h)) 20 : (rect-fill 30 30 50 70)) 21 : 22 : (def rect2 (lambda (fill x y w h) (rect-fill x y w h)) 23 : (rect-fill 100 53 50 70)) 24 : 25 : (def rect3 (lambda (fill x y w h) (rect-fill x y w h)) 26 : (rect-fill 192 35 50 70)) 27 : 28 : (def main 29 : (draw [rect1 rect2 rect3])) 30 :</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">The three rectangle definitions share a lot of identical code. Create a function <code>rect-fill</code> that generates a <code>30 x 70</code> rectangle given color and position arguments, and define <code>rect1</code>, <code>rect2</code>, and <code>rect3</code> in terms of <code>rect-fill</code>. |
| <pre>1 2 (def ring (lambda (cx cy num) 3 (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray") 4 (circle-fill 150 150 (* 30 i))))) 5 6 (def target (lambda (startIndex endIndex) 7 (map ring (reverse (range startIndex endIndex))))) 8 9 (def main 10 (draw (target 1 4))) 11 12 ; The final program should look something like: 13 14 : (def ring (lambda (cx cy num) 15 : (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray") 16 : (circle-fill cx cy (* num i))))) 17 : 18 : (def target (lambda (startIndex endIndex) 19 : (map ring (reverse (range 1 numRings))))) 20 : 21 : (def main 22 : (draw (target 4 150 30))) 23 :</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Remove the <code>startIndex</code> argument; its value should always be 1.Rename <code>endIndex</code> to <code>numRings</code>.Move the <code>ring</code> function inside the <code>target</code> definition.Add the center position and ring width as arguments to <code>target</code>. |

Mix & Match Tasks (free to use both modes)

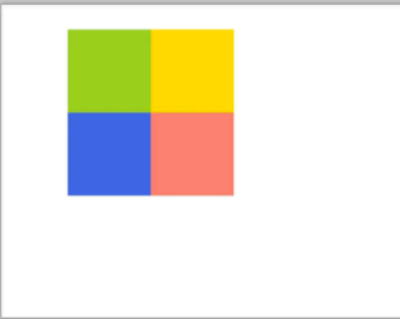

| | |
|---|---|
| <pre>1 2 (def fourSquares 3 (let [x y w] [80 30 100] 4 [(rect "yellowgreen" (+ x (* 0 w)) (+ y (* 0 w)) w w) 5 (rect "gold" (+ x (* 1 w)) (+ y (* 0 w)) w w) 6 (rect "royalblue" (+ x (* 0 w)) (+ y (* 1 w)) w w) 7 (rect "salmon" (+ x (* 1 w)) (+ y (* 1 w)) w w)])) 8 9 10 11 (def main 12 (draw fourSquares)) 13 14 ; The final program should look something like: 15 16 : (def fourSquares (lambda (x y w) (let [fill num num2] 17 : [(rect-fill (+ x (* num w)) (+ y (* num2 w)) w w) 18 : (rect-fill (+ x (* num w)) (+ y (* num2 w)) w w) 19 : (rect-fill (+ x (* num w)) (+ y (* num2 w)) w w) 20 : (rect-fill (+ x (* num w)) (+ y (* num2 w)) w w)])) 21 : 22 : (oneCorner (lambda (topLeft topRight botLeft botRight) 23 : (oneCorner topLeft 1 0) 24 : (oneCorner botLeft 1 1) 25 : (oneCorner botRight 1 1)])) 26 : 27 : (def main 28 : (draw fourSquares)) 29 :</pre> |  <p>Use TEXT-SELECT MODE and/or BOX-SELECT MODE to perform the edits below. You can mix both modes, like in the tutorial. Text edits are disabled. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Introduce a helper function called <code>oneCorner</code> that factors the code that is common to the four calls to <code>rect-fill</code>.Turn <code>fourSquares</code> into a function that takes <code>x</code>, <code>y</code>, and <code>w</code> arguments, as well as color arguments called <code>lookLeft</code>, <code>lookRight</code>, <code>lookLeft</code>, and <code>lookRight</code>. |
| <pre>1 2 (def rectangle 3 (rect-fill x y w h)) 4 5 (def line1 6 (line "white" 5 20 30 (+ 20 100) (+ 30 120))) 7 8 (def line2 9 (line "white" 5 20 (+ 30 120) (+ 20 (/ 100 2)) (+ 30 (/ 120 2)))) 10 11 (def logo 12 [rectangle line1 line2]) 13 14 (def main 15 (draw logo)) 16 17 ; The final program should look something like: 18 19 : (def [x y w h] [20 30 100 120]) 20 : (def [fill stroke strokeWidth] ["black" "white" 5]) 21 : 22 : (def rectangle 23 : (rect-fill x y w h)) 24 : 25 : (def line1 26 : (line stroke strokeWidth x (+ x w) (+ y h))) 27 : 28 : (def line2 29 : (line stroke strokeWidth x (+ y h) (+ x (/ w 2)) (+ y (/ h 2)))) 30 : 31 : (def logo 32 : [rectangle line1 line2]) 33 :</pre> |  <p>The initial program draws a 100 x 100 pixel lambda icon at xy-position (20, 30), but the use of duplicated constants requires many changes if we want to draw the icon at a different position or change the style of the lines.</p> <p>Use TEXT-SELECT MODE and/or BOX-SELECT MODE to perform the edits below. You can mix both modes, like in the tutorial. Text edits are disabled. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Define and use four new variables called <code>x</code>, <code>y</code>, <code>w</code>, and <code>h</code> for the x-position, y-position, width, and height, respectively, of the logo. These variables should be defined in a single <code>let</code>. |

Tutorial

Head-to-Head Tasks (2x; once per mode)

| | |
|--|---|
| <pre>1 2 (def rect1 3 (let [x 20 4 y 20 5 height 250 6 width 80] 7 (let fill "black" 8 (rect-fill x y height width)))) 9 10 (def main 11 (draw [rect1])) 12 13 ; The final program should look something like: 14 15 ; 16 (def rect1 17 (let [fill x y width height] ["black" 20 20 80 250] 18 (rect-fill x y width height))) 19 20 (def main 21 (draw [rect1])) 22</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">The programmer intended the rectangle to be 800 pixels tall and 80 pixels wide, but the <code>height</code> and <code>width</code> arguments to <code>rect-fill</code> are in the wrong order. Swap them.Rearrange the five variable definitions into a single tuple definition. The order of variables should match the order of arguments to <code>rect-fill</code>. |
| <pre>1 2 (def connectedCircles 3 (let [startX 50 4 endY 50 5 startY 70 6 endX 150] 7 [(circle "gray" startX startY 30) 8 (circle "gray" endX endY 30) 9 (line "gray" 10 startX startY endX endY)])) 10 11 (def main 12 (draw connectedCircles)) 13 14 ; The final program should look something like: 15 16 (def connectedCircles (\startX startY endX endY) 17 [(circle "gray" startX startY 30) 18 (circle "gray" endX endY 30) 19 (line "gray" 10 startX startY endX endY)])) 20 21 (def main 22 (draw connectedCircles)) 23 24 ; 25 (def main 26 (draw (connectedCircles 50 70 150 50))) 27 28</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Turn <code>connectedCircles</code> into a function that takes <code>startX</code>, <code>startY</code>, <code>endX</code>, and <code>endY</code> arguments, and draws two gray circles at <code>(startX, startY)</code> and <code>(endX, endY)</code>, connected by a line. |
| <pre>1 2 (def rect1 3 (let [fill x y w h] ["red" 30 30 50 70] 4 (rect-fill x y w h))) 5 6 (def rect2 7 (let [fill x y w h] ["green" 109 53 50 70] 8 (rect-fill x y w h))) 9 10 (def rect3 11 (let [fill x y w h] ["blue" 192 35 50 70] 12 (rect-fill x y w h))) 13 14 (def main 15 (draw [rect1 rect2 rect3])) 16 17 ; The final program should look something like: 18 19 (def rect1 20 (def rect_50_70 (\fill x y) 21 (let [w h] [50 70] 22 (rect-fill x y w h)))) 23 24 (def rect1 25 (rect_50_70 "red" 30 30)) 26 27 (def rect2 28 (rect_50_70 "green" 109 53)) 29</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">The three rectangle definitions share a lot of identical code. Create a function <code>rect_50_70</code> that generates a 50 x 70 rectangle given color and position arguments, and define <code>rect1</code>, <code>rect2</code>, and <code>rect3</code> in terms of <code>rect_50_70</code>. |
| <pre>1 2 (def ring (\ 3 (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray") 4 (circle-fill 150 150 (* 30 i)))))) 5 6 (def target (\startIndex endIndex) 7 (map ring (reverse (range startIndex endIndex)))) 8 9 (def main 10 (draw (target 1 4))) 11 12 ; The final program should look something like: 13 14 (def ring (\ 15 (let ring (\ 16 (let fill (if (= 0 (mod i 2)) "firebrick" "lightgray") 17 (circle-fill cx cy (* num i)))))) 18 (map ring (reverse (range 1 numRings))))) 19 20 (def main 21 (draw (target 4 150 30))) 22 23</pre> |  <p>Use only BOX-SELECT MODE to perform the edits below. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Remove the <code>startIndex</code> argument; its value should always be 1.Rename <code>endIndex</code> to <code>numRings</code>.Move the <code>ring</code> function inside the <code>target</code> definition.Add the center position and ring width as arguments to <code>target</code>. |

Mix & Match Tasks (free to use both modes)

| | |
|--|---|
| <pre>1 2 (def fourSquares 3 (let [x y w] [80 30 100] 4 [(rect "yellowgreen" (+ x (* 0 w)) (+ y (* 0 w)) w) 5 (rect "gold" (+ x (* 1 w)) (+ y (* 0 w)) w) 6 (rect "royalblue" (+ x (* 0 w)) (+ y (* 1 w)) w) 7 (rect "salmon" (+ x (* 1 w)) (+ y (* 1 w)) w)])) 8 9 10 11 (def main 12 (draw fourSquares)) 13 14 ; The final program should look something like: 15 16 (def fourSquares (\(x y w topLeft topRight botLeft botRight) 17 (let oneCorner (\(fill num num2) 18 (rect-fill (+ x (* num w)) (+ y (* num2 w)) w)) 19 [(oneCorner topLeft 0 0) 20 (oneCorner topRight 1 0) 21 (oneCorner botLeft 0 1) 22 (oneCorner botRight 1 1)])) 23)) 24 25 (def main 26 (draw fourSquares)) 27 28</pre> |  <p>Use TEXT-SELECT MODE and/or BOX-SELECT MODE to perform the edits below. You can mix both modes, like in the tutorial. Text edits are disabled. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Introduce a helper function called <code>oneCorner</code> that factors the code that is common to the four calls to <code>rect-fill</code>.Turn <code>fourSquares</code> into a function that takes <code>x</code>, <code>y</code>, and <code>w</code> arguments, as well as color arguments called <code>topLeft</code>, <code>topRight</code>, <code>botLeft</code>, and <code>botRight</code>. |
| <pre>1 2 (def rectangle 3 (rect-fill x y w h)) 4 5 (def line1 6 (line "white" 5 20 30 (+ 20 100) (+ 30 120))) 7 8 (def line2 9 (line "white" 5 20 (+ 30 120) (+ 20 (/ 100 2)) (+ 30 (/ 120 2)))) 10 11 (def logo 12 [rectangle line1 line2]) 13 14 (def main 15 (draw logo)) 16 17 ; The final program should look something like: 18 19 (def [x y w h] [20 30 100 120]) 20 (def [fill stroke strokeWidth] ["black" "white" 5]) 21 22 (def rectangle 23 (rect-fill x y w h)) 24 25 (def line1 26 (line stroke strokeWidth x (+ x w) (+ y h))) 27 28 (def line2 29 (line stroke strokeWidth x (+ y h) (+ x (/ w 2)) (+ y (/ h 2)))) 30 31 (def logo 32 [rectangle line1 line2]) 33</pre> |  <p>The initial program draws a 100 x 100 pixel lambda icon at xy-position (20, 30), but the use of duplicated constants requires many changes if we want to draw the icon at a different position or change the style of the lines.</p> <p>Use TEXT-SELECT MODE and/or BOX-SELECT MODE to perform the edits below. You can mix both modes, like in the tutorial. Text edits are disabled. When you are done, press Next Step.</p> <p>Goals:</p> <ul style="list-style-type: none">Define and use four new variables called <code>x</code>, <code>y</code>, <code>w</code>, and <code>h</code> for the x-position, y-position, width, and height, respectively, of the logo. These variables should be defined in a single <code>let</code>. |

Exit Survey

Deuce more effective than **Traditional**?

Deuce more effective than Traditional?

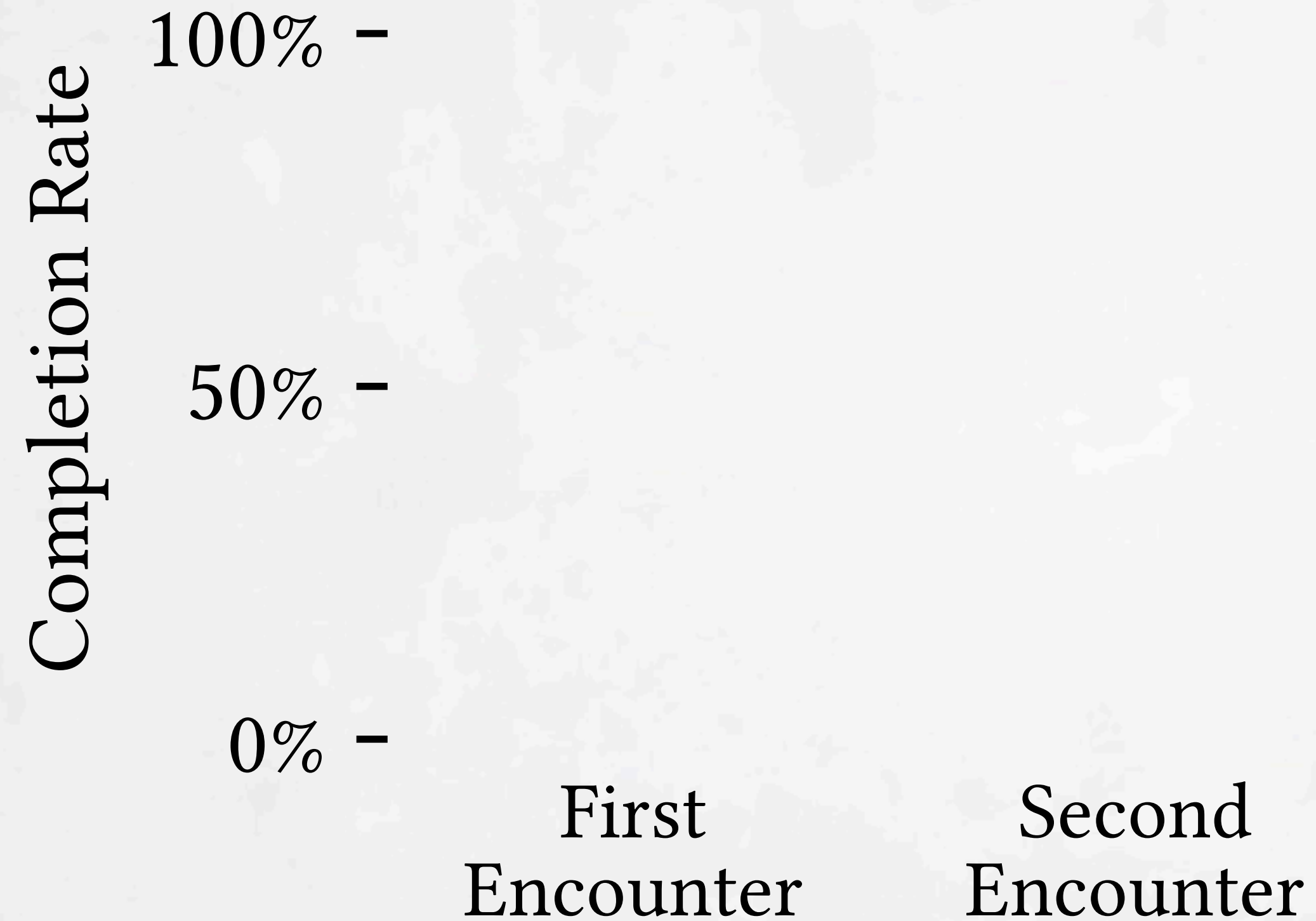
Completion Rate

100% -

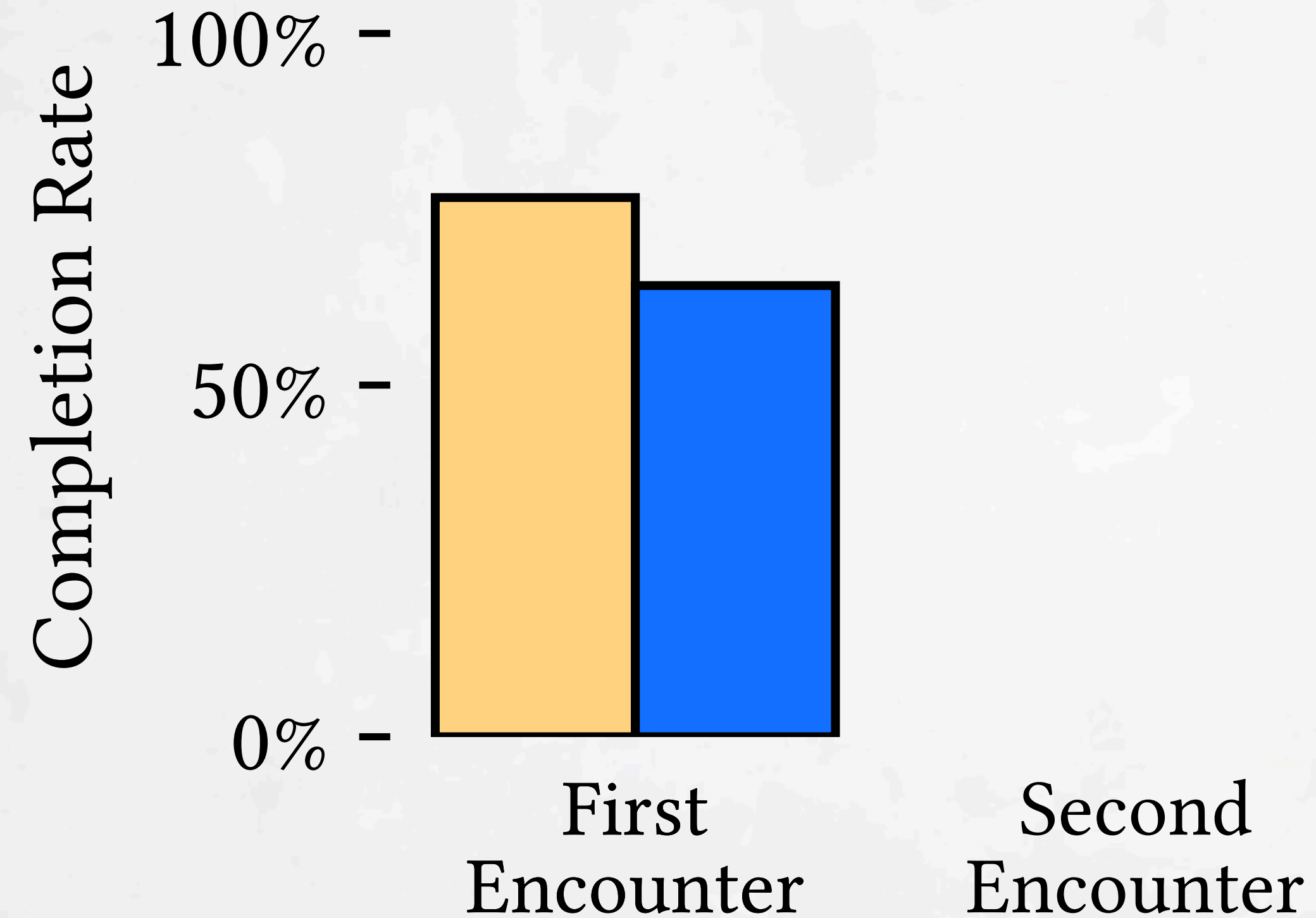
50% -

0% -

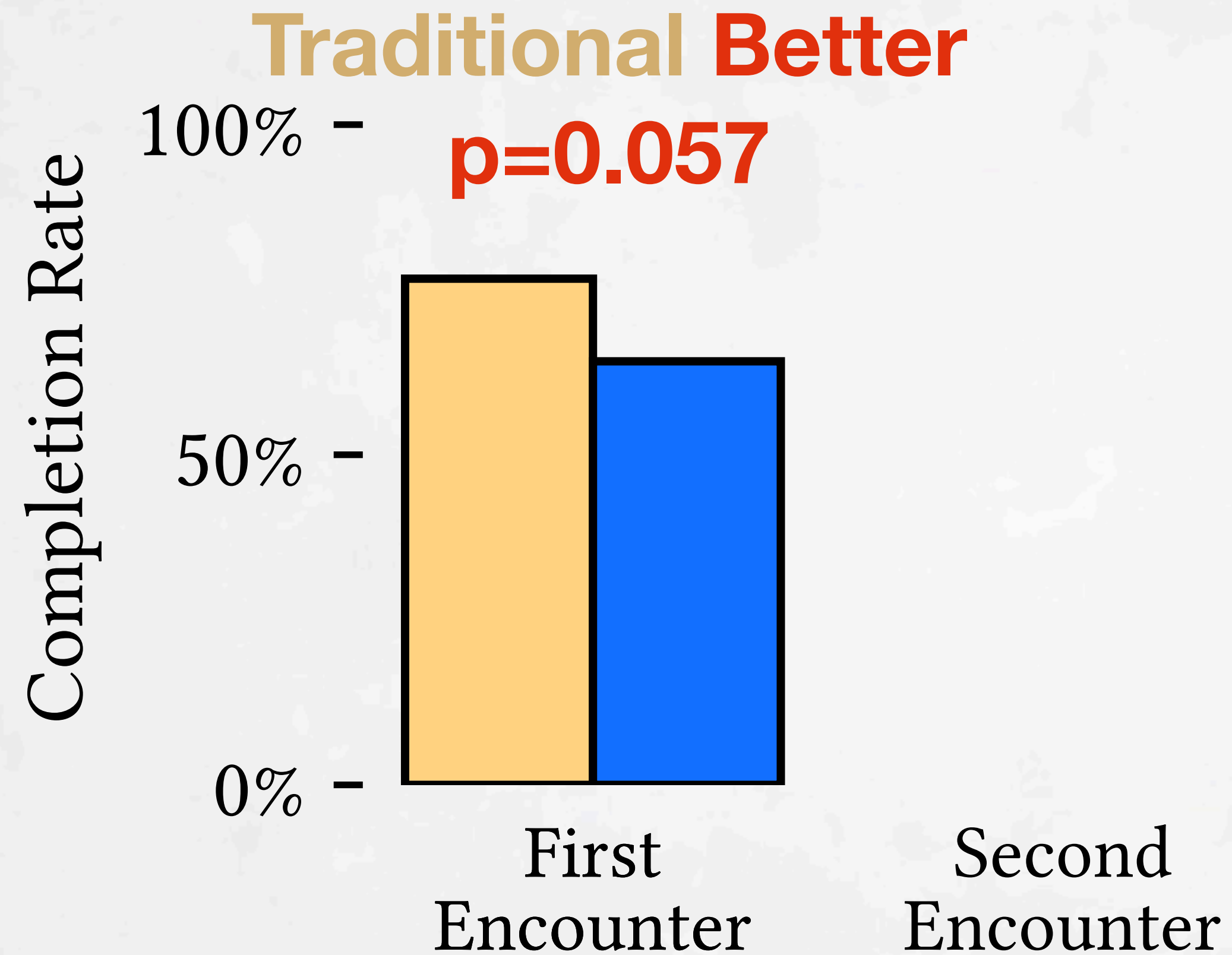
Deuce more effective than Traditional?



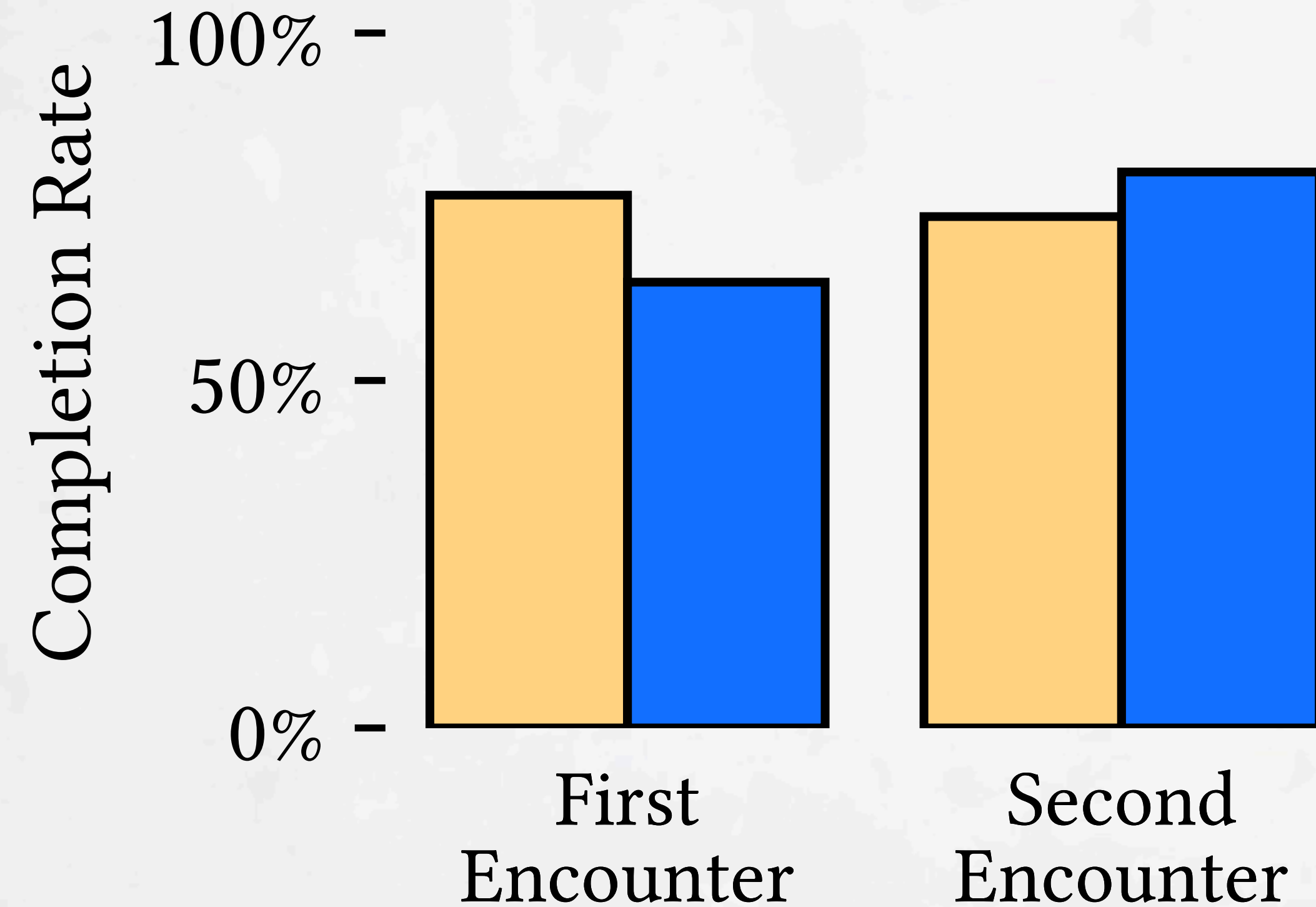
Deuce more effective than Traditional?



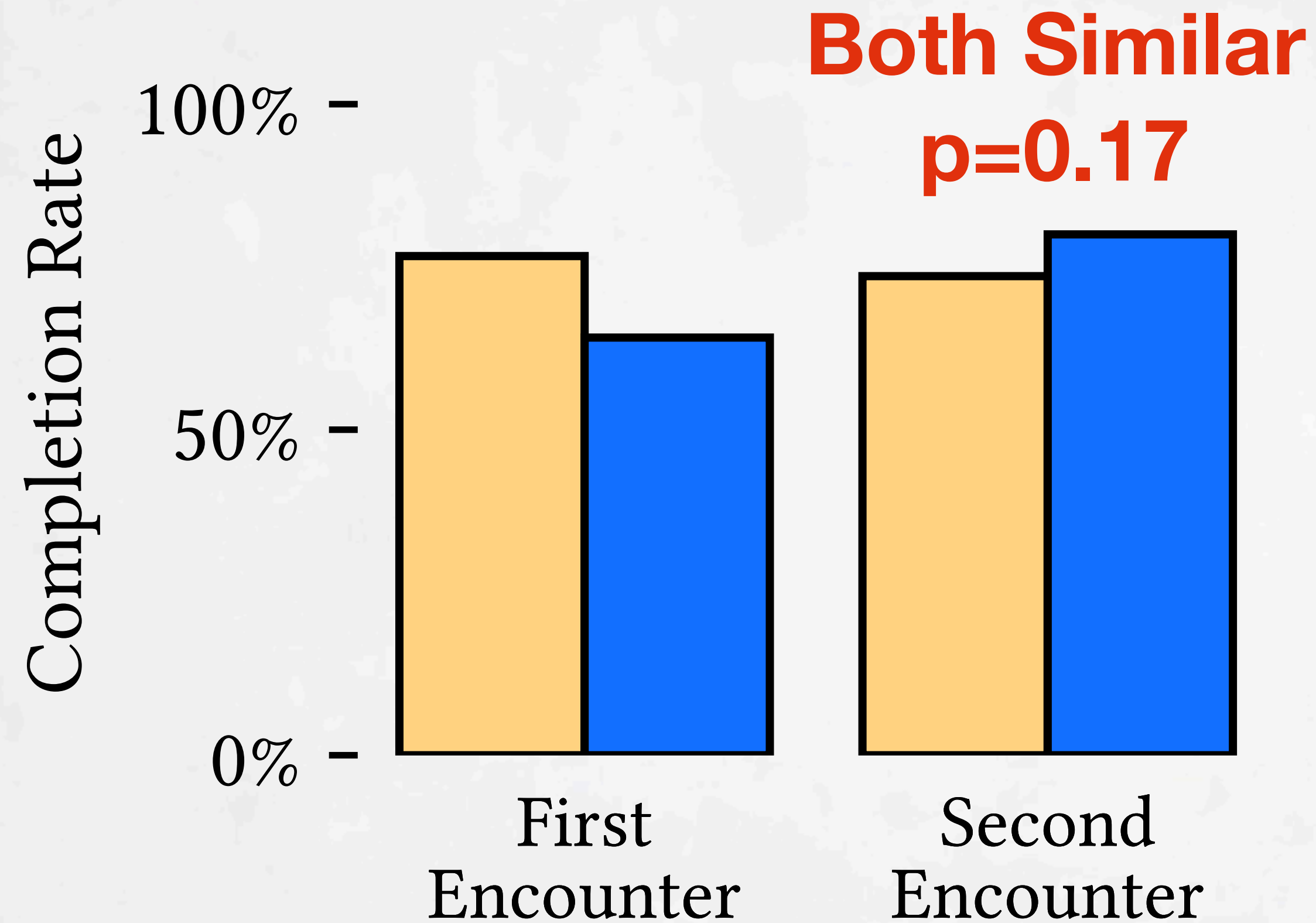
Deuce more effective than Traditional?



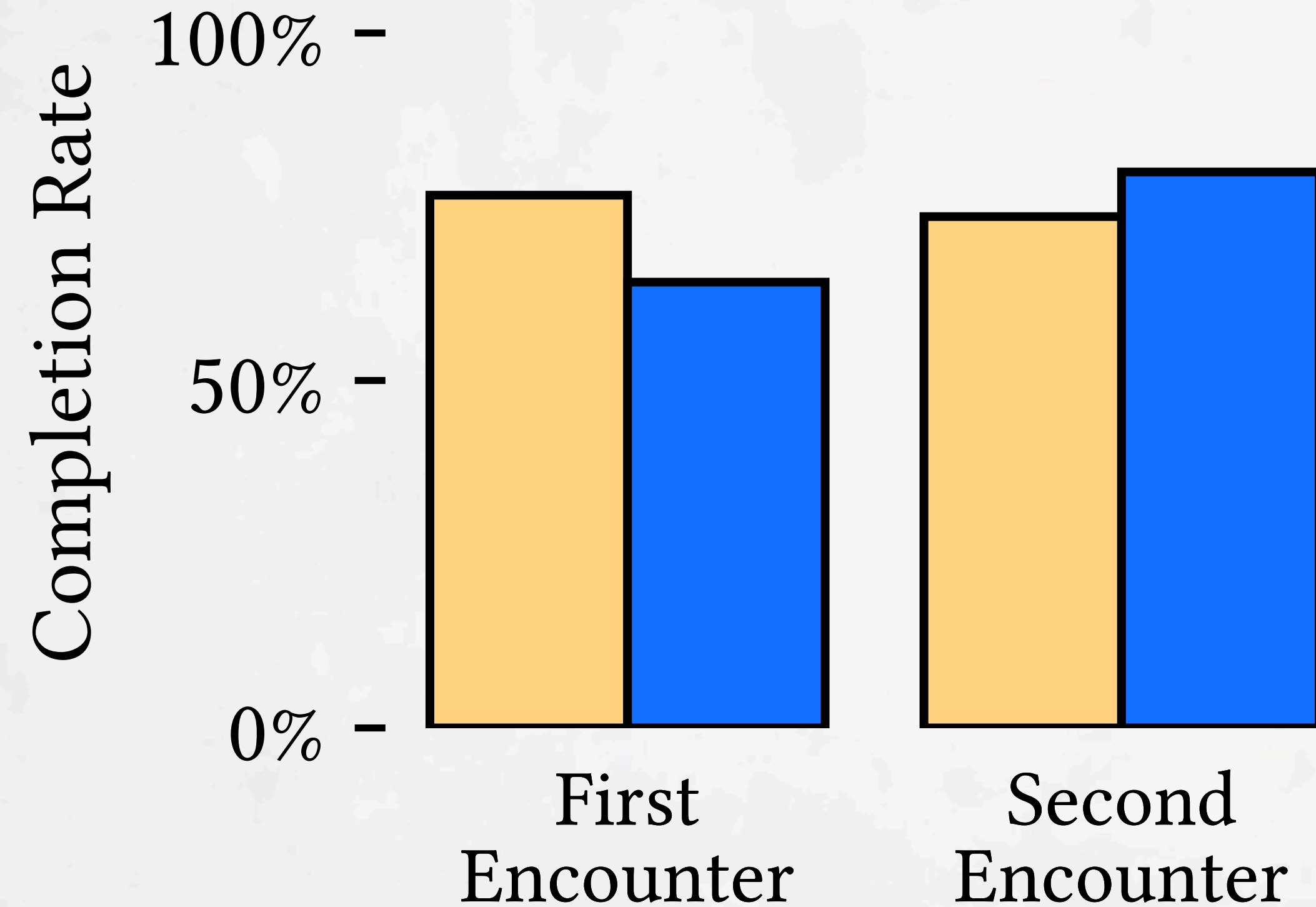
Deuce more effective than Traditional?



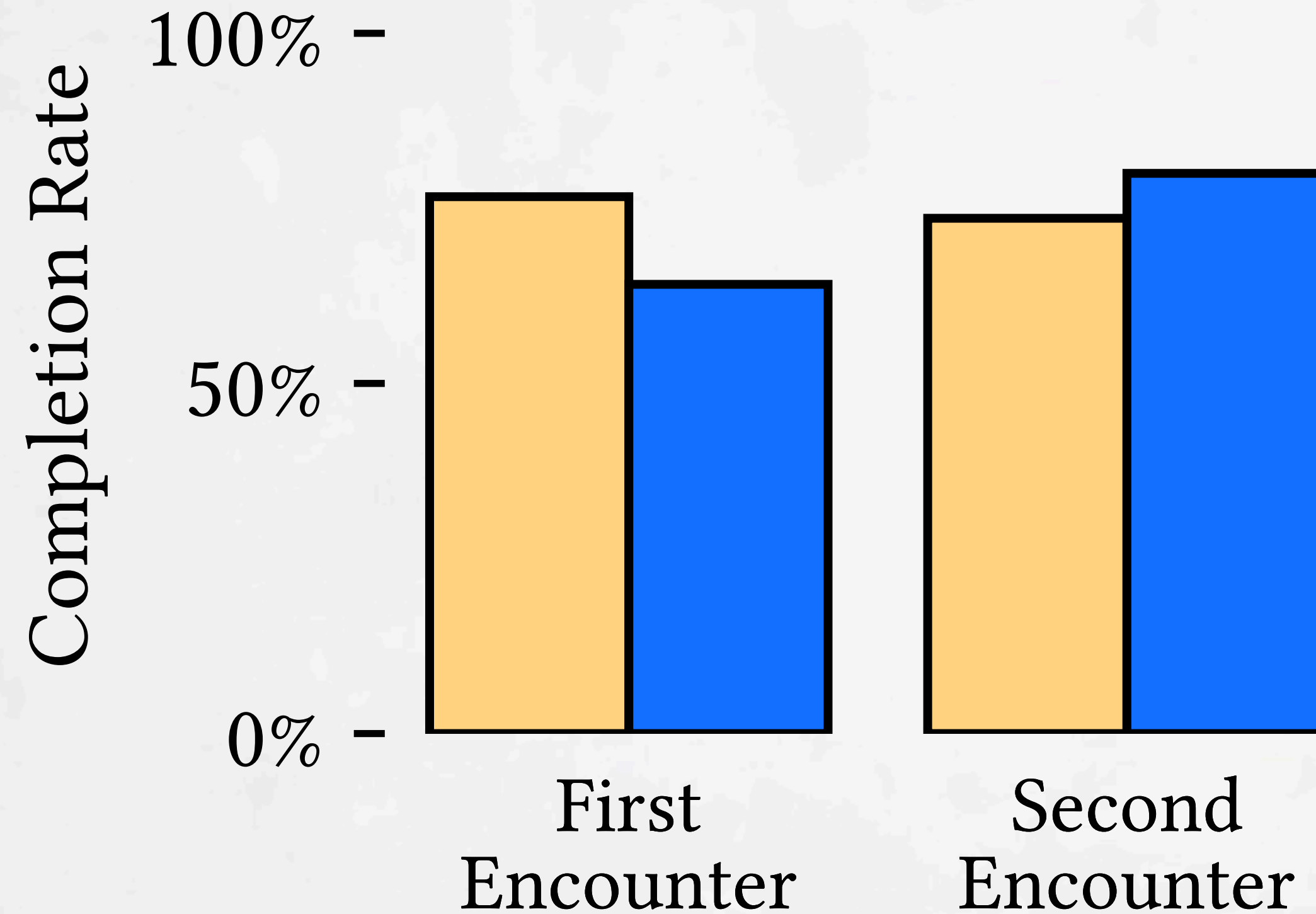
Deuce more effective than Traditional?



Deuce more effective than Traditional?



Deuce more effective than Traditional?



Deuce doesn't help discoverability

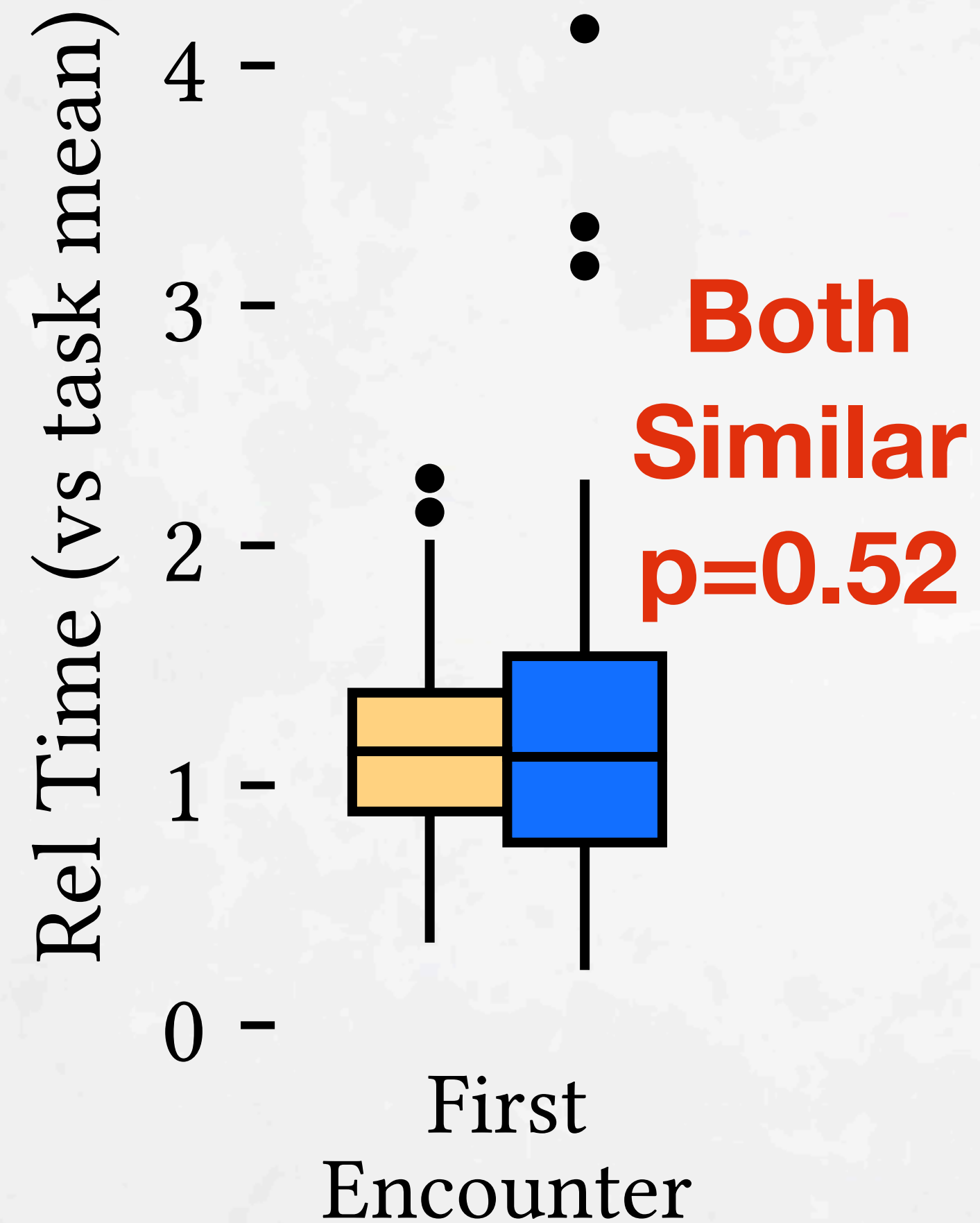
Deuce more effective than Traditional?

Rel Time (vs task mean)

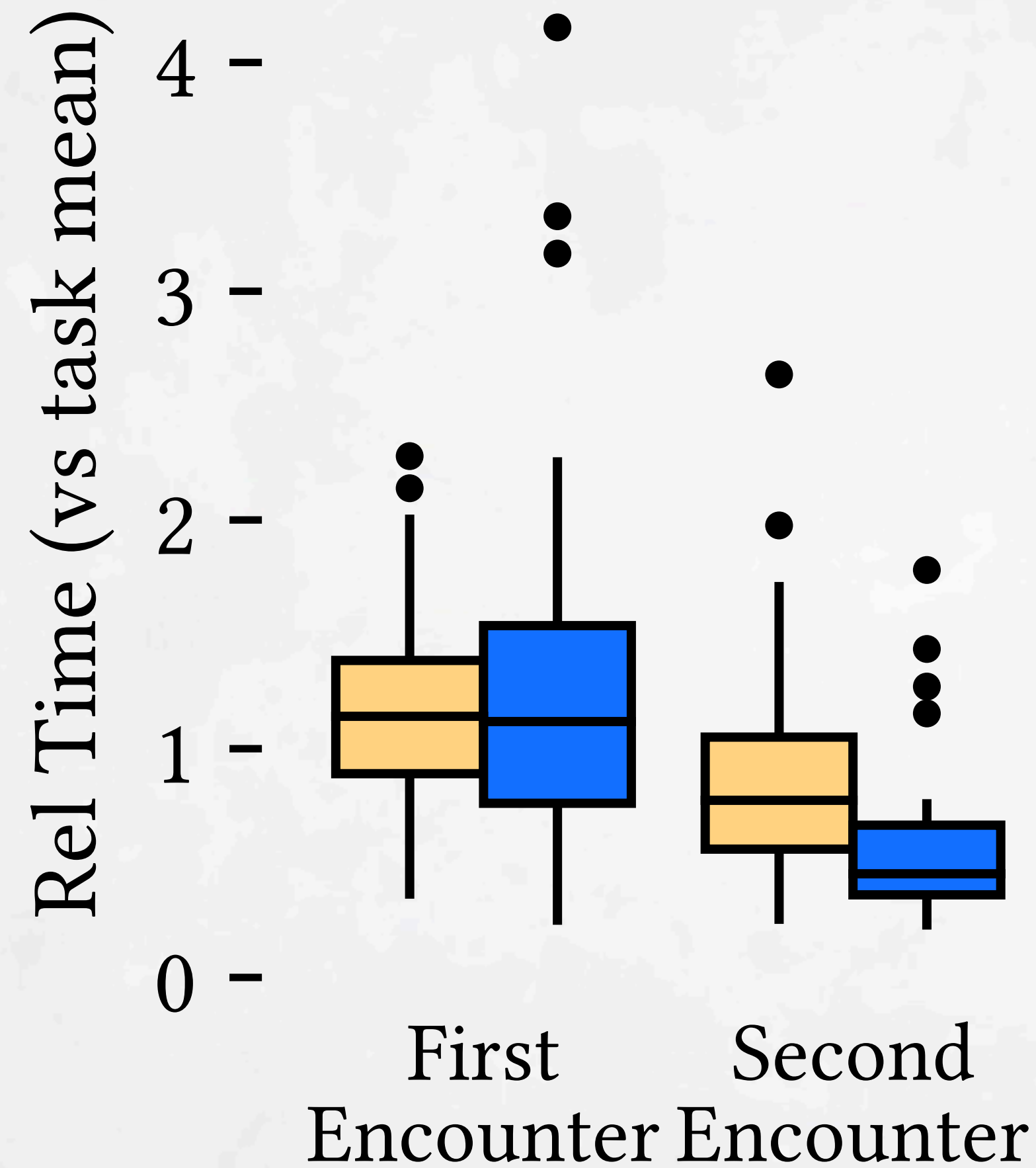


4 -
3 -
2 -
1 -
0 -

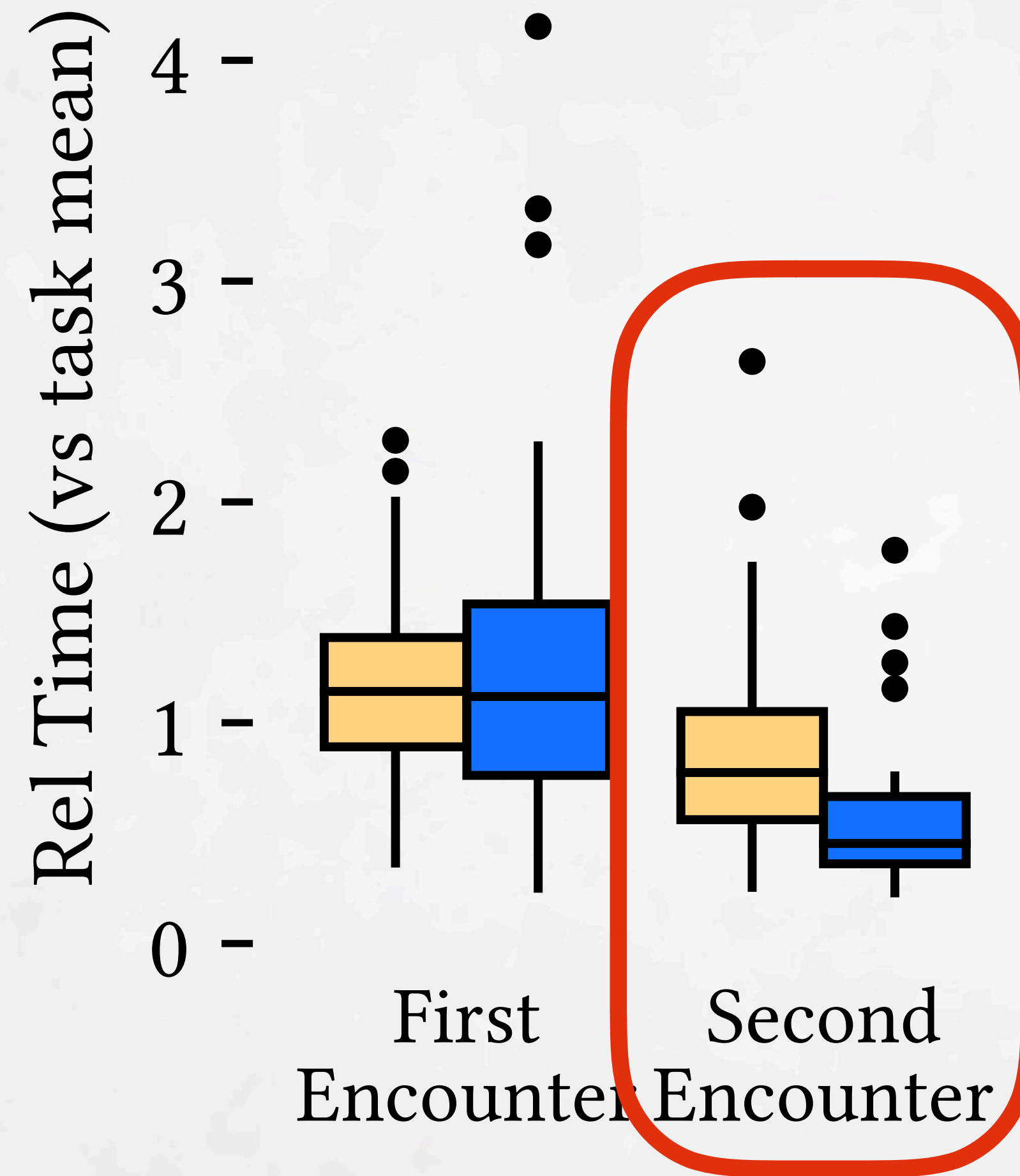
Deuce more effective than Traditional?



Deuce more effective than Traditional?

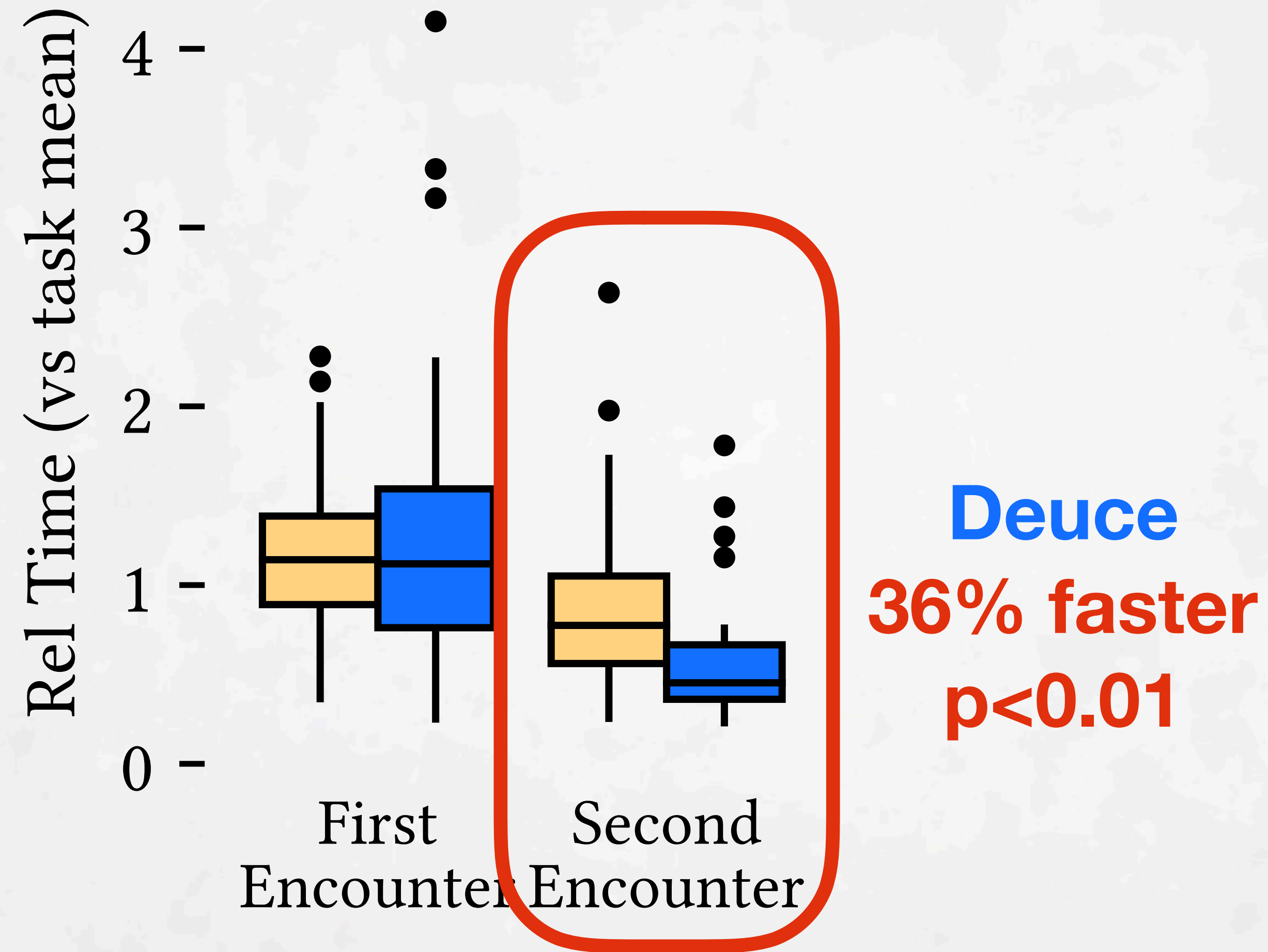


Deuce more effective than Traditional?



Deuce
36% faster
p<0.01

Deuce more effective than Traditional?



Deuce may be faster once learned

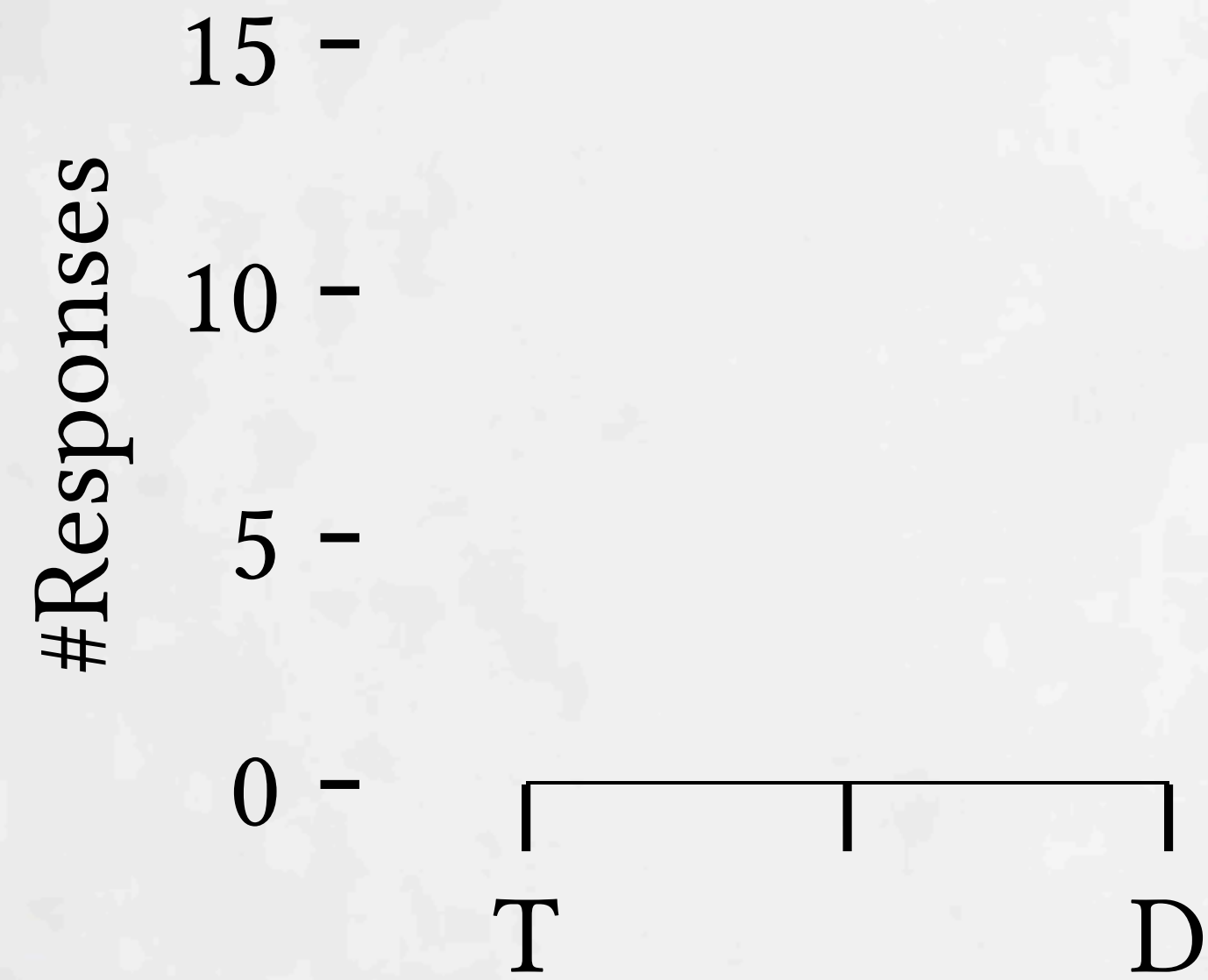
Deuce preferred to **Traditional**?

Deuce preferred to Traditional?

Survey

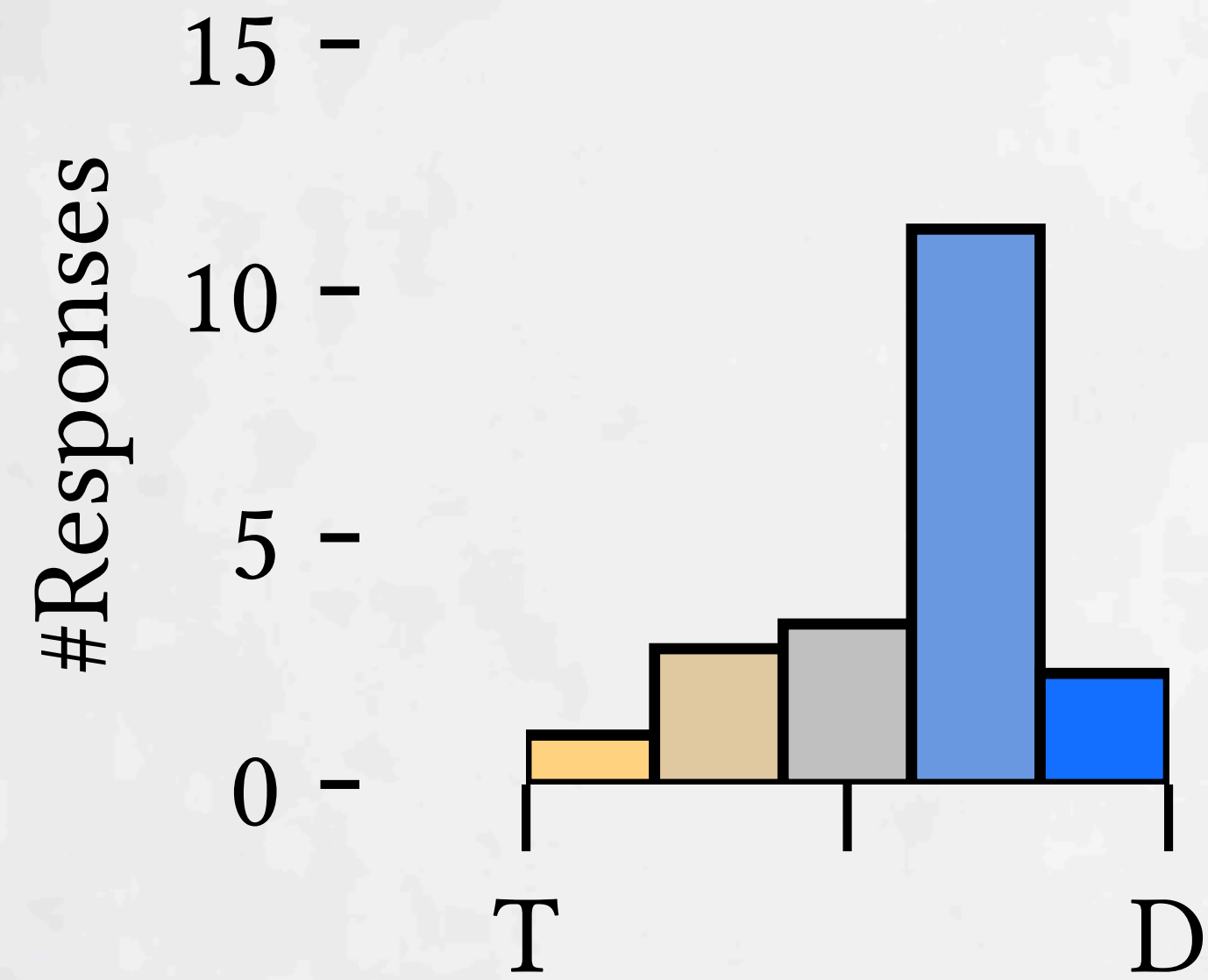
Deuce preferred to Traditional?

Survey



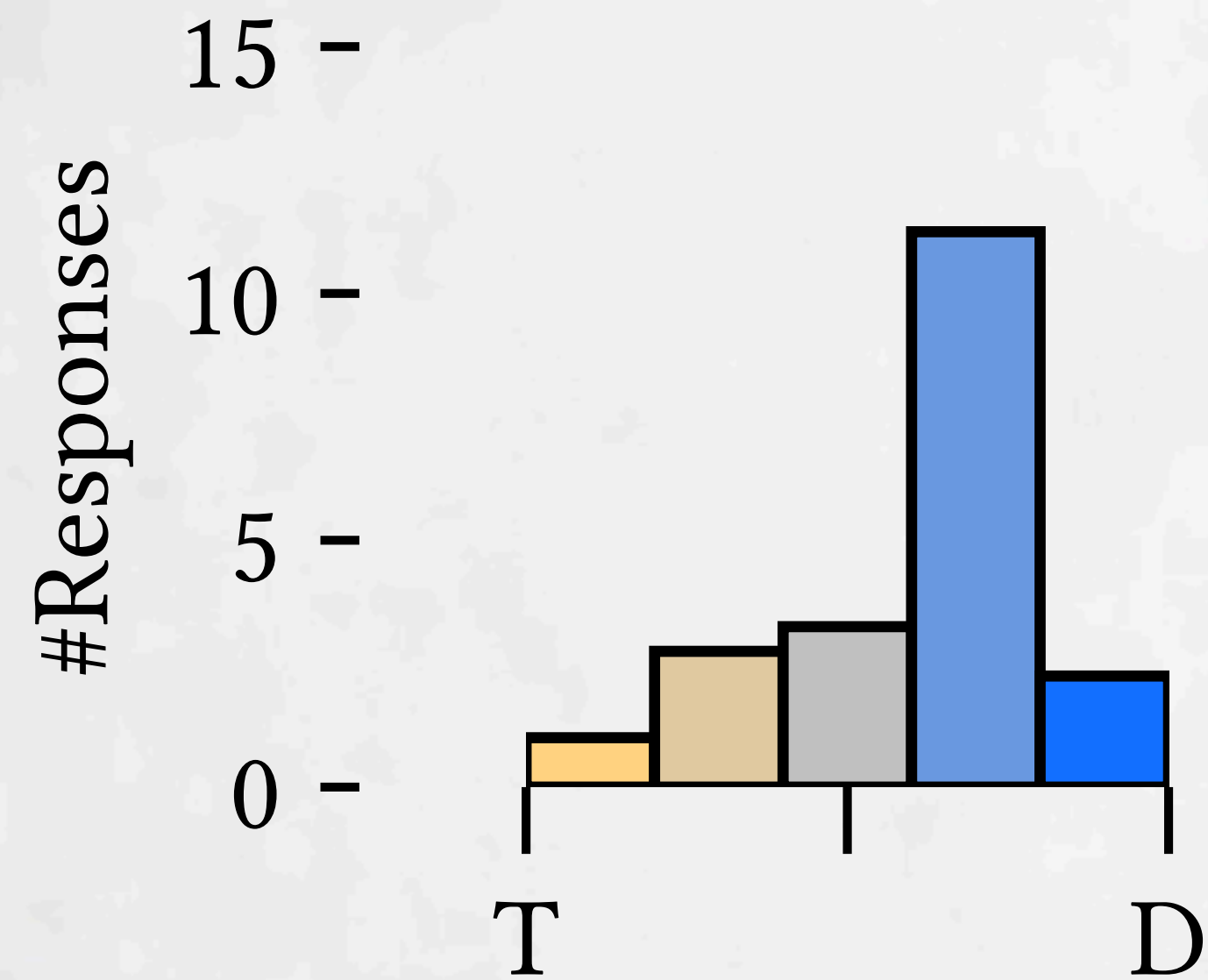
Deuce preferred to Traditional?

Survey



Deuce preferred to Traditional?

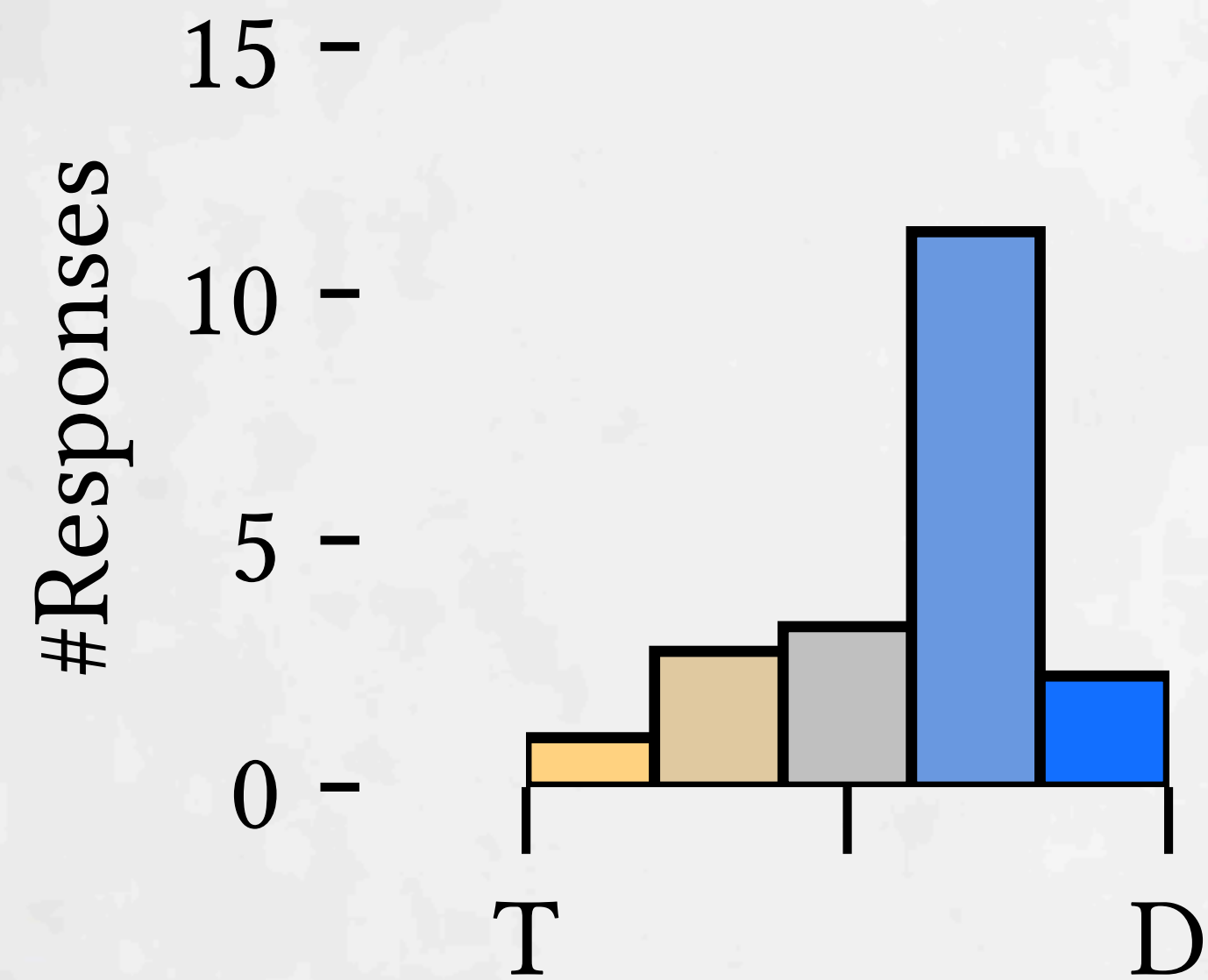
Survey



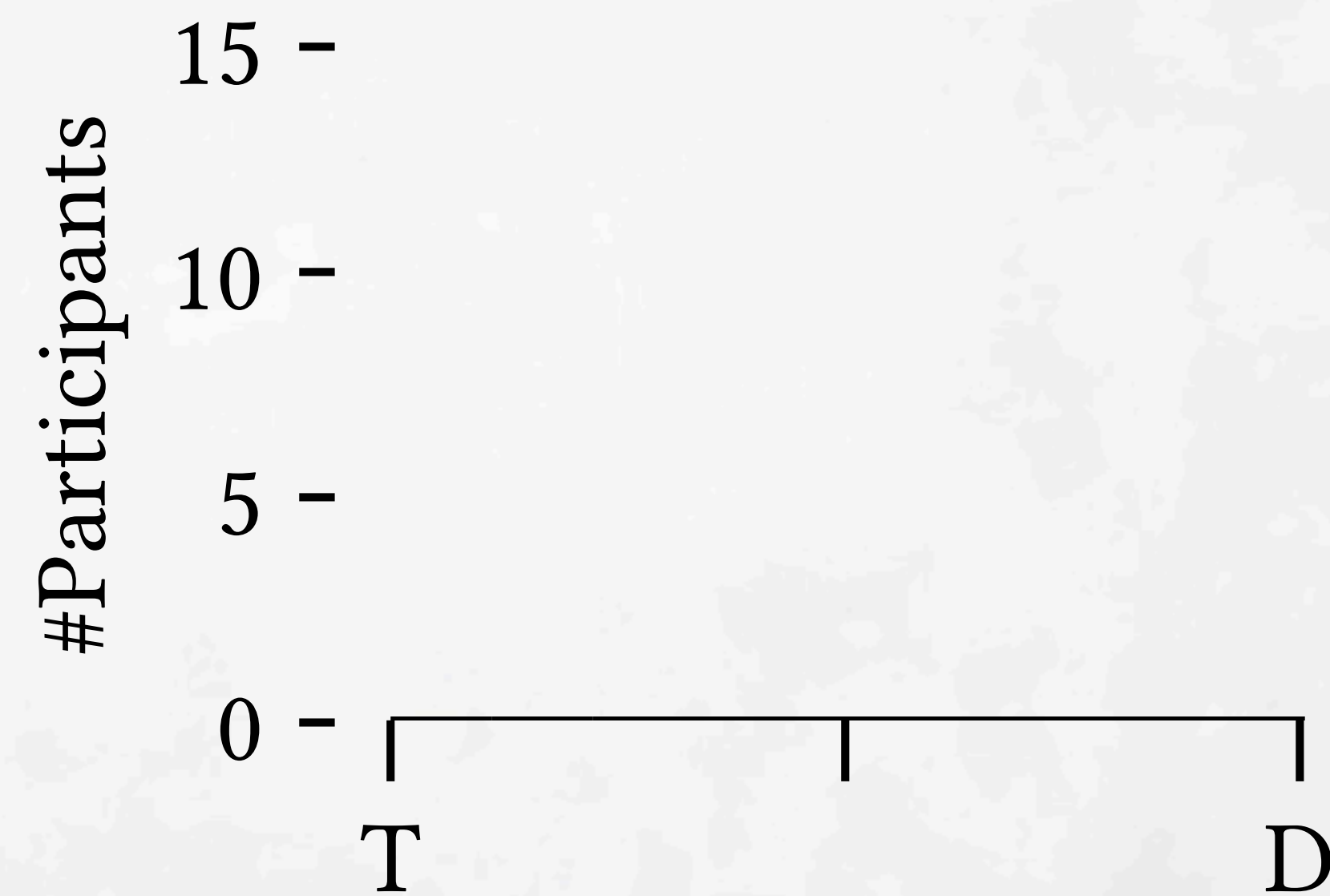
Modest subjective preference for Deuce

Deuce preferred to Traditional?

Survey



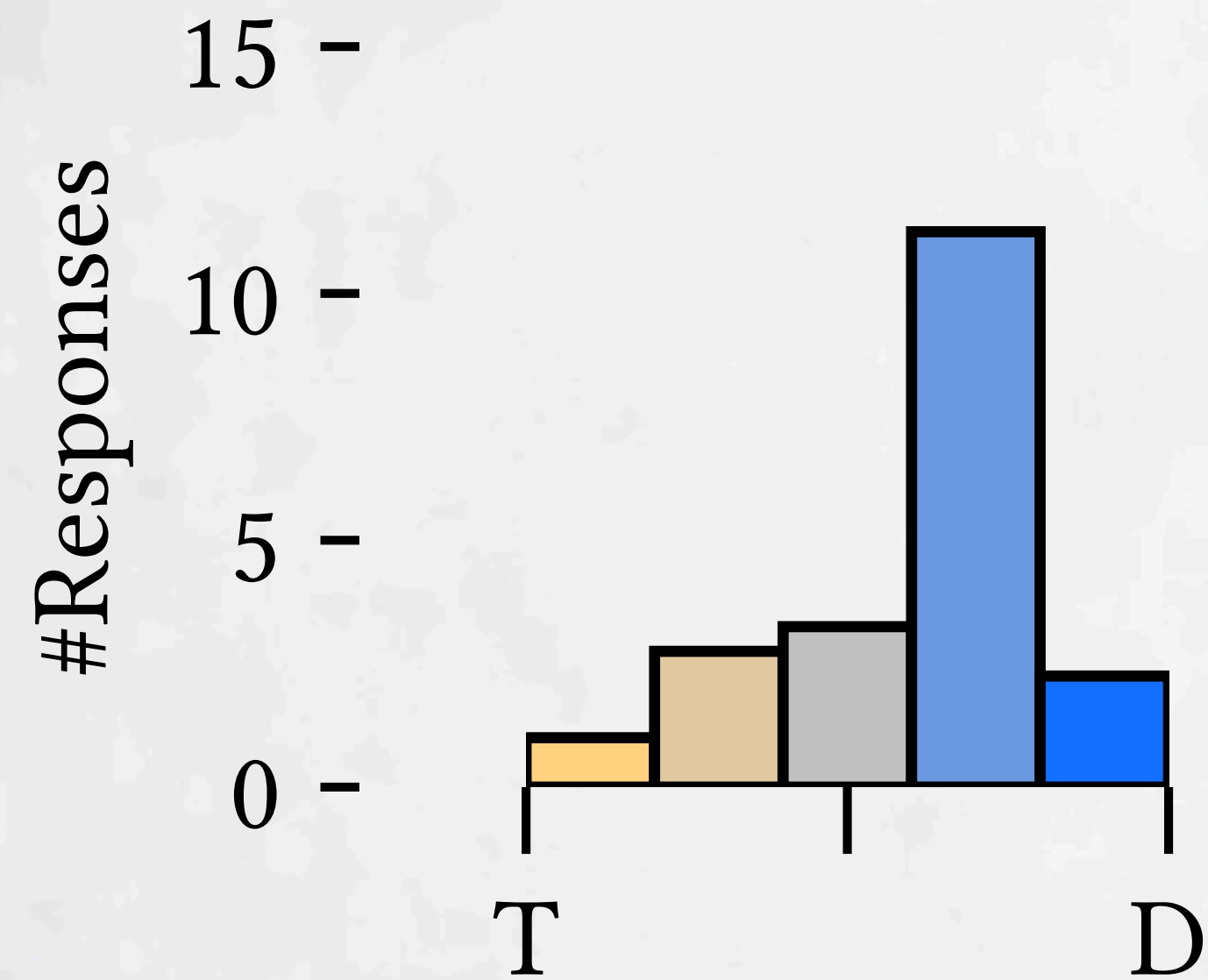
Observed



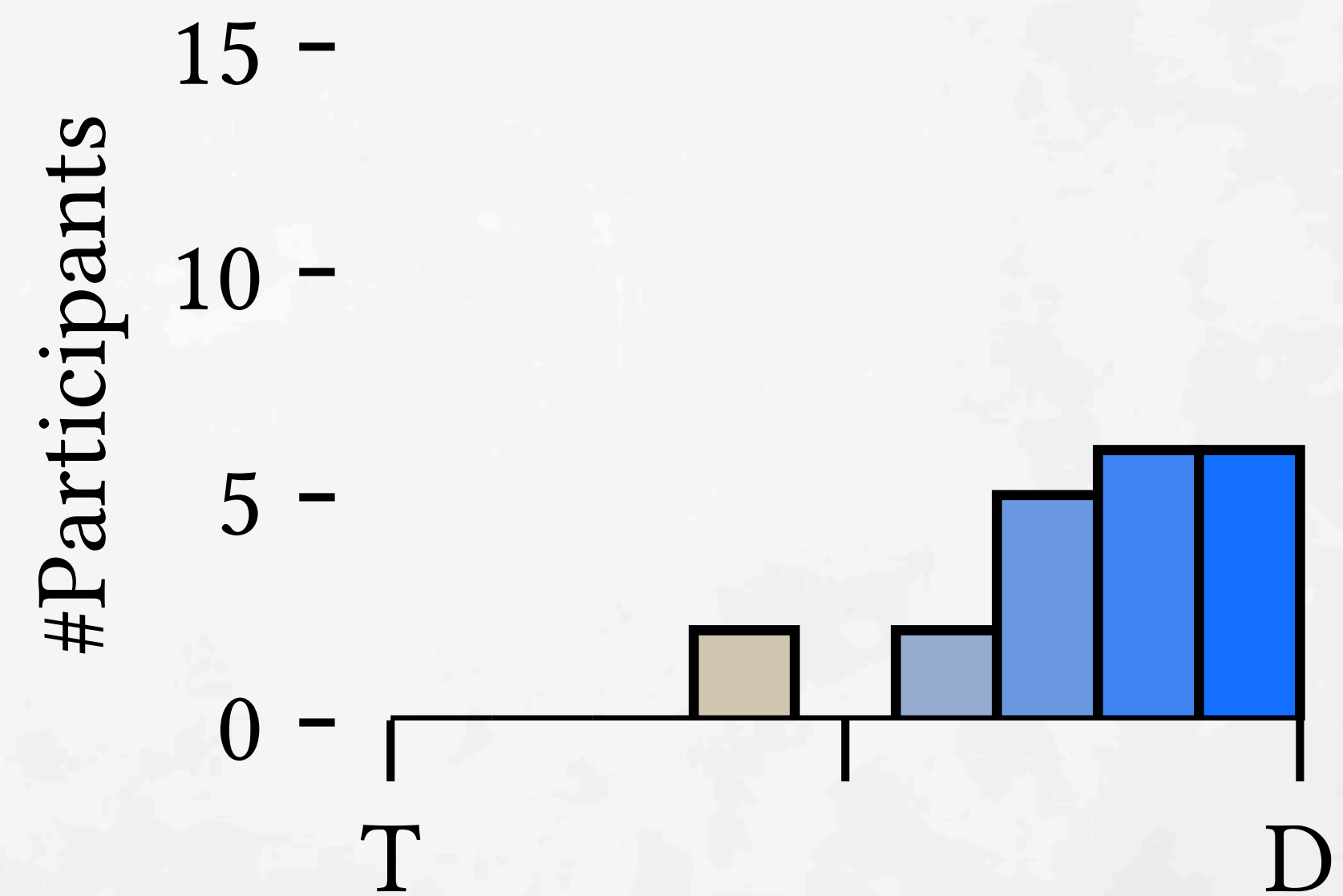
Modest subjective preference for Deuce

Deuce preferred to Traditional?

Survey



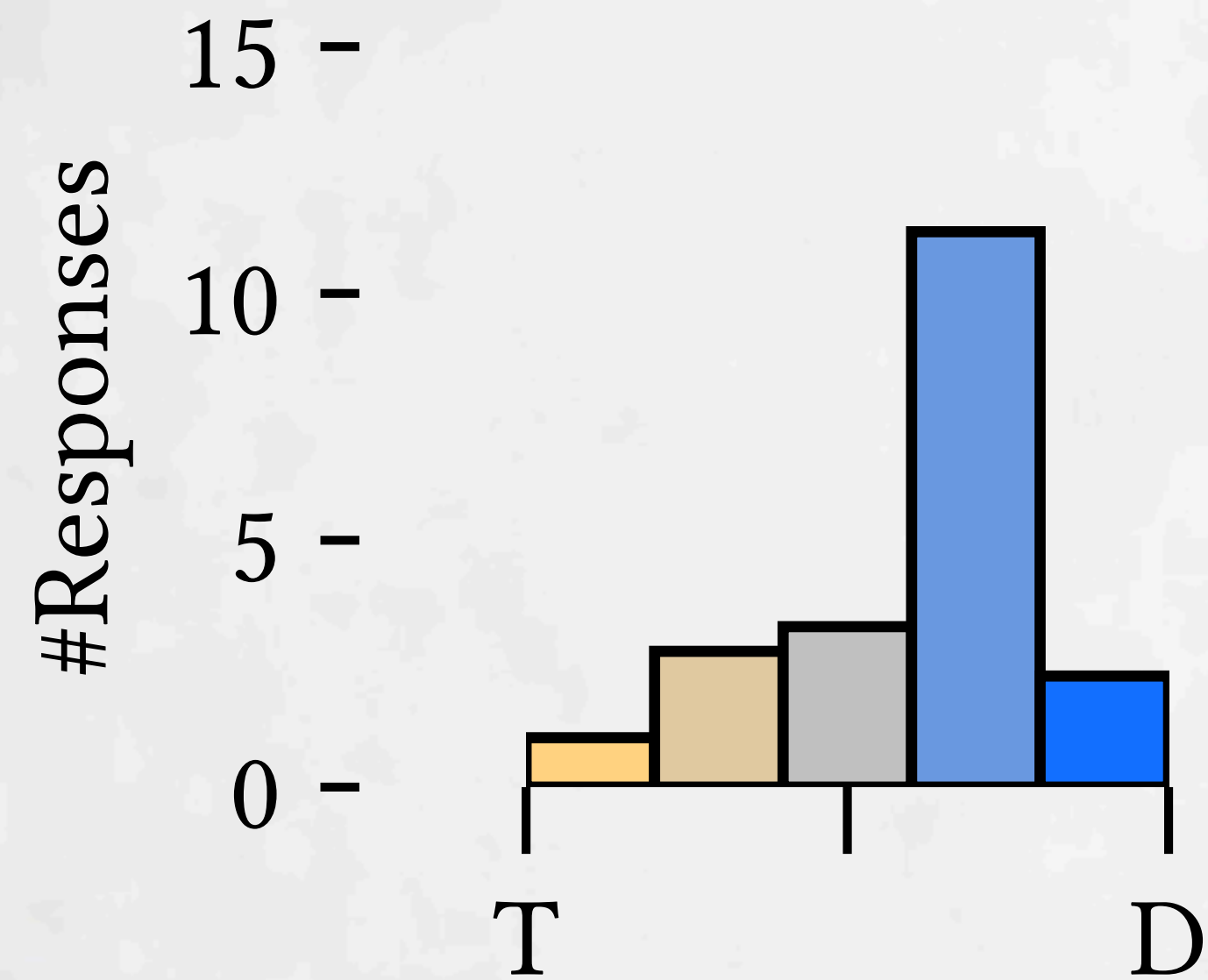
Observed



Modest subjective preference for Deuce

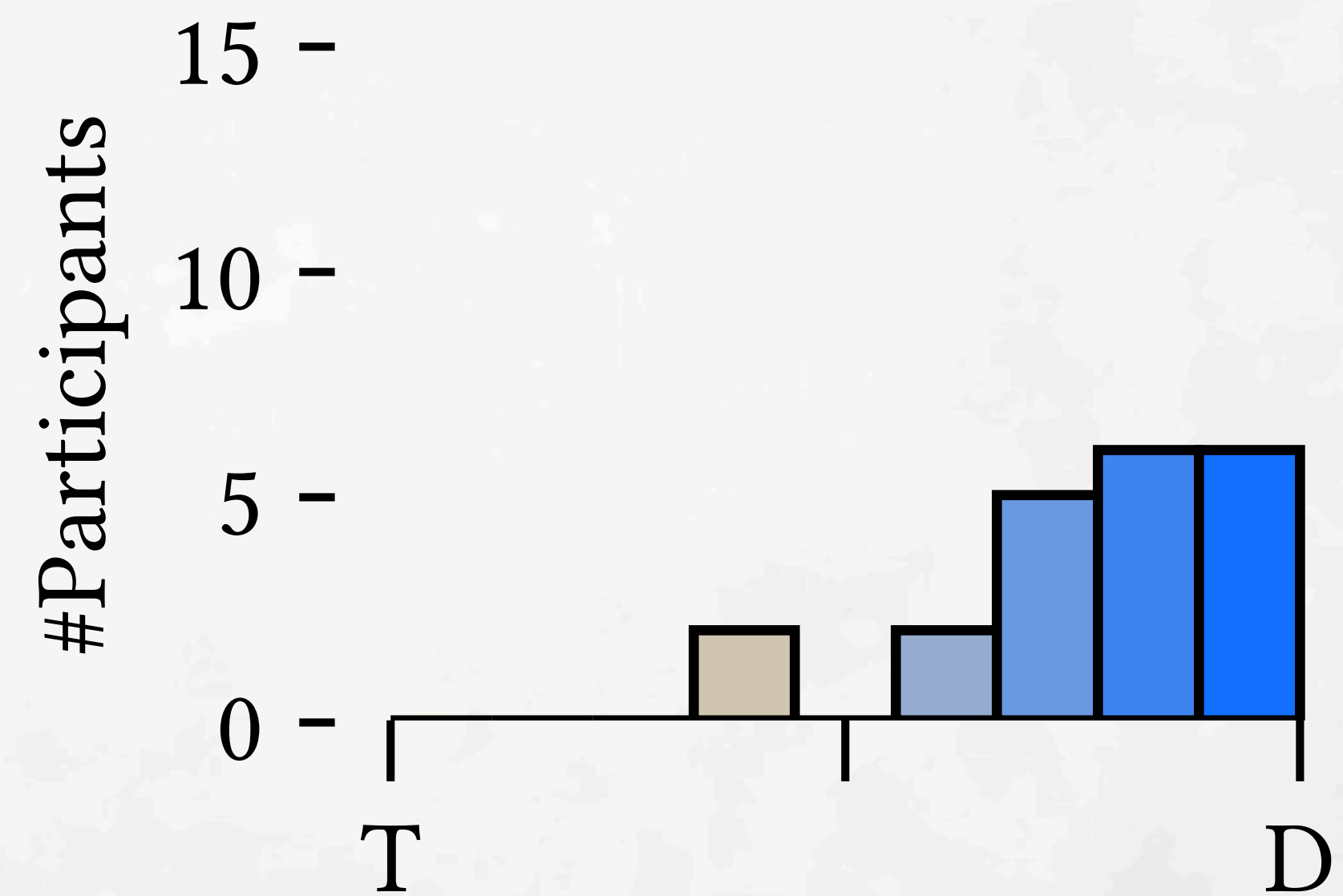
Deuce preferred to Traditional?

Survey



Modest subjective preference for Deuce

Observed



Almost everyone used Deuce more

Mix & Match Tool Usage

Mix & Match Tool Usage

Rename

Make Equal with Single Variable

Introduce Variable(s)

Add Argument(s)

Create Function from Arguments

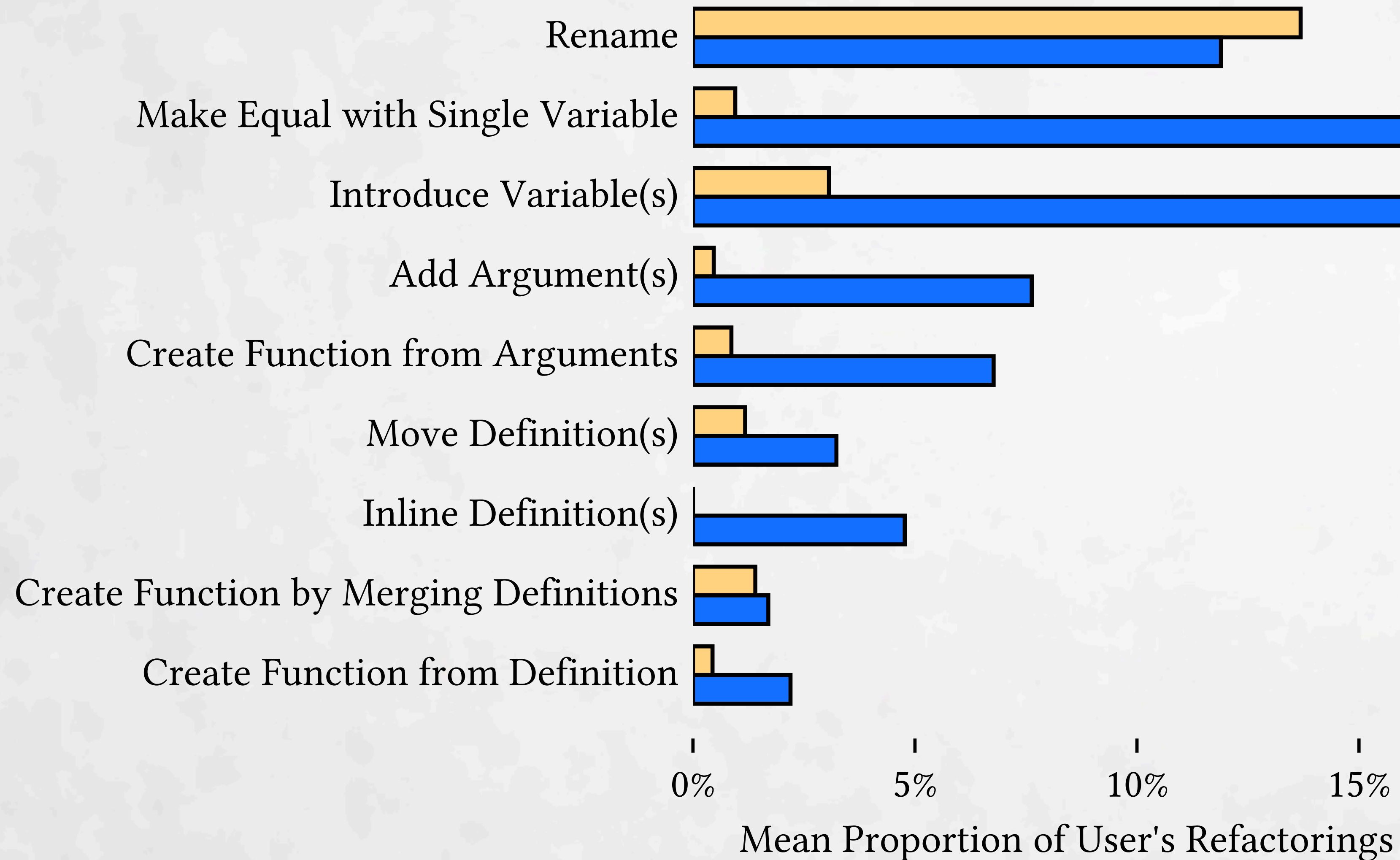
Move Definition(s)

Inline Definition(s)

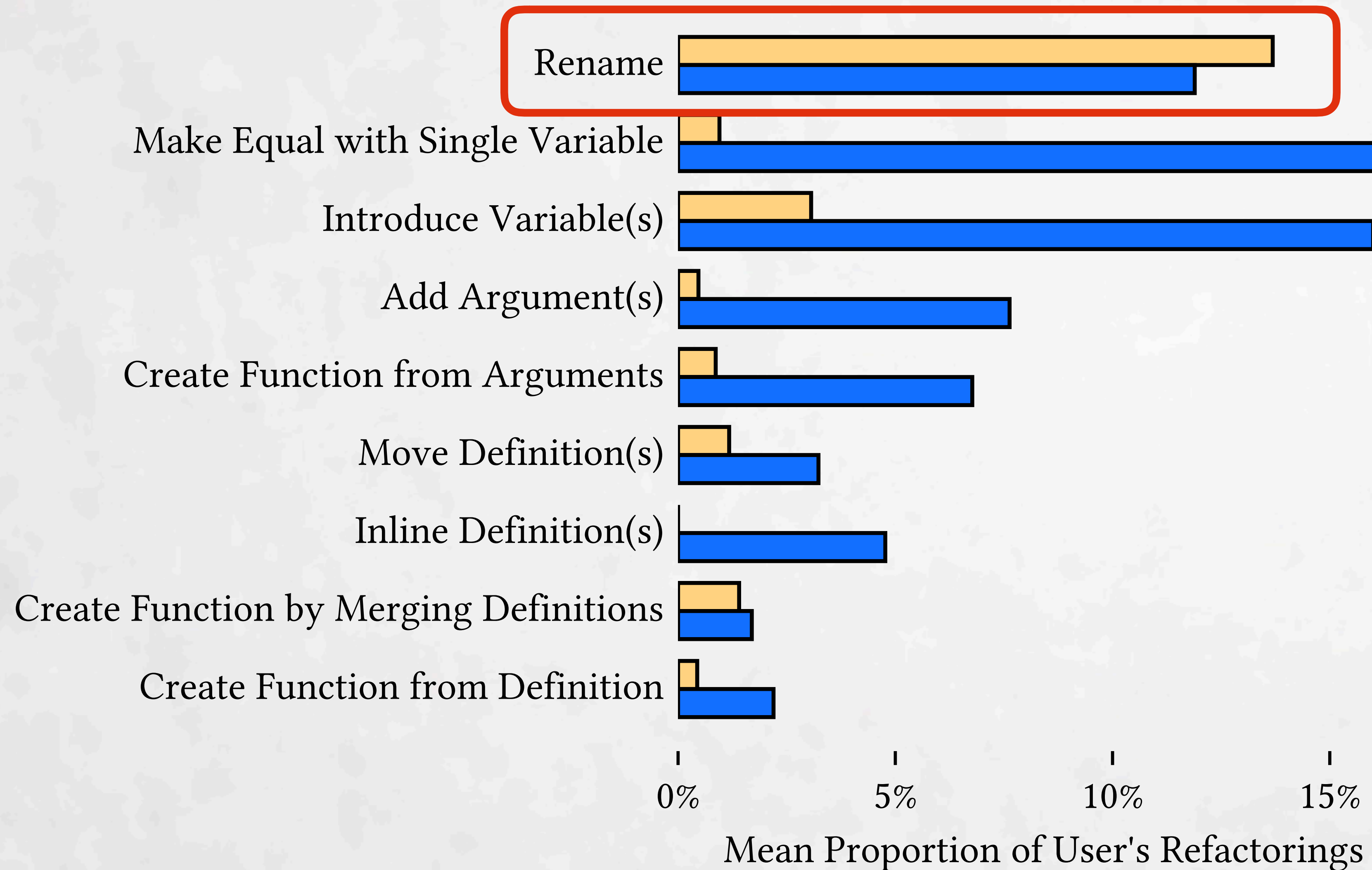
Create Function by Merging Definitions

Create Function from Definition

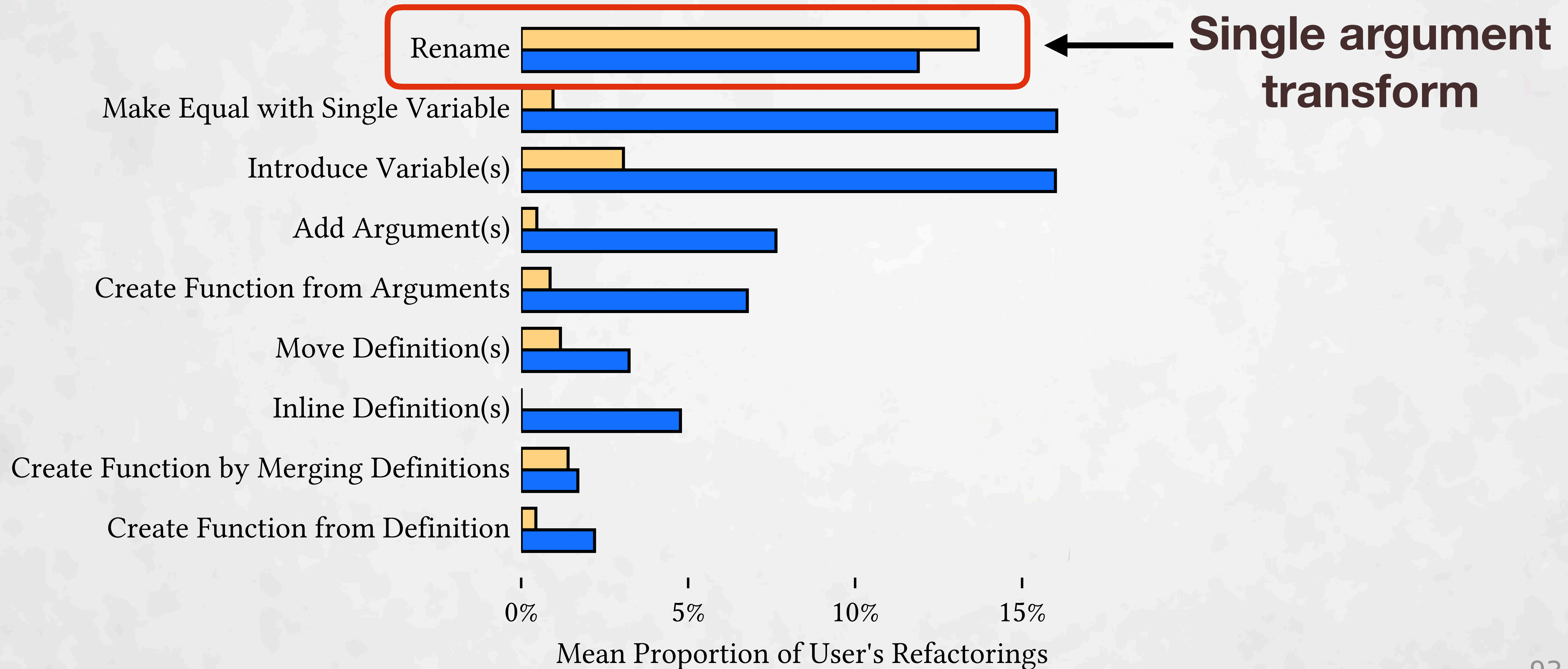
Mix & Match Tool Usage



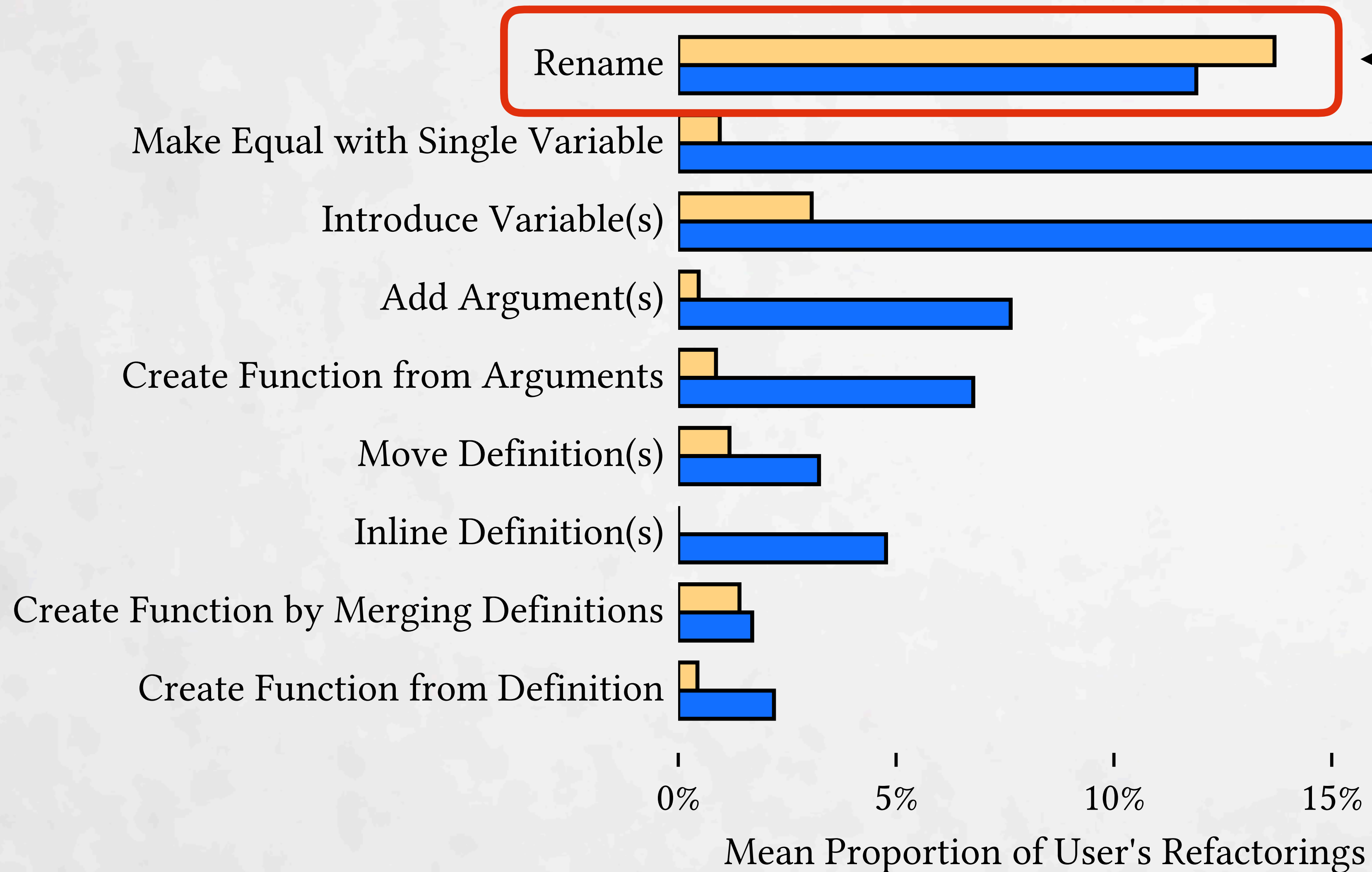
Mix & Match Tool Usage



Mix & Match Tool Usage



Mix & Match Tool Usage



← Single argument transform

Hypothesis:
Deuce better for multi-argument transformations

Deuce vs Traditional

Deuce vs Traditional

Traditional may be better for learning

Deuce vs Traditional

Traditional may be better for learning

Deuce may be faster once learned

Deuce vs Traditional

Traditional may be better for learning

Deuce may be faster once learned

Deuce strongly preferred

Future Work

Future Work

UI concerns for larger programs

Future Work

UI concerns for larger programs

How to encourage refactoring?

Future Work

UI concerns for larger programs

How to encourage refactoring?

DSL for defining new transformations

Future Work

UI concerns for larger programs

How to encourage refactoring?

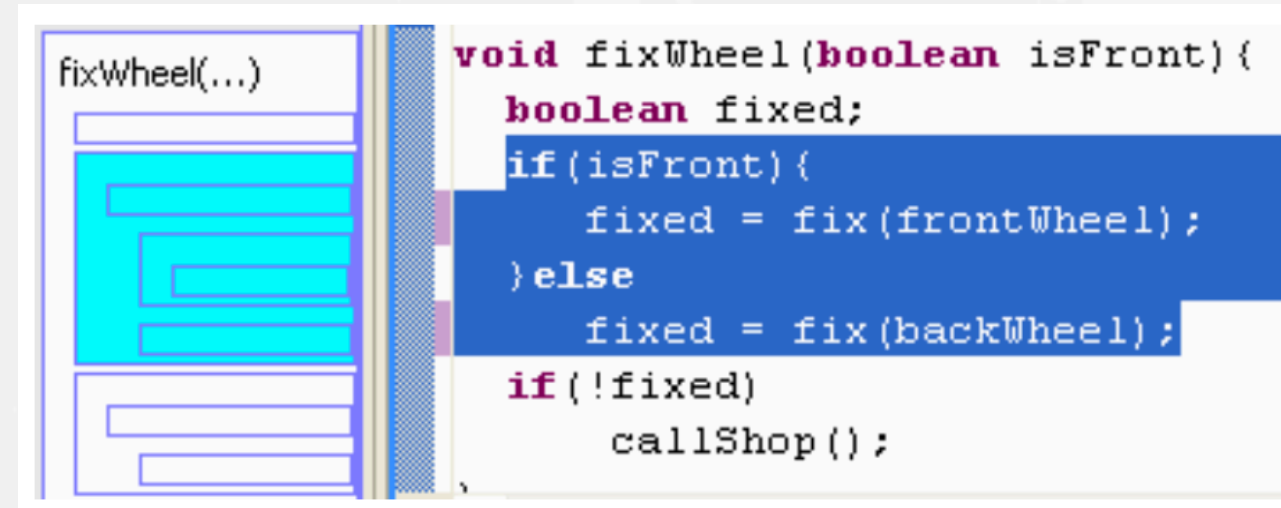
DSL for defining new transformations

Real languages in existing editors

Related Work

Related Work

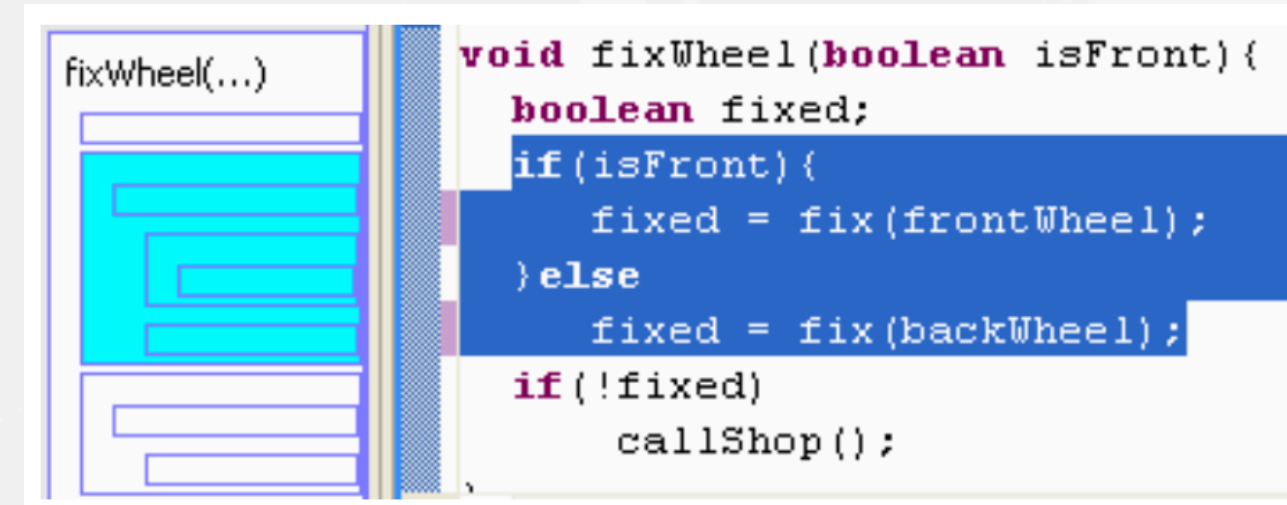
```
boolean isWellDressed(){  
    if(jersey.isSpandex()){  
        return shorts.isSpandex();  
    }  
    return true;  
}
```



Selection Assist + Box View
(Murphy-Hill and Black 2008)

Related Work

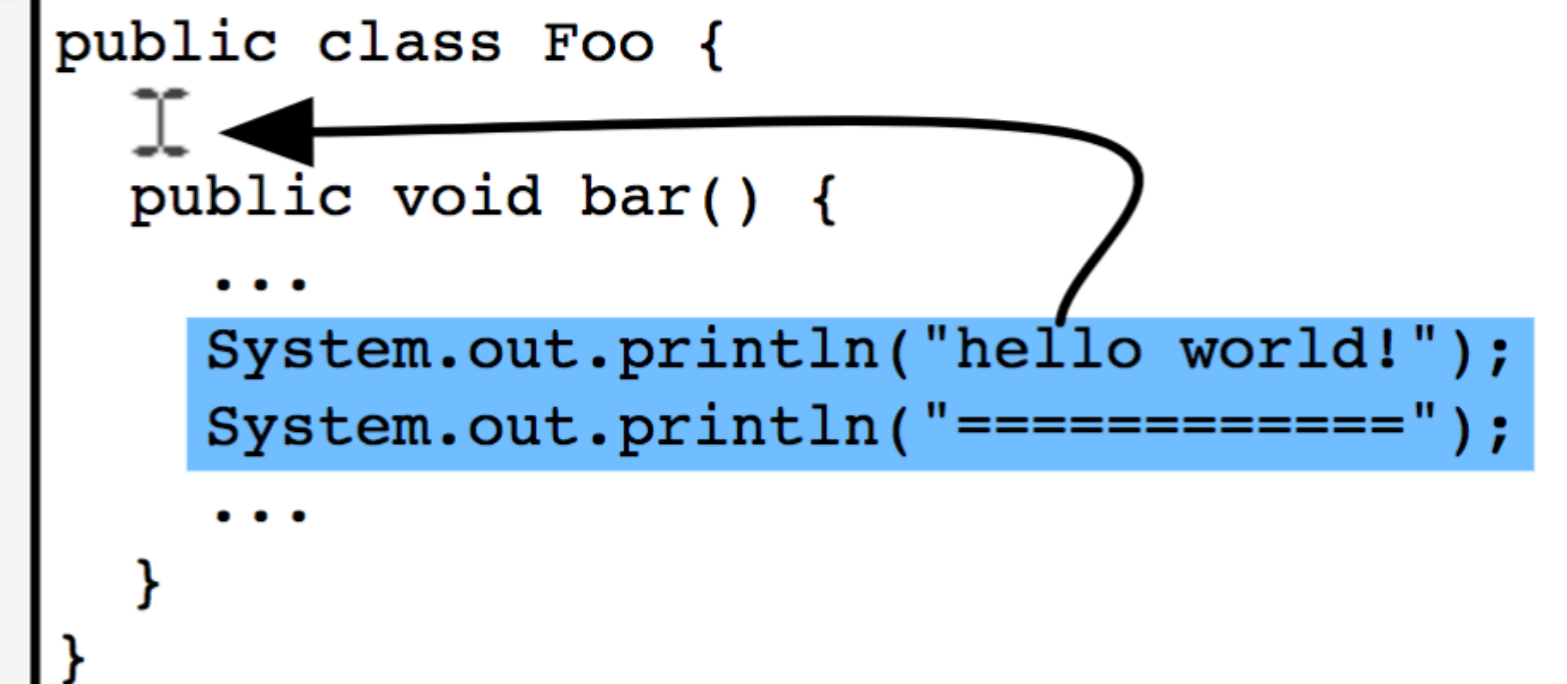
```
boolean isWellDressed(){  
    if(jersey.isSpandex()){  
        return shorts.isSpandex();  
    }  
    return true;  
}
```



The screenshot shows a code editor with a function call `fixWheel(...)` on the left. A box view is displayed, showing the function's signature `void fixWheel(boolean isFront)` and its implementation. The implementation includes a `boolean fixed` variable, an `if(isFront)` block, an `else` block, and an `if(!fixed)` block. The code is color-coded, and the box view highlights the function's body.

```
void fixWheel(boolean isFront){  
    boolean fixed;  
    if(isFront){  
        fixed = fix(frontWheel);  
    }else  
        fixed = fix(backWheel);  
    if(!fixed)  
        callShop();  
}
```

Selection Assist + Box View
(Murphy-Hill and Black 2008)



The screenshot shows a code editor with a class `public class Foo` containing a method `public void bar()`. A cursor is positioned at the start of the `bar()` method. A blue box highlights the code `System.out.println("hello world!");` and `System.out.println("=====");` within the method. A black arrow points from the cursor to the start of the highlighted code, indicating a drag-and-drop refactoring operation.

```
public class Foo {  
    public void bar() {  
        ...  
        System.out.println("hello world!");  
        System.out.println("=====");  
        ...  
    }  
}
```

DNDRefactoring
(Lee et al. 2013)

Related Work

```
boolean isWellDressed(){  
    if(jersey.isSpandex()){  
        return shorts.isSpandex();  
    }  
    return true;  
}
```

```
fixWheel(...)  
void fixWheel(boolean isFront){  
    boolean fixed;  
    if(isFront){  
        fixed = fix(frontWheel);  
    }else  
        fixed = fix(backWheel);  
    if(!fixed)  
        callShop();  
}
```

```
public class Foo {  
    public void bar() {  
        ...  
        System.out.println("hello world!");  
        System.out.println("=====");  
        ...  
    }  
}
```

Selection Assist + Box View
(Murphy-Hill and Black 2008)

DNDRefactoring
(Lee et al. 2013)

```
swapped = false  
for each(var int i : 1 .. n-1)  
    if (vals[i] < vals[i-1])  
        var int t = vals[i]  
        vals[i] = vals[i-1]  
        vals[i-1] = t  
        swapped = true
```

Greenfoot
(Brown et al. 2016)

```
public class Distance  
{  
    Computes the distance between two points.  
    public static final double main(double x1, double y1, double x2, double y2)  
    {  
        return  $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$ ;  
    }  
}
```

Barista
(Ko and Myers 2006)

```
public Color getDefaultColor() {  
    return  
    0,  
    0,  
    128); // navy  
}
```

navy
R:0 G:0 B:128

Graphite
(Omar et al. 2012)

Structure Select

```
(def image_url
  "img/icse-2018-large-icon-small.png")

(def image1
  (let [width height [324 200]
        x y [100 100]]
    (image "lightgrey" x y width height 15 image_url)))

(def main
  (draw (concat [ image1 ])))
```


Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

Short Menu

Code Tools

- Create Function from Definition ▶
- Inline Definition ▶
- Make Single Line ▶

Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

Short Menu

Code Tools

- Create Function from Definition ▶ Abstract image1 over its constants
- Inline Definition ▶ Abstract image1 over its named constants
- Make Single Line ▶

Defaults

Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

Short Menu

| Code Tools | |
|---------------------------------|--------------------------------------|
| Create Function from Definition | ▶ Abstract image1 over its constants |
| Inline Definition | ▶ Abstract image1 over its named |
| Make Single Line | ▶ constants |

Defaults

Deuce provides *streamlined structural editing* in a *familiar text-based environment*.

Structure Select

```
(def image_url
  "img/icse-2018-large-icon-small.png")

(def image1
  (let [width height [324 200]
        x y [100 100]
        (image "lightgrey" x y width height 15 image_url)]))

(def main
  (draw (concat [ image1 ])))
```

Short Menu

| Code Tools | |
|---------------------------------|--------------------------------------|
| Create Function from Definition | ▶ Abstract image1 over its constants |
| Inline Definition | ▶ Abstract image1 over its named |
| Make Single Line | ▶ constants |

Defaults

Search “sketch n sketch” to play with **Deuce**

Thank you!

Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

Short Menu

| Code Tools | |
|---------------------------------|--------------------------------------|
| Create Function from Definition | ▶ Abstract image1 over its constants |
| Inline Definition | ▶ Abstract image1 over its named |
| Make Single Line | ▶ constants |

Defaults

Search “sketch n sketch” to play with **Deuce**

Extra Slides

$$\begin{array}{l}
\textit{program} ::= \bullet \boxed{(\text{def } x_0 \ e_0)} \bullet \cdots \bullet \boxed{(\text{def main } e)} \\
e ::= c \mid x \mid (\lambda \mathbf{p} \ e) \mid (e_1 \ e_2) \mid [e_1 \mid e_2] \\
\quad \mid (\boxed{\text{let } \mathbf{p} \ e_1} \ e_2) \mid (\text{case } e \ \boxed{(\mathbf{p}_1 \ e_1)} \cdots) \\
p ::= c \mid x \mid [] \mid [\mathbf{p}_1 \mid \mathbf{p}_2] \\
\text{Expressions } \mathbf{e} ::= \bullet \boxed{e} \bullet \quad \text{Patterns } \mathbf{p} ::= \bullet \boxed{p} \bullet
\end{array}$$

Figure 1: Syntax of LITTLE. The orange boxes and blue dots identify features for structural selection.


```
EditorState = { code: Program, selections: Set Selection }
ActiveState = Active | NotYetActive | Inactive
Options     = NoOptions | StringOption String
Result      = { description: String, code: Program }

CodeTool =
  { name : String
  , requirements : String
  , active : EditorState -> ActiveState
  , run : (EditorState, Options) -> List Result }
```

Figure 2: Code tool interface.

```
1 | _____|
2 | (def redSquare
3 |   (let [x y] [50 70])
4 |   (rect "salmon" x y 120 80)))
- |
```

```
1 |
2 | (def redSquare
3 |   (let [x y] [50 70])
4 |   (rect "salmon" x y 120 80)))
- |
```

```
1 |
2 | (def redSquare
3 |   (let [x y] [50 70])
4 |   (rect "salmon" x y 120 80)))
- |
```

Figure 3: Example target positions.

Code Tools

Create Function from Definition...

Create Function from Arguments...

Create Function by Merging Definitions...

Add Argument...

Remove Argument...

Reorder Arguments...

Rename Variable...

Introduce Local Variable...

Swap Variable Names and Usages...

Swap Variable Usages...

Make Equal with Single Variable...

Make Equal by Copying...

Move Definition...

Swap Definitions...

Inline Definition...

Duplicate Definition...

Reorder Expressions...

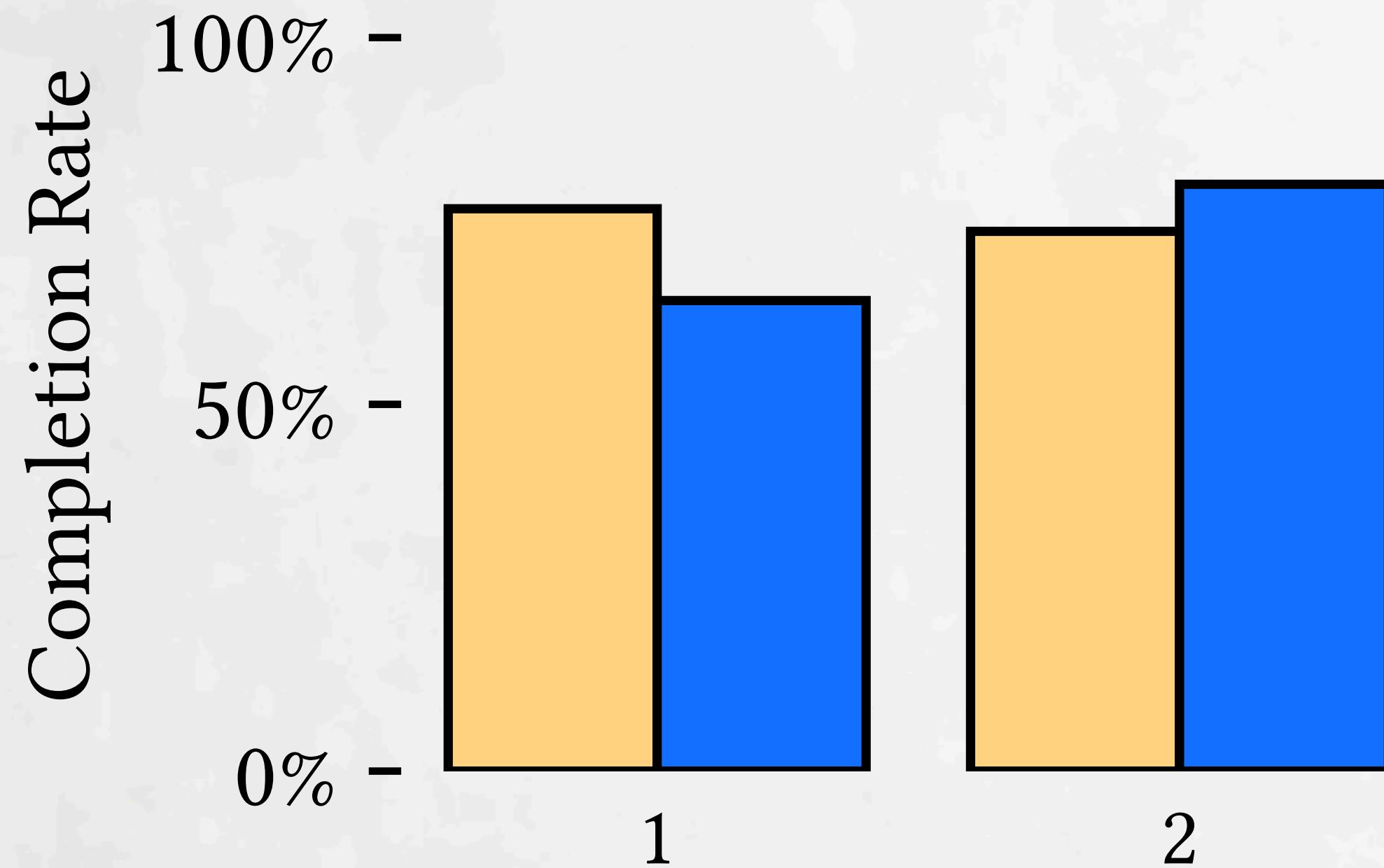
Swap Expressions...

Make Single Line...

Make Multi-line...

Align Expressions...

Head-to-Head Tasks

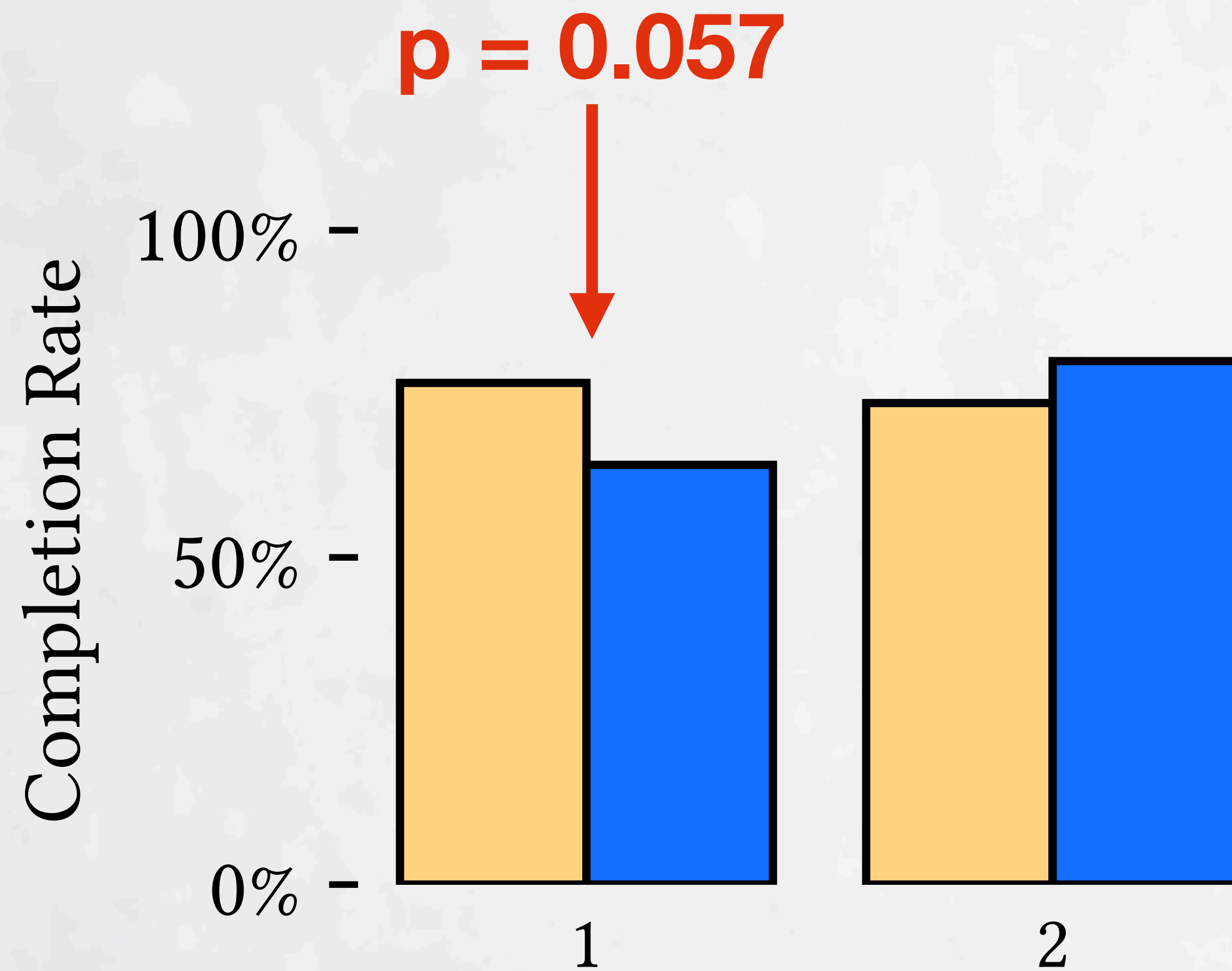


Mixed effects model to “control” for:

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

 Traditional
 Deuce

Head-to-Head Tasks

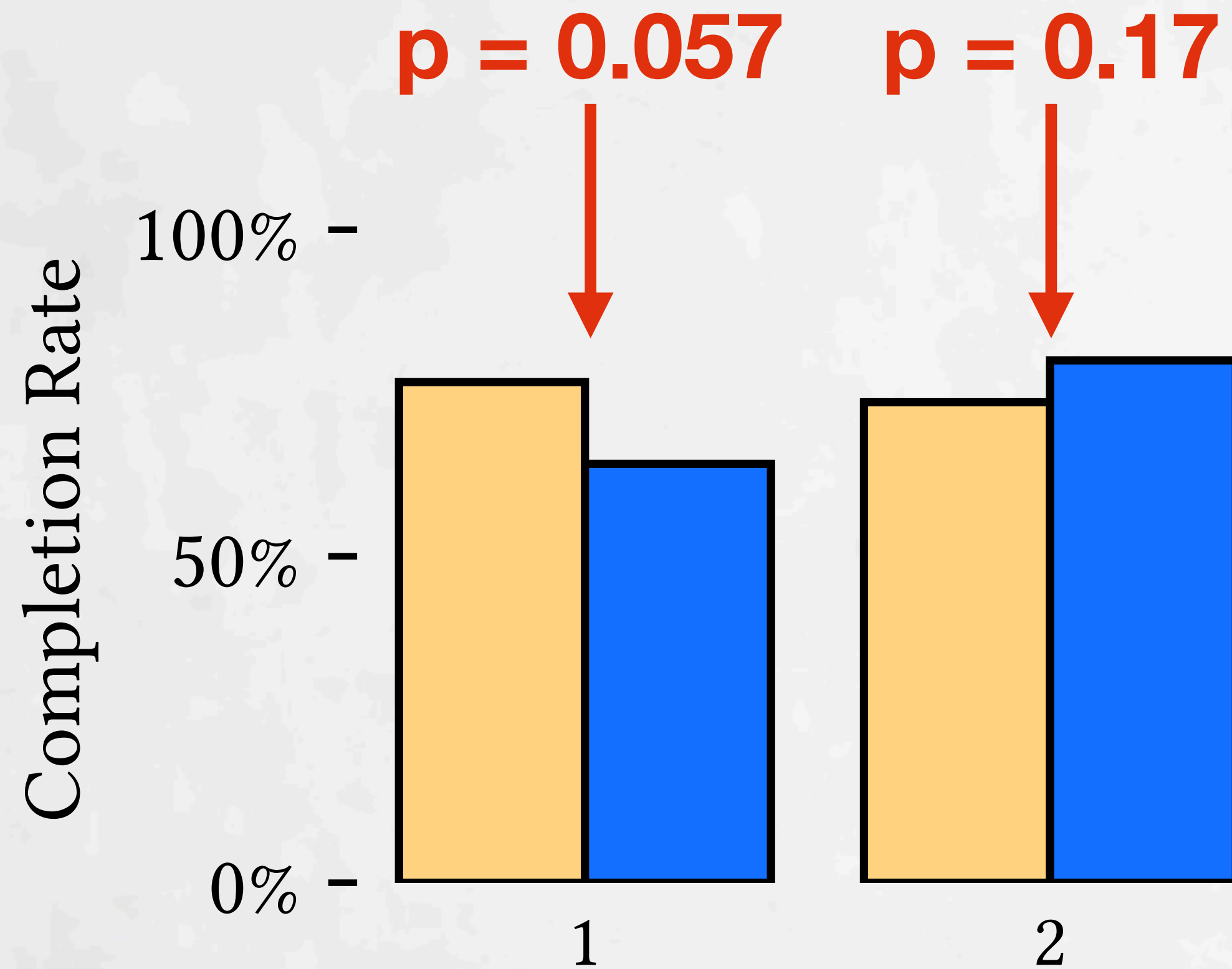


Mixed effects model to “control” for:

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

 Traditional
 Deuce

Head-to-Head Tasks



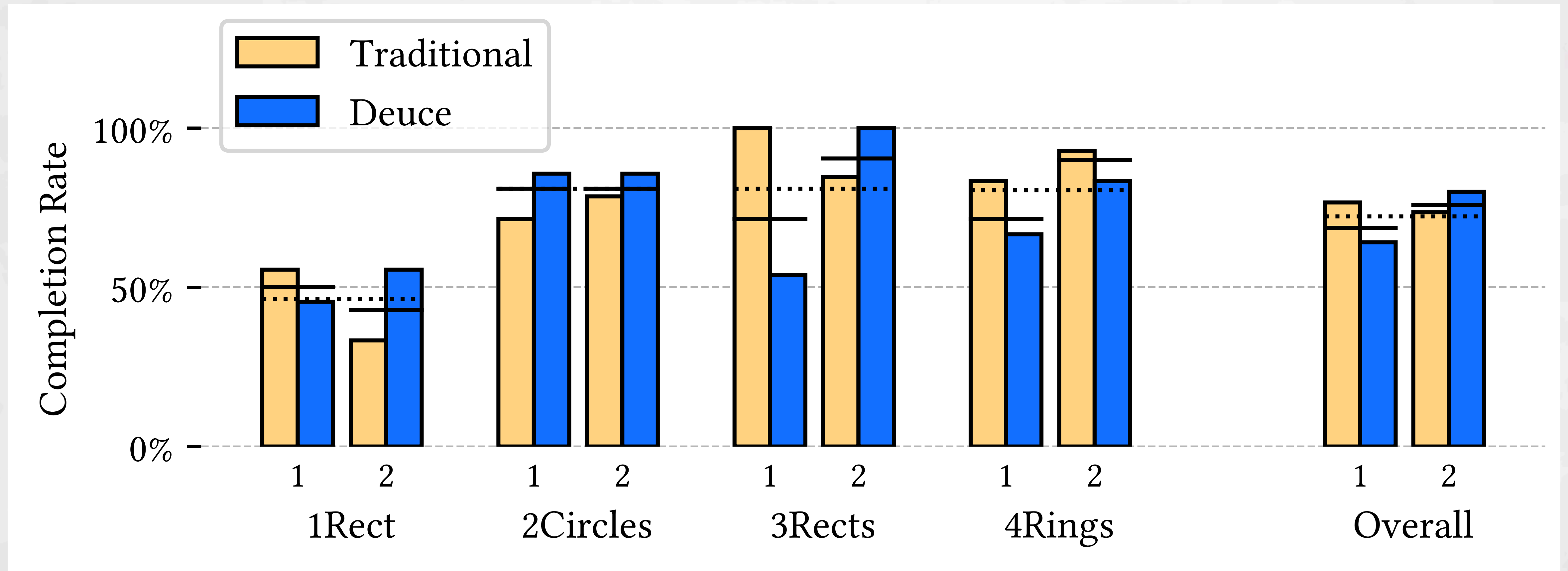
Traditional

Deuce

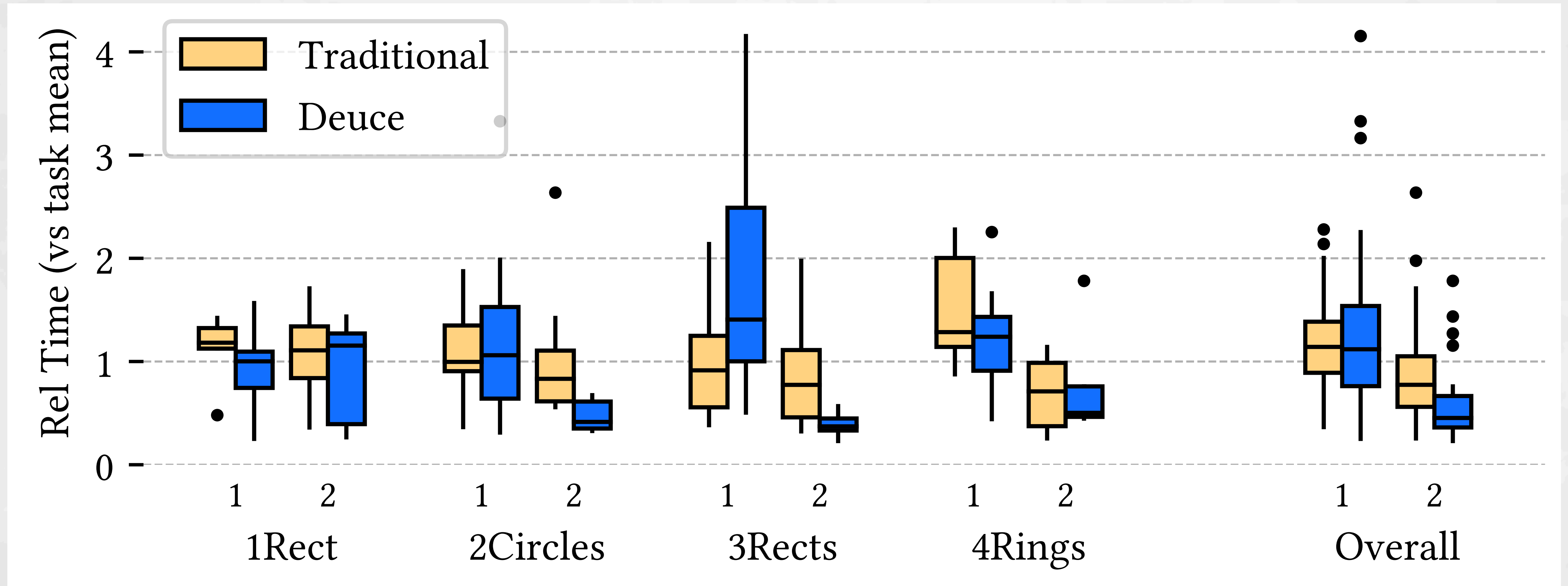
Mixed effects model to “control” for:

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

Head-to-Head Tasks

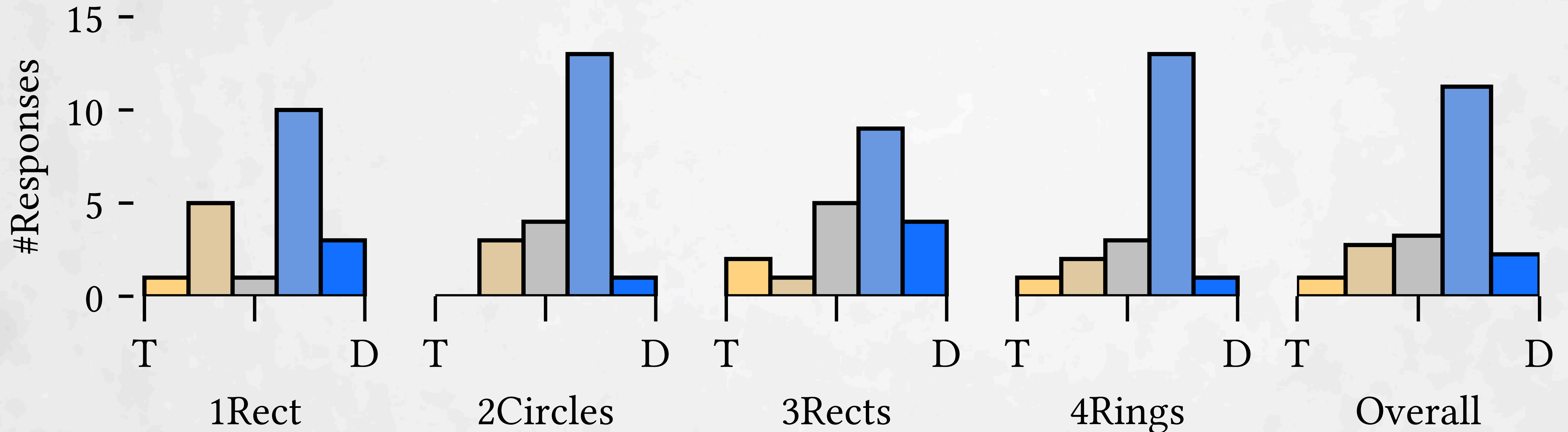


Head-to-Head Tasks



Head-to-Head Tasks

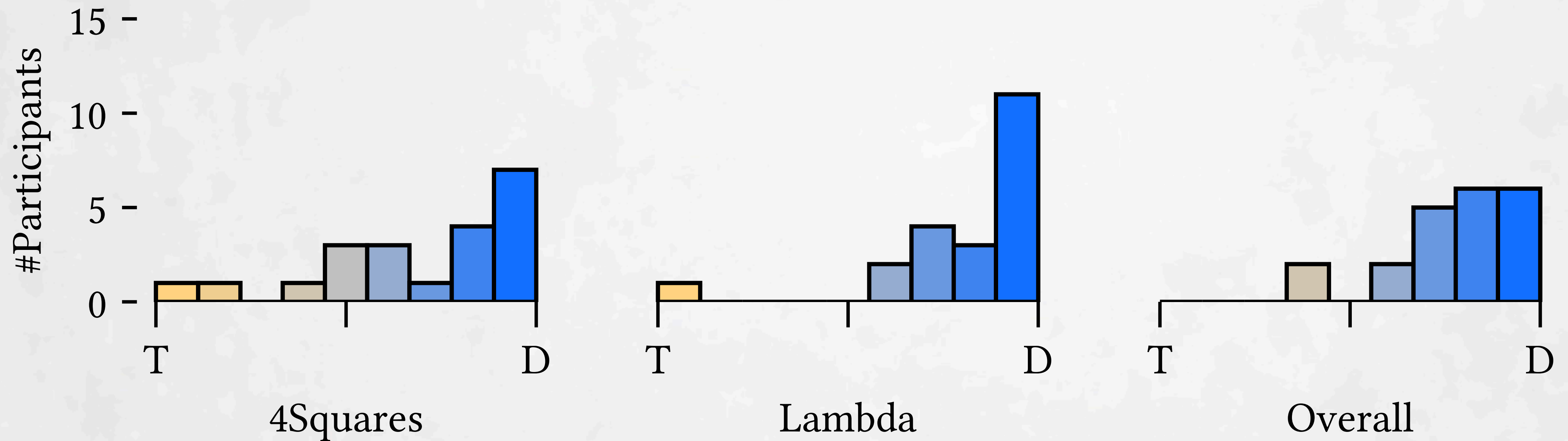
“Which interaction worked better for the ... task?”



Modest subjective preference for Deuce

Mix & Match Tasks

What did participants actually use?



Deuce preferred by almost all users.