Hazel: Semantic Foundations for Interactive Programming Tools

Cyrus Omar

Carnegie Mellon University

IBM PL Day 2016

Q: What do programmers interact with?

Q: What do programmers interact with?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m,
    median =
```

syntactically malformed program text

Syntactic error recovery heuristics

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, □),
    median = □
  }
```

syntactically malformed program text → term with holes

[Kats et al., OOPSLA 2009]

Syntactic structure editors

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

~~syntactically malformed program text~~ → term with holes

[Teitelbaum and Reps, Comm. ACM 1981; many others]

Q: How to **reason statically** about terms with holes?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

Q: How to **reason statically** about terms with holes?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here?

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here?

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here?

A: A **static semantics** for terms with holes.

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here?

A: A **static semantics** for terms with holes.

[Omar et al., POPL 2017]

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here?

```
matrix<float> →
  { mean   : vec<float>,
    std    : vec<float>,
    median : ▢ }
```

A: A **static semantics** for terms with holes.

[Omar et al., POPL 2017]

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢
  }
```

What **type** of expression is expected here? `(RowWise + ColumnWise)`

```
matrix<float> →
  { mean   : vec<float>,
    std    : vec<float>,
    median : ▢ }
```

A: A **static semantics** for terms with holes.

[Omar et al., POPL 2017]

Q: How to **reason statically** about terms with type errors?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, “oops”),
    median = ▢
  }
```

A: A **static semantics** for terms with holes.

[Omar et al., POPL 2017]

Q: How to **reason statically** about terms with type errors?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix<float>) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, "oops"),
    median = ▢
  }
```

```
matrix<float> →
  { mean   : vec<float>,
    std    : vec<float>,
    median : ▢ }
```

A: A **static semantics** for terms with holes.

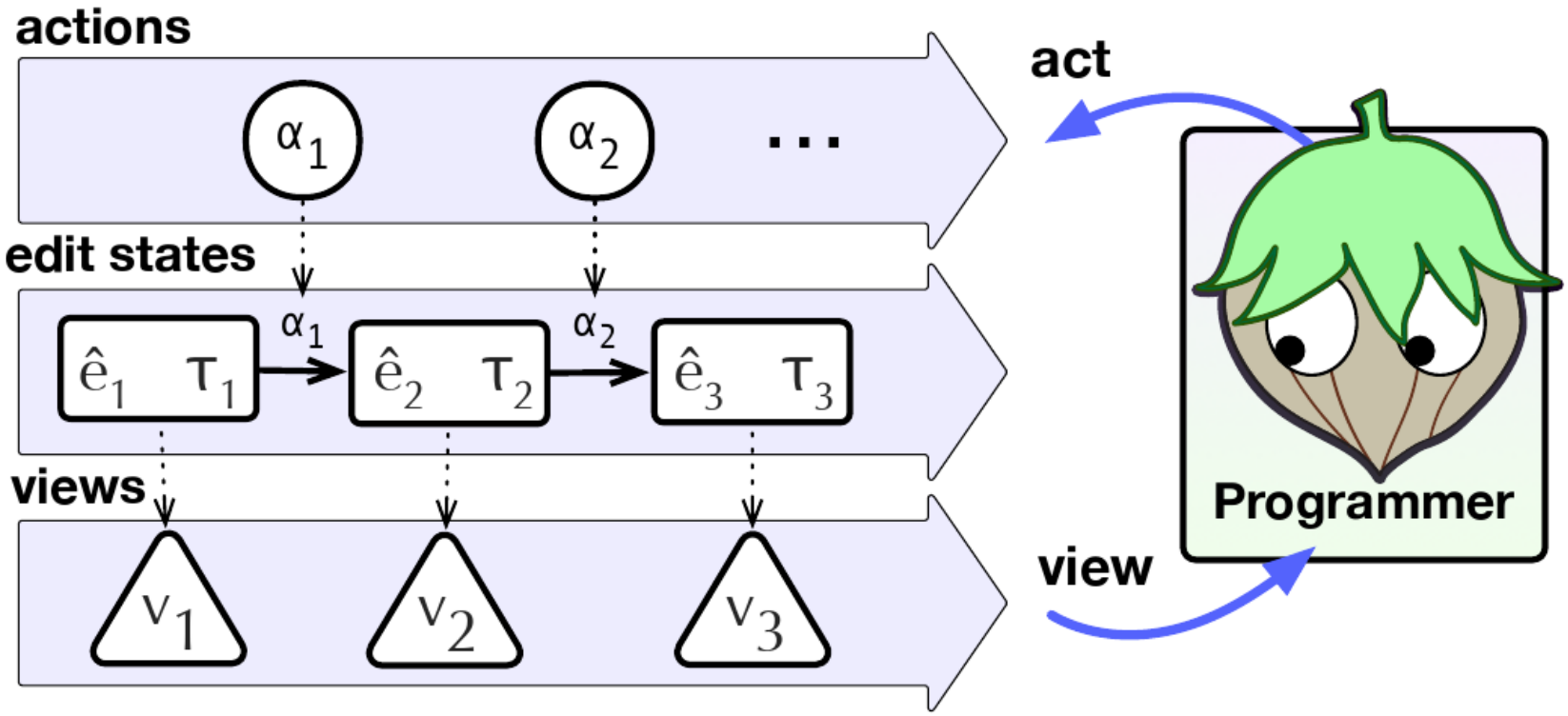
[Omar et al., POPL 2017]

A **static semantics** for lambda terms with holes

$$\begin{array}{llll} \text{HTyp} & \dot{\tau} & ::= & (\dot{\tau} \to \dot{\tau}) \mid \texttt{num} \mid \text{⦇⦈} \\ \text{HExp} & \dot{e} & ::= & x \mid (\lambda x.\dot{e}) \mid \dot{e}(\dot{e}) \mid \underline{n} \mid (\dot{e} + \dot{e}) \mid \dot{e} : \dot{\tau} \mid \text{⦇⦈} \mid \text{⦇}\dot{e}\text{⦈} \end{array}$$

[Omar et al., POPL 2017]

A typed edit action semantics



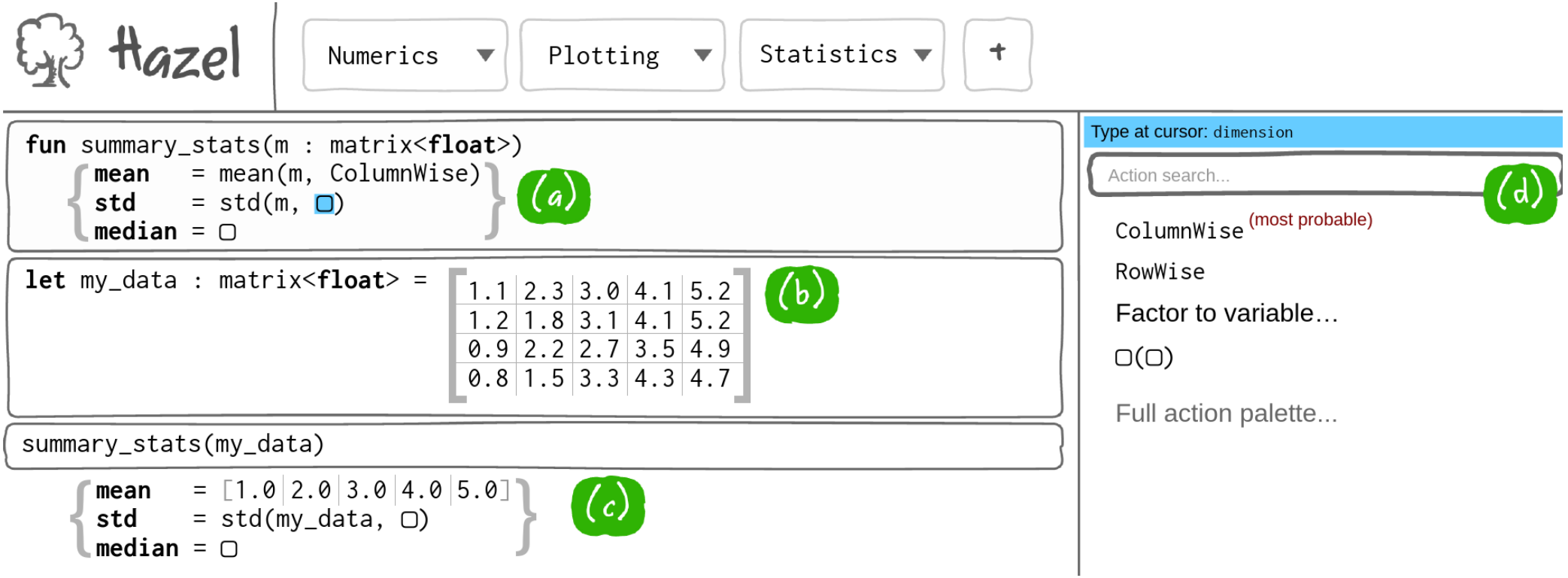
[Omar et al., POPL 2017]

Demo

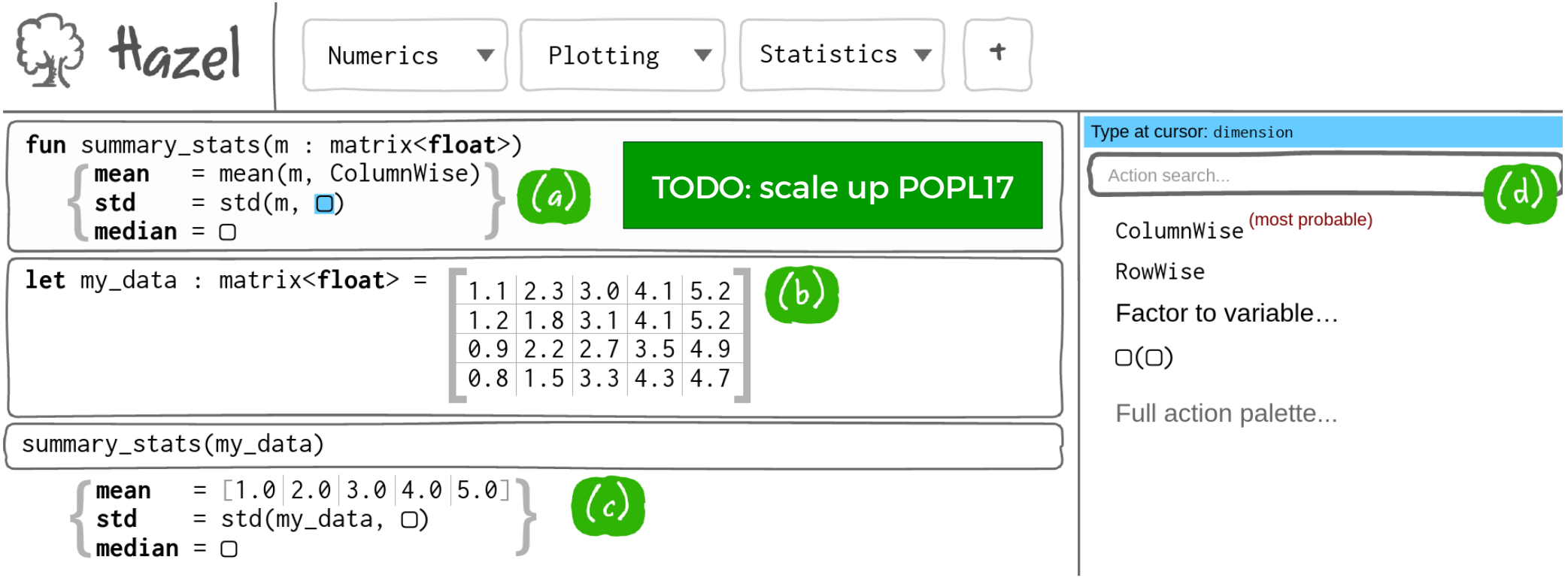
See http://hazelgrove.github.io/

[Omar et al., POPL 2017]

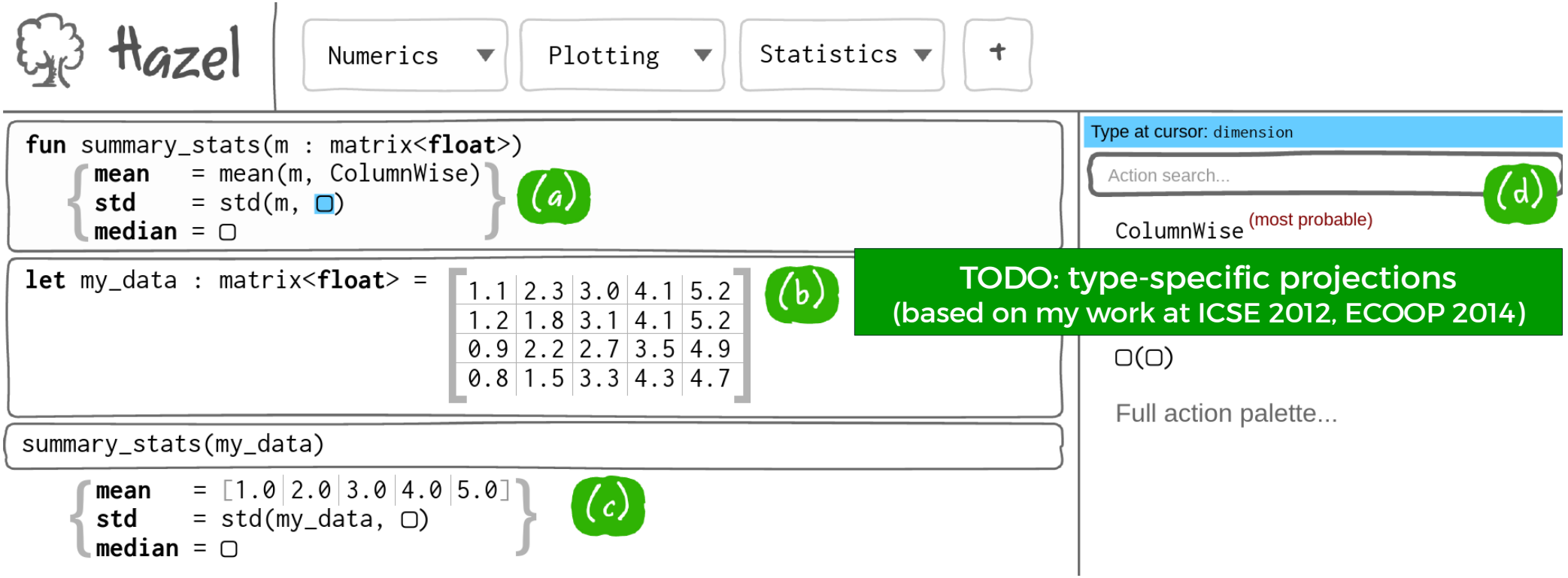
From Hazelnut to Hazel



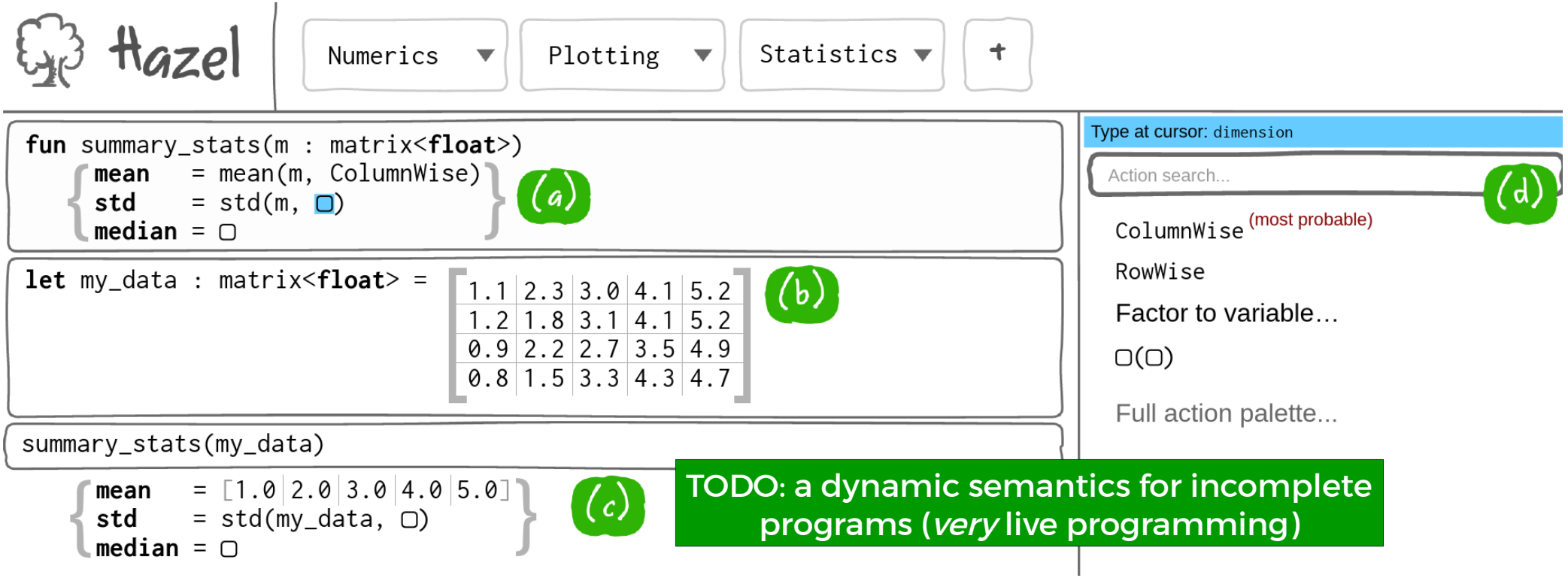
From Hazelnut to Hazel



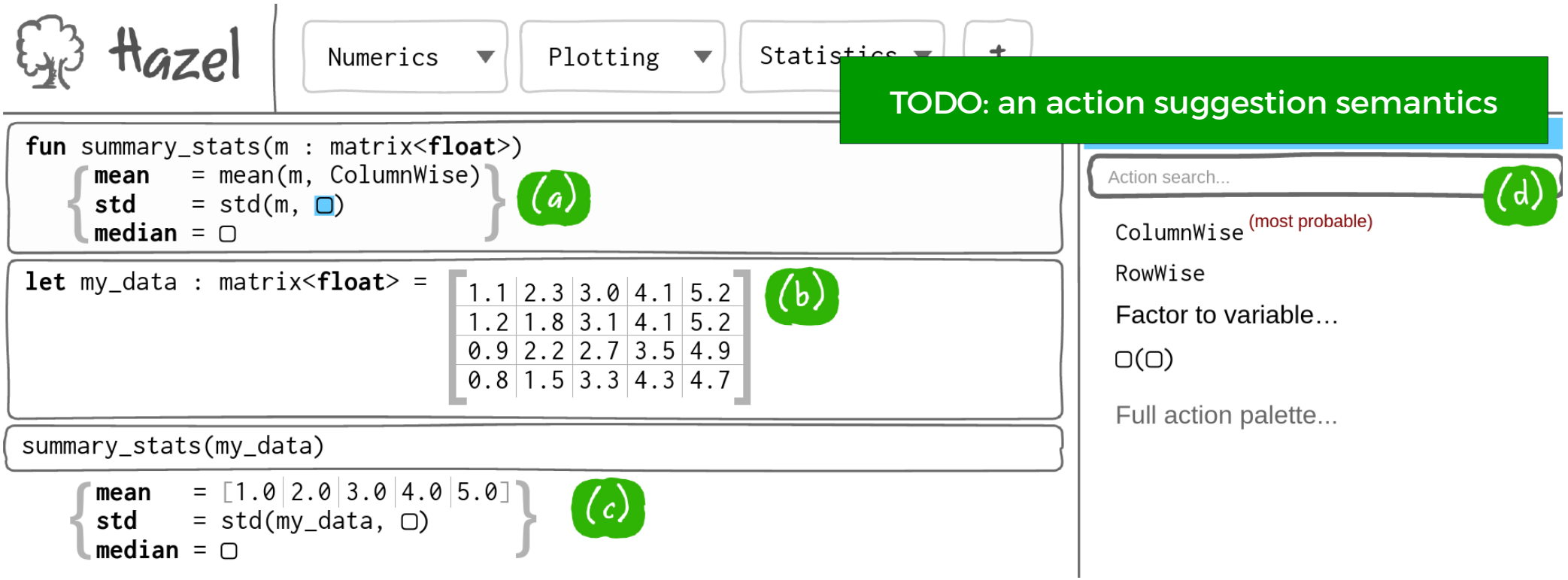
From Hazelnut to Hazel



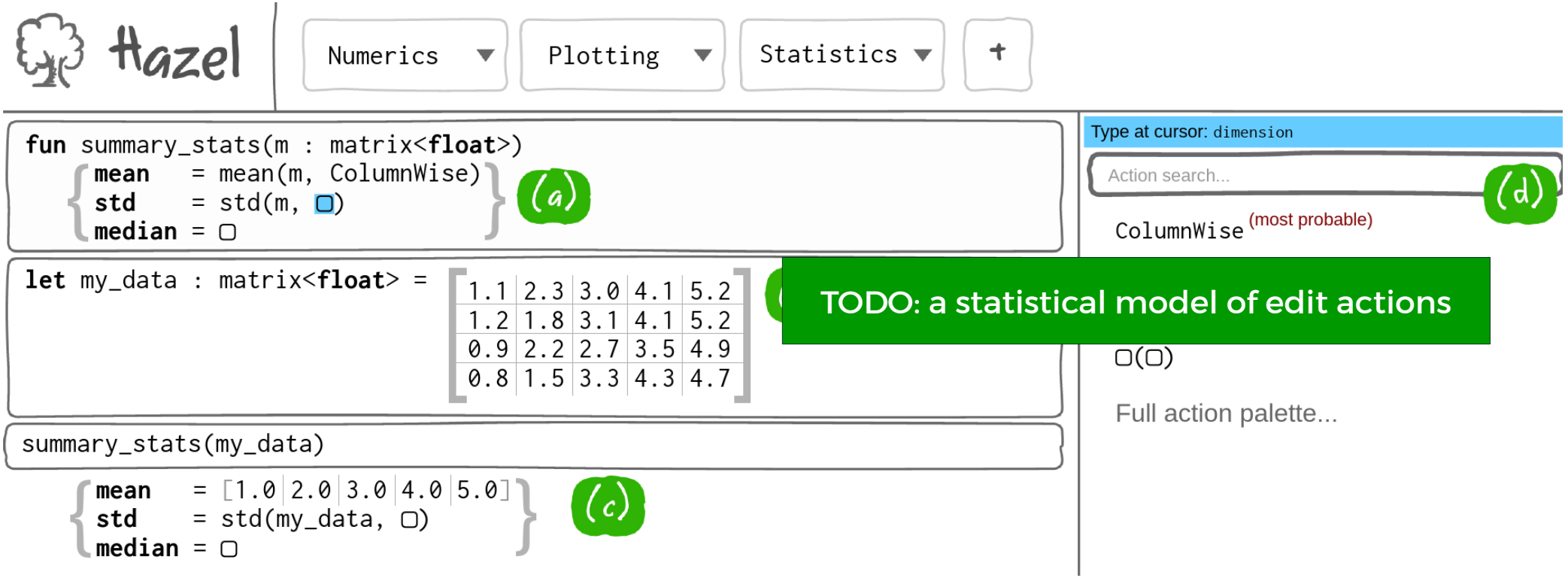
From Hazelnut to Hazel



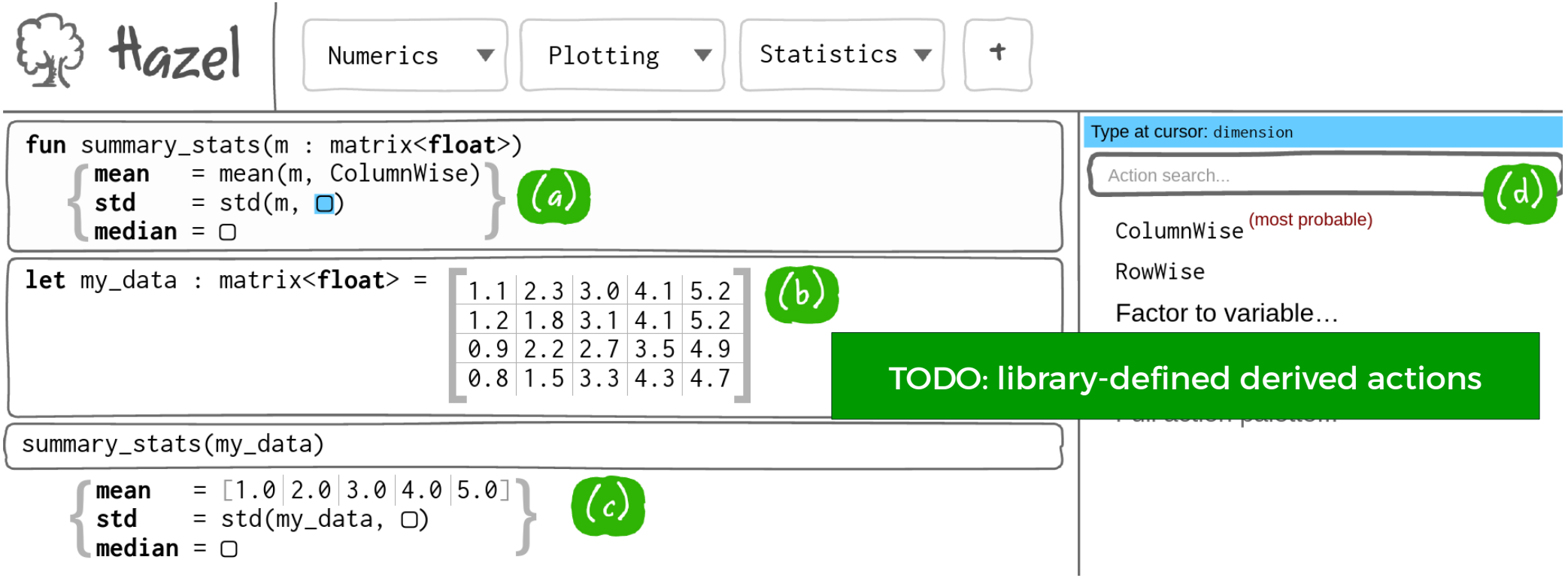
From Hazelnut to Hazel



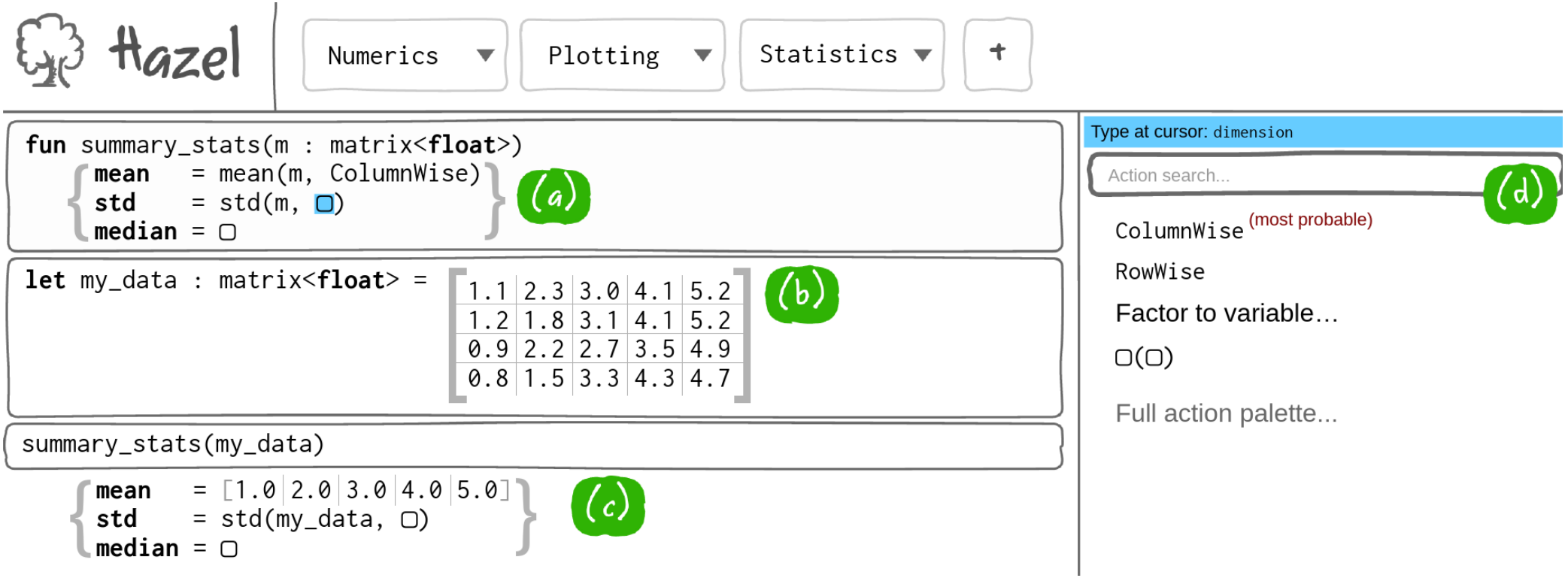
From Hazelnut to Hazel



From Hazelnut to Hazel



From Hazelnut to Hazel



Joint work with Ian Voysey (CMU), Matt Hammer (CU Boulder), Michael Hilton (Oregon State), Claire Le Goues (CMU), Jonathan Aldrich (CMU), Josh Sunshine (CMU).

Interested? Contact me! http://hazelgrove.github.io/