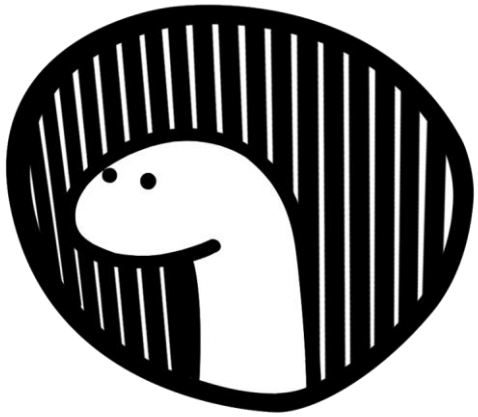
denoの外部モジュールのバージョンについての実証分析

信州大学　実証的ソフトウェア工学研究室

中谷　颯人

背景と目的



●バージョン管理

・バージョンのピン止めなし
・バージョンが古い

⇒

・新しい事象に対応できない
・エラーの発生

●denoという新環境

V8 JavaScriptエンジンをベースに実装された
JavaScript/TypeScriptランタイム環境

・新しい動作環境
・新しいモジュールシステム

⇒ バージョン管理の現状が不明

denoのバージョン管理の現状を分析

◆denoのモジュールシステム

URLを指定してモジュールをimport

https://deno.land/std@0.50.0/http/server.ts から関数serveをimportできる

```
import { serve } from “https://deno.land/std@0.50.0/http/server.ts”;
```

◆importのされ方の分類

自身のモジュール（相対パス）

```
import { groupBy } from "../collections/group_by.ts";
```

外部モジュール（バージョン定義なし、絶対パス）

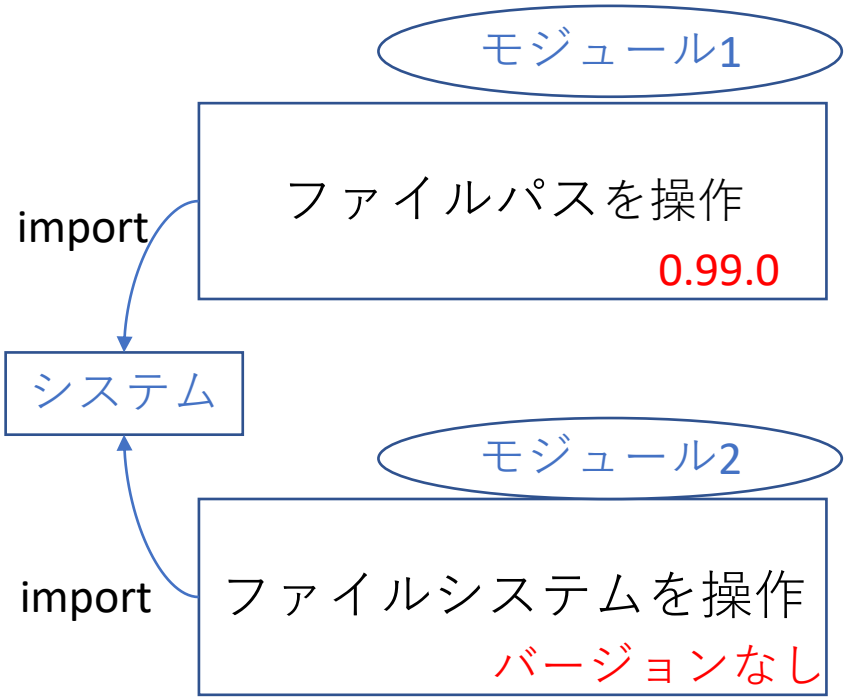
```
import fc from "https://cdn.skypack.dev/fast-check";
```

外部モジュール（バージョン定義あり、絶対パス）

```
import {
  assert,
  assertEquals,
} from "https://deno.land/std@0.90.0/testing/asserts.ts";
```

◆バージョン管理しないと・・・

バージョン指定なしの場合、自動的に最新のバージョンになる



モジュール2の新しいバージョンが出る

↓

モジュール2のバージョンが自動で最新に

↓

・モジュール1との依存関係に問題
・悪意のあるアップデート

◆実際に壊れた例

●概要

denoでstd/hashというモジュールが削除、
std/cryptoというモジュールに移行

その際に、壊れるモジュールが発生した

●原因

バージョン指定がされていなかったから

今回の事例

https://deno.land/std/hash/mod.ts

バージョン指定をしていないために、削除された際に参照先がなくなってしまってエラーが起きた

解決例

https://deno.land/std**@0.160.0**/hash/mod.ts

バージョン指定をしていればdeno上の過去のバージョンを参照してエラーがでない

分析方法

① deno上の4,811個のモジュールから
importされているURLを全て取得

```
import { serve } from “https://deno.land/std@0.50.0/http/server.ts”;

import { groupBy} from “../collections/group_by.ts”;
```

② ①の中で、外部モジュール
（絶対パスでimportされているURL）を取得

```
import { serve } from “https://deno.land/std@0.50.0/http/server.ts”;

import{ groupBy} from “../collections/group_by.ts”;
```

③　外部モジュールのURLからレジストリ名、組織名、パッケージ名、バージョン名、モジュール名を取得

https://deno.land/std@0.130.0/http/http_status.tsでの例

レジストリ	組織	パッケージ	バージョン	モジュール
deno.land/std	undefined	undefined	0.130.0	http/http_status.ts

④　③で得たデータを集計し、図表を作成、分析

リサーチクエスチョン

1.バージョン指定にどのような特徴があるか

2.バージョンの分布にどのような特徴があるか

3.レジストリごとのバージョン指定にはどのような特徴があるか

4.バージョン指定する側にはどのような特徴があるか

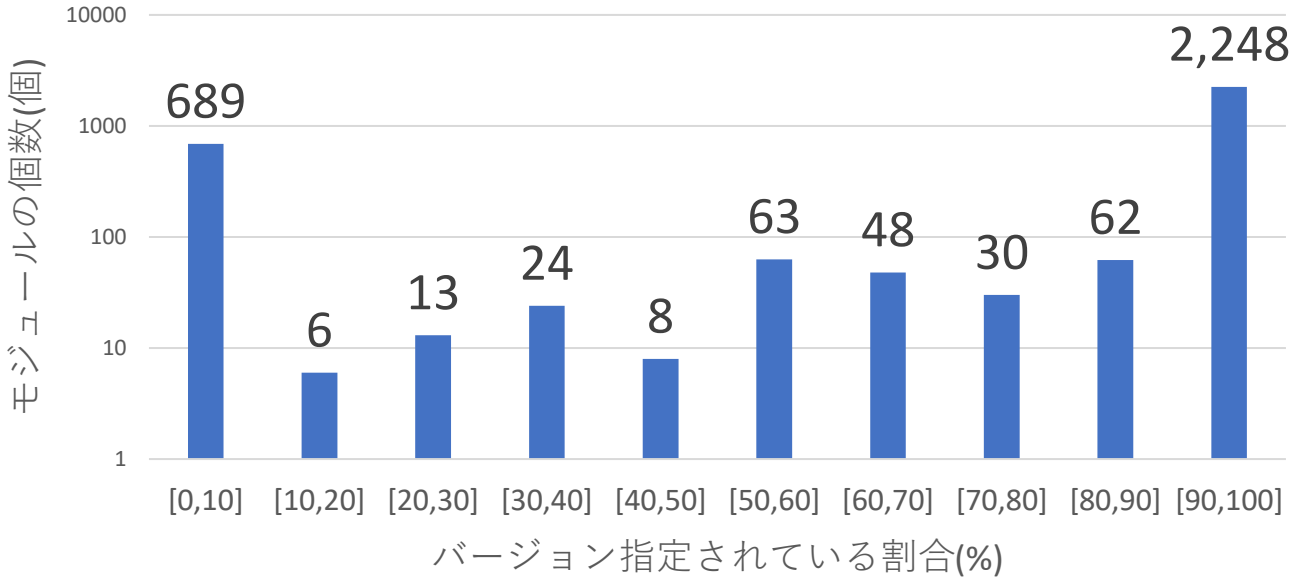
分析結果

RQ1.バージョン指定の特徴

モジュールごとのバージョン指定の表

importされたモジュール名	バージョン指定されている数(回)	バージョン指定されていない数(回)	importされた回数(回)	バージョン指定されている割合(%)
deno.land/std/testing/asserts.ts	2,652	602	3,254	81.5
deno.land/std/path/mod.ts	996	142	1,138	87.52
deno.land/x/polkadot/util/mod.ts	927	0	927	100.0
・ ・	・ ・	・ ・	・ ・	・ ・

バージョン指定されている割合のグラフ



・グラフが右寄り

・バージョン指定されている割合の平均値は84.8％

バージョン指定されているモジュールは多い

RQ2.モジュールごとのバージョン分布

モジュールごとのバージョンの種類の表

importされたモジュール名	バージョンの種類と回数
deno.land/x/polkadot/types/interfaces/recovery/index.ts	{'0.0.6': 1, '0.0.3': 1, '0.0.7': 1}
esm.sh/@unocss/preset-web-fonts	{'0.43.2': 1, '0.41.1': 1, '0.31.6': 1}
deno.land/x/postcss/mod.js	{'8.3.0.1': 1, '8.4.5': 1, '8.3.5': 1, '8.3.11': 1, '8.3.6': 1, '8.3.0': 1, '8.4.13': 3}
・ ・	・ ・ **({バージョン:回数})**

評価方法

・モジュールごとに、バージョンの中央値の得点（中央得点）と、
最頻値の得点（最頻得点）を付ける

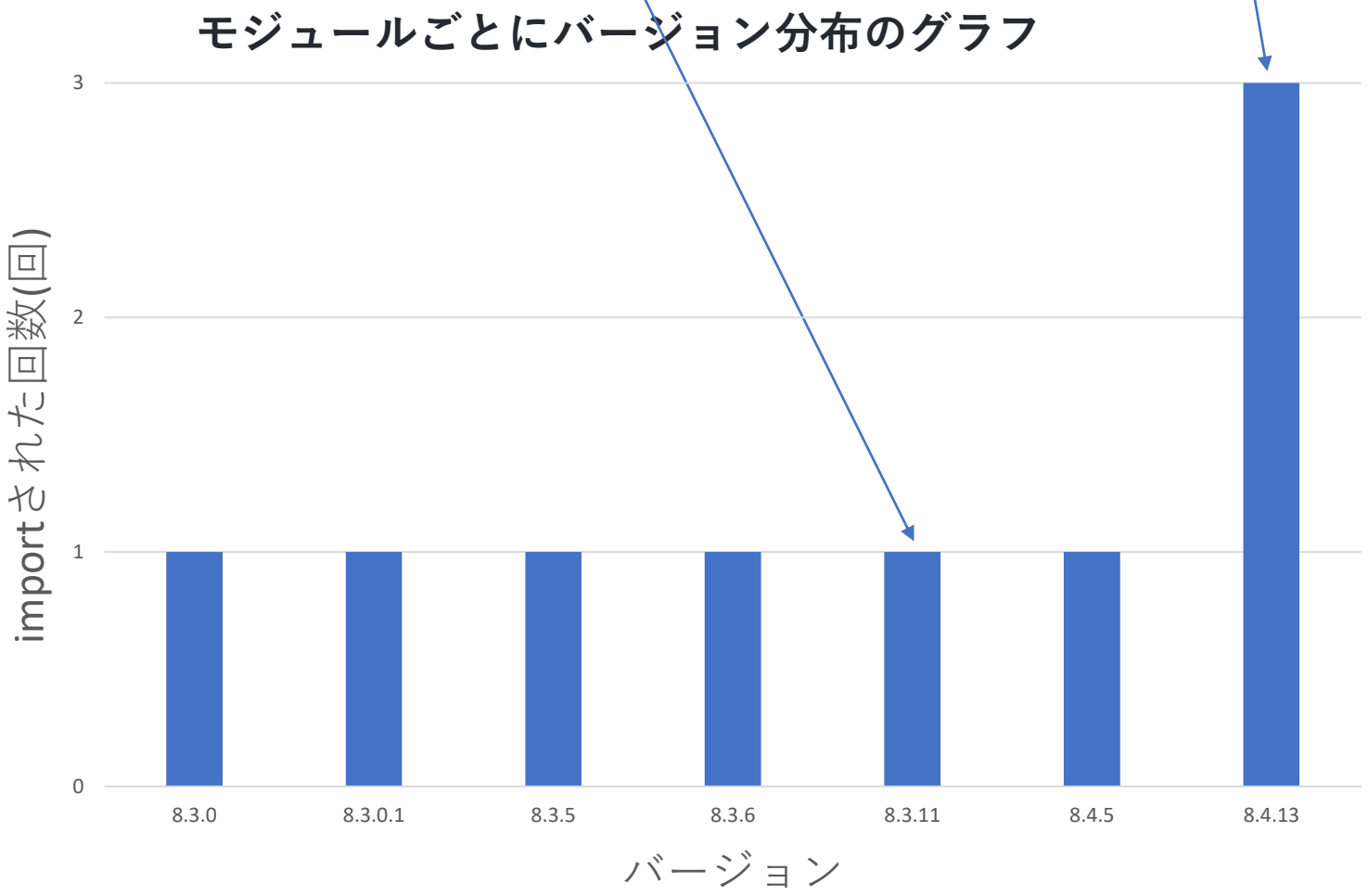
・両方の得点が高いほど、バージョンが新しいものが多く、良い

$$\text{得点} = \frac{\text{中央（最頻）値のある階級が何番目にあるか}}{\text{バージョンの種類数}} \times 100$$

得点付けの例

中央値：5つめの階級　　**最頻値**：7つ目の階級

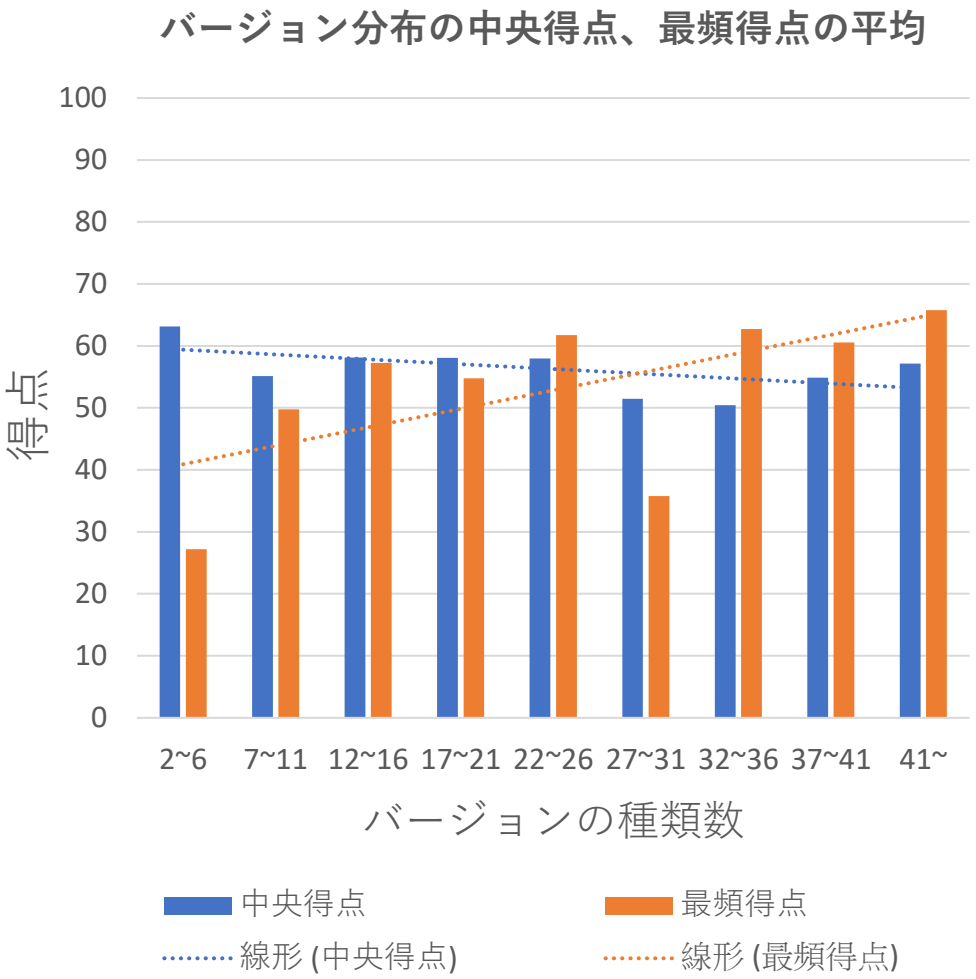
中央得点：(5÷9)×100=55.6　　**最頻得点**：(9÷9)×100=100



	2~6	7~11	12~16	17~21	22~26	27~31	32~36	37~41	41~ (種)
中央得点(点)	63.13	55.11	58.15	58.07	57.99	51.49	50.43	54.89	57.15
最頻得点(点)	27.19	49.74	57.25	54.76	61.74	35.75	62.71	60.54	65.77
モジュールの個数 (個)	569	34	17	6	12	6	5	6	9

・最頻得点は、バージョンの種類数が多いほど、増加傾向にある
(27～31種類の階級は例外)

バージョンの種類数が多いほど、新しいバージョンをimportする傾向にある



RQ3.レジストリごとのバージョン指定

レジストリの分類

レジストリ名	特徴
・deno.land/x ・deno.land	deno公式のモジュールレジストリ
・github.com ・denopkg.com ・ghuc.cc	Github関連のレジストリ
・esm.sh ・jspm.dev ・unpkg.com ・skypack.dev	npmパッケージを配布するレジストリ
・nest.land	ブロックチェーン上のパッケージレジストリ

レジストリごとのバージョン指定の割合

レジストリ名	importされた回数(回)	バージョン指定数(回)	バージョン指定されている割合(%)
deno.land/x	10,409	9,012	86.6
deno.land	9,735	8,501	87.3
github.com	444	241	54.3
denopkg	77	45	58.4
ghuc.cc	75	0	0
esm.sh	1,494	1,276	85.4
jspm.dev	561	177	31.6
unpkg.com	85	75	88.2
skypack.dev	544	365	67.1
nest.land	48	48	100

・Github関連のレジストリのバージョン指定の割合が低い

・deno公式のレジストリのバージョン指定の割合が高い

RQ4.importする側の特徴

deps.ts：export文をまとめたファイル

メリット：バージョンの変更が容易に
(deps.tsのみ編集すれば良い)

deps.ts

```
export {decode} from "https://deno.land/std@0.53.0/encoding/utf8.ts";
export {
 deferred,
 Deferred,
} from "https://deno.land/std@0.53.0/async/deferred.ts";
```

ここを変更するだけ

mod.ts

```
import { deferred, decode } from "./deps.ts";
```

ファイル	ファイルの個数 (個)	バージョン指定した数 (回)	Importした回数 (回)	ファイルの数の割合 (%)	バージョン指定した割合 (%)
deps.ts	1,502	4,334	4,852	13.3%	89.3%
deps.ts以外	9,860	15,685	18,823	86.7%	83.3%

・deps.tsの方が、バージョン指定した割合が6％高い

・deps.tsが使われている割合は13.3%と低い

まとめ

・バージョン指定はされているモジュールが多い
→しかし、レジストリによってされやすいもの、されにくいものがある

・バージョンの種類数が多いほど、新しいバージョンがimportされやすい

・deps.tsを使用しているモジュールが少ない
→使用しているモジュールの方がバージョン指定率も高いため、使用者が増えたほうが良い

今後の展望

・ネットワーク図を用いた分析

・データの再収集、比較