Quarto pro vědecké výpočty a publikování

Jan Rutterle

Obsah

Chapter 1

Úvod

V tomto dokumentu si představíme Quarto. Quarto je open-source vědecký a technický publikační systém, který využívá pro tvorby markdown jazyk nebo Jupyter notebooky. V dokumentech jednoduše zakomponujeme i kusy kódu, pro vytvoření více dynamického vzhledu. Díky využití Pandoc konvertoru vytvoříme s tímto nástrojem téměř cokoliv od článků, přes prezentace, až po webové stránky nebo celé knížky. V tomto dokumentu si představíme základy tvorby takovýchto dokumentů. Zdrojový kód tohoto dokumentu, který je vytvořen též v Quartu, naleznete zde: https://github.com/ruttejan/SemestralProject. Příklad prezentace vytvořené pomocí Quarto naleznete zde: https://github.com/ruttejan/PCA.

Chapter 2

Instalace a příprava

Pro práci s Quarto dokumenty si musíme nejprve stáhnout samotné Quarto. Na této URL adrese (https://quarto.org/docs/get-started/) naleznete odkazy pro stažení na různé operační systémy. Dále zde najdete i krátké tutoriály na tvorbu samotných dokumentů v různých editorech. V tomto dokumentu se zaměříme na vývoj ve VS Code. Pro práci s Quartem ve VS Code je potřeba nainstalovat rozšíření "Quarto". To uděláme přímo ve VS Code v sekci "Extensions" nebo si můžeme rozšíření stáhnout zde: https://marketplace.visualstudio.com/items?itemName=quarto.quarto-vscode. (Scheidegger et al., n.d.)

2.1 Potřebné soubory a výstupní formáty

Zdrojový kód pro Quarto dokumenty se píše do souborů s příponou „.qmd". Pro další úpravy jsou však vhodné i jiné typy souborů jako například „.css" (kaskádové styly pro úpravu vzhledu dokumentu), „.bib" (seznam využité bibliografie) a další. Zpracování souborů vypadá následovně. Nejprve jsou zpracovány bloky kódu Jupyterem, následně se vše převede do čistého markdown jazyku, který je upraven Pandoc konvertorem do finální podoby. Jelikož je Pandoc velice robustní převodník, možností výstupního formátu je spoustu. Formátem, který byl použit pro tvorbu ukázkové prezentace, je Revealjs, což je open source šablona pro tvorbu HTML prezentací. Dalšími výstupními formáty pro prezentace jsou PowerPoint a Beamer. Pro jiné dokumenty můžeme využít jako výstup třeba HTML nebo PDF. S Quartem lze vytvořit i webové stránky nebo knížky. Možností je opravdu hodně. Veškeré výstupní formáty naleznete zde: https://quarto.org/docs/output-formats/all-formats.html.

2.2 Render a práce ve VS Code

2.2.1 Render

Pro render dokumentu ve VS Code s Quarto rozšířením využijeme tlačítka „Preview“ v pravém horním rohu okna aplikace nebo využijeme klávesové zkratky “Ctrl+Shift+K”. Případně můžeme použít příkazovou řádku:

```
quarto preview nazevsouboru.qmd
```

Toto vyrenderuje dokument do předem nastaveného formátu a zobrazí náhled ve webovém prohlížeči. Výchozím formátem je HTML.

Náhled se nám zobrazí lokálně ve webovém prohlížeči a vyrenderované soubory budou uloženy do složky ke zdrojovému kódu. Pokud chceme pouze vyrenderovat soubory bez náhledu, uděláme to přes příkazovou řádku.

```
quarto render názevsouboru.qmd
```

Pokud chceme soubor vyrenderovat do jiného formátu, než máme nastavený, použijeme parametr `--to` a název formátu.

Například pro render do docx souboru použijeme:

```
quarto render názevsouboru.qmd --to docx
quarto preview názevsouboru.qmd --to docx
```

2.2.2 Render on Save

Pro zjednodušení práce můžeme využít možnosti „Render on Save“, kterou si zapneme v nastavení VS Code (Settings -> Extensions -> Quarto -> Render: Render On Save). Popřípadě si toto chování můžeme nastavit zvlášť pro každý dokument v YAML možnostech (viz. Nastavení dokumentu) pomocí:

```
    editor:
        render-on-save: true
```

Toto nastavení nám vyrenderuje dokument po každém uložení souboru.

2.2.3 Možnosti editorů

Ve VS Code můžeme v základu psát Quarto dokumenty pomocí markdown jazyku a dalších textových příkazů, které jsou vysvětleny níže. Dále je možné využít Visual editoru, který umožňuje psát dokumenty podobně jako v MS Word. Poslední možností je vytvořit .ipynb notebook, který vyrenderujeme standardním způsobem, akorát jako vstupní dokument použijeme právě .ipynb soubor.

```
quarto render nazevsouboru.ipynb
```

Chapter 3

Nastavení dokumentu

Pro globální nastavení dokumentu použijeme YAML (Yet Another Markup Language), jazyk sloužící k reprezentaci dat v čitelné podobě pro lidi. Možnosti nastavení pomocí YAML píšeme na úplný počátek dokumentu. Pro tvorbu webové stráky můžeme použít separátního _quarto.yaml souboru pro specifikaci obecných informací platících pro všechny části webové stránky, jako jsou například navbar nebo page-footer. Počet možností YAML je veliký, proto si zde shrneme jenom základní nastavení. Pokud si chcete prohlédnout všechny možnosti pro jednotlivé formáty dokumentů, můžete využít stejný odkaz jako v sekci Section 2.1.

3.1 Titulek a autor

Titulek slouží jako hlavní nadpis. V případě prezentací v Revealjs je to nadpis úvodního slajdu. Vyplněný autor se zobrazí hned pod titulkem.

```
    title: "PCA"
    author: "Jan Ruterle"
```

Můžeme doplnit i podtitulek (subtitle: “ ”), datum (date: “ ”), institut (institute: “ “).

3.2 Jupyter

Pro executování kódu v dokumentu je potřeba mít nainstalovaný Jupyter kernel pro programovací jazyk, který chceme použít. Programovací jazyky, které jsou v Quartu podporované pro exekuci, jsou Python, R, Julia a Observable JS. Použitý kernel specifikujeme pomocí:

```
  jupyter: julia-1.9
```

Abychom mohli jednotlivé kernely používat, musíme je nejprve stáhnout. Návody na instalaci naleznete zde:

- Python: https://quarto.org/docs/computations/python.html.
- R: https://quarto.org/docs/computations/r.html.
- Julia: https://quarto.org/docs/computations/julia.html.
- Observable JS: https://quarto.org/docs/computations/ojs.html.

3.3 Nastavení výstupního formátu

Nastavení výstupního formátu je klíčové pro tvorbu dokumentu. Výstupní formát určuje, jak bude dokument vyrenderován. Výstupní formát nastavíme pomocí:

```
  format: revealjs
```

Každý výstupní formát má svá specifická nastavení, která můžeme upravit. Všechny možnosti výstupních formátů a jejich nastavení najdeme zde: https://quarto.org/docs/output-formats/all-formats.html.

Pro revealjs jsme si již v sekci 3.1 představili nastavení titulku a autora. Další užitečná nastavení si představíme v následujících sekcích.

3.3.1 Nastavení vzhledu prezentací

Vzhled dokumentu můžeme upravit pomocí atributu „theme“. Všechna dostupná témata najdeme zde: https://revealjs.com/themes/. Pro nastavení vzhledu použijeme:

```
  theme: white
```

Případně využijeme i vlastního scss souboru. Nebo kombinaci vlastního nastavení a předdefinovaného tématu.

```
  theme: [white, custom.scss]
```

Úpravy vzhledu můžeme provést i pomocí nastavení „css“.

```
  css: styles.css
```

3.3.2 Číslování a Obsah

Pro číslování nadpisů použijeme atribut "number-sections". Pro upřesnění toho, jaké nadpisy se budou číslovat použijeme "number-depth". Pro vypsání obsahu použijeme "toc" a "toc-title". Jaké všechny nadpisy se budou vypisovat do obsahu můžeme nastavit pomocí "toc-depth".

```
number-sections: true
number-depth: 2
toc: true
toc-title: Obsah
toc-depth: 2
```

3.3.3 Slajdy

V následujících bodech si představíme užitečné atributy pro prezentování slajdů.

Incremental

Atribut "incremental" slouží k zobrazení obsahu slajdu po částech v případě nastavení na "true". V případě nastavení na "false" se zobrazí celý obsah slajdu najednou.

```
incremental: true
```

Slide-number

Atribut "slide-number" slouží k zobrazení čísla slajdu. V případě nastavení na "true" se zobrazí číslo defaultně ve tvaru "aktuální slajd / celkový počet slajdů". Dalšími možnostmi jsou kromě "false" ještě "c" (aktuální slajd), "h.v" (aktuální slajd ve vertikálním pořadí . aktuální slajd ve vodorovném pořadí) a "h/v" (stejně jako "h.v", ale jiném stylu). Horozontální slajdy jsou ty, které jsou na stejné úrovni, vertikální jsou ty, které jsou o úroveň níže. Vertikální slajdy se vytvoří defaultně za použití nadpisu 3. úrovně. O nadpisech více níže v sekci Section 4.1.

```
slide-number: true
slide-number: h/v
```

Center-title-style

Atribut "center-title-style" slouží k vertikálnímu vycentrování nadpisu na titulním slajdu. Defaultně je nastaven na "true".

```
center-title-style: false
```

Logo

Atribut “logo” slouží k vložení loga do pravého dolního rohu slajdů. Logo musí být uloženo ve složce s dokumentem.

```
logo: logo.png
```

Footer

Atribut “footer” slouží k vložení textu do středu patičky slajdů.

```
footer: "FEL ČVUT"
```

Scrollable

Atribut “scrollable” slouží k nastavení scrollování slajdů. V případě, že obsah přesahuje slajd, tak se, při nastavení na “true”, zobrazí scrollbary.

```
scrollable: true
```

Toto se dá nastavit zvlášť pro každý slajd. K nadpisu doplníme “{.scrollable}”.

```
# Nadpis slajdu {.scrollable}
```

Smaller

Atribut “smaller” slouží k nastavení menší velikosti textu na slajdech.

```
smaller: true
```

Tento atribut se dá nastavit zvlášť pro každý slajd. K nadpisu doplníme “{.smaller}”.

```
# Nadpis slajdu {.smaller}
```

Overview

Overview je možnost zobrazit všechny slajdy najednou. Tuto možnost můžeme zapnout během prezentováí pomocí klávesy “o”. Defaultně je tato možnost zapnutá.

```
overview: false
```

Menu

Toto nastavení slouží k zobrazení menu v levém dolním rohu slajdů, které obsahuje seznam všech slajdů a umožňuje libovolně přecházet mezi nimi. Defaultně je tato možnost zapnutá.

```
    menu: false
```

Chalkboard

Chalkboard slouží k zobrazení tabule na kreslení na slajdech. Máme na výběr kreslit na separátní tabuli nebo přímo do slajdů. Defaultně je tato možnost vypnutá.

```
    chalkboard: true
```

Transition

Atribut "transition" slouží k nastavení přechodů mezi slajdy. Všechny tipy přechodů jsou: none, fade, slide, convex, concave, zoom. Defaultně je nastaven na "none".

```
    transition: slide
```

Transition-speed

Pomocí "transition-speed" můžeme nastavit rychlost přechodů mezi slajdy. Všechny možnosti jsou: default, fast, slow. Defaultně je nastaven na "default".

```
    transition-speed: fast
```

Chapter 4

Základy tvorby dokumentu

4.1 Nadpisy

Pomocí nadpisů oddělujeme jednotlivé slidy. Nadpisy se vytváří pomocí znaku #. V Quartu můžeme vytvořit 6 úrovní nadpisů.

```
# Nadpis 1. úrovně
## Nadpis 2. úrovně
### Nadpis 3. úrovně
#### Nadpis 4. úrovně
##### Nadpis 5. úrovně
###### Nadpis 6. úrovně
```

4.2 Text

Pod nadpisy píšeme text, který můžeme různě upravovat. Můžeme použít tučné písmo, kurzívu, přeškrtnutí, horní/dolní index nebo doslovný kód.

markdown	výstup
`*kurzíva*`	*kurzíva*
`**tučné písmo**`	**tučné písmo**
`***tučná kurzíva***`	***tučná kurzíva***
`~~přeškrtnutí~~`	~~přeškrtnutí~~
`H~2~O`	H_2O (dolní index)
`X^2^`	X^2 (horní index)
‘doslovný kód‘	`doslovný zápis`

4.3 Odkazy

Do dokumentů můžeme vkládat odkazy na webové stránky. URL adresu vložíme do <>.

```
<https://quarto.org>
```

https://quarto.org

Můžeme doplnit i alternativní text, který se zobrazí místo celé URL adresy. Tento text umístíme do hranatých závorek před odkaz, který nyní ohraničíme normálními závorkami místo zobáčkových.

```
[Quarto](https://quarto.org)
```

Quarto

4.4 Obrázky

Obrázky vkládáme podobně jako odkazy. K obrázkům můžeme vložit URL adresu, na kterou budeme odkázáni po kliknutí na daný obrázek. U každého obrázku vidíte jeho zdrojový kód a výstup. Každý obrázek má upravenou velikost. Tyto úpravy provede pomocí css parametrů, které napíšeme do složených závorek. Obrázky musí být uloženy ve složce s dokumentem.

Obrázek s popiskem (pokud nechceme k obrázku žádný popisek, necháme hranaté závorky prázdné):

```
![roman_numerals1](roman_numerals.jpg){width=100px, height=100px}
```

Figure 4.1: roman_numerals1

Obrázek s odkazem:

```
[![roman_numerals2](roman_numerals.jpg){width=100px, height=100px}](/odkaz)
```

Figure 4.2: roman_numerals2

Obrázek s odkazem a textem, který se zobrazení po najetí myší na obrázek:

```
[![roman_numerals3](roman_numerals.jpg){width=100px,height=100px}](/odkaz "Římské číslice")
```

Figure 4.3: roman_numerals3

Zdroj obrázku: https://cs.wikipedia.org/wiki/%C5%98%C3%ADmsk%C3%A9_%C4%8D%C3%ADslice

Můžeme přidat i alternativní text v případě nevykreslení obrázku:

```
[![](obrázek.png){fig-alt="Alt text"}](https://quarto.org)
```

4.5 Seznamy

Seznamy můžeme tvořit seřazené i neseřazené. Seřazené seznamy vytváříme pomocí číslic a písmen. Neseřazené seznamy pomocí pomlček, hvězdiček nebo plusů. Vnořené seznamy vytváříme pomocí odsazení. Před každým seznamem je potřeba nechat volný řádek, aby se správně vyrenderoval.

```
1. První položka
    i. Druhá položka
    ii. Třetí položka
```

1. První položka
    i. Druhá položka

ii. Třetí položka

```
+ První položka
    - Druhá položka
    - Třetí položka
```

- První položka
  - Druhá položka
  - Třetí položka

Můžeme vytvářet jednoduše i definice pojmů.

```
Pojem
:   Definice
```

Pojem Definice

4.6 Tabulky

Tabulky v Quartu tvoříme několika způsoby. Základním způsobem je pomocí markdown. V prvním řádku definujeme názvy sloupců. Sloupce vždy oddělujeme "|". V druhém řádku můžeme definovat zarovnání sloupců pomocí vhodně umístněné dvojtečky. Dále už jen definujeme obsah tabulky po řádcích. V následující tabulce je ukázka možností zarovnání sloupců. (Scheidegger et al., n.d.)

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
|   12  |  12  |    12   |    12  |
|  123  |  123 |   123   |   123  |
|    1  |    1 |     1   |     1  |

: Tabulka1 {#tbl-alignment}
```

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Table 4.2: Tabluka1

K tabulkám můžeme přidat i popisek. Popisek se vytvoří pomocí dvojtečky za tabulkou a textu popisku. Dále můžeme upravit i šířku sloupců (v procentech).

```
: Tabulka s různým zarovnáním sloupců {tbl-colwidths="[10, 10, 10, 70]"}
```

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Table 4.3: Tabulka s různým zarovnáním sloupců

Můžeme také vytvořit referenční odkaz na tabulky pomocí `{#název}` (viz. první ukázka tabulky). Na tabulku se odkážeme pomocí `@název`. Odkaz na první tabulku: Table 4.2

Popisek tabulky se nám zde zobrazil pod tabulkou. Toto můžeme přenastavit pomocí YAML atribut "tbl-cap-location". Možnosti jsou "top" (popisek nad tabulkou), "bottom" (popisek pod tabulkou) a "margin" (popisek na okraji stránky). Defaultně je nastaven na "top".

```
    tbl-cap-location: bottom
```

Tabulky můžeme tvořit i jinými způsoby. Pro další možnosti vytváření tabulek si můžete přečíst dokumentaci zde: https://quarto.org/docs/authoring/tables.html.

4.7 Matematické vzorce

Do dokumentů můžeme vkládat i matematické vzorce. Ty vkládáme do jednoduchých nebo dvojitých dolarů. Vzorce napsané v jednoduchých dolarech se vyrenderují na stejný řádek, vzorce v dvojitých dolarech na samostatný řádek.

```
    Na stejném řádku: $a^2 + b^2 = c^2$

    Na novém řádku: $$y = ax + b$$
```

Na stejném řádku: $a^2 + b^2 = c^2$.

Na novém řádku:

$$y = ax + b$$

Pro vytváření vzorců používáme LaTeX syntaxi, jelikož jsou vzorce defaultně zpracovávány pomocí MathJax. Pokud chceme, aby byly vzorce zpracovány jinak, použijeme YAML atribut "html-math-method". Základními možnostmi

jsou “mathjax”, “katex” a “plain”. Všechny možnosti naleznete zde: https://quarto.org/docs/output-formats/html-basics.html#latex-equations.

```
    html-math-method: katex
```

4.8 Úpravy vzhledu dokumentu

Dokument a jeho části můžeme upravovat pomocí CSS a SCSS souborů. V RevelaJS prezentacích můžeme využít nejprve předdefinovaných témat Section 3.3.1. Další úpravy, které budou platit pouze pro určité slajdy, můžeme specifikovat přímo k nadpisu slajdu. Úpravy provedeme pomocí css parametrů, které napíšeme do složených závorek.

```
    # Nadpis slajdu {.style="color: red;"}
```

Popřípadě můžeme text obalit do divu a upravit jeho vzhled pomocí css parametrů stejně jako ukázáno nahoře nebo si úpravy můžeme napsat do separátního souboru. CSS soubor musí být uložen ve složce s dokumentem a propojen pomocí YAML atributu “css”.

```
    css: styles.css
```

Vytvoření divu s názvem třídy:

```
    ::: {.nazevtridy}
    text
    :::
```

Příklad obsahu CSS souboru:

```
    .nazevtridy {
        color: red;
    }
```

Dále můžeme využít integrovanýc “callout” bloků. Máme 5 typů těchto bloků: “note”, “tip”, “warning”, “important” a “caution”.

```
    ::: {.callout-note}
    Poznámka
    :::
```

::: {.callout-note}
Poznámka
:::

::: {.callout-tip}
Tip
:::

::: {.callout-warning}
Varování
:::

::: {.callout-important}
Důležité
:::

::: {.callout-caution}
Opatrně
:::

Chapter 5

Integrace zdrojových kódů

Jednou z hlavních výhod Quarto dokumentů je začlenění kusů kódu, které můžeme pouze vložit pro ukázku nebo si můžeme prohlédnout jejich výstup přímo v dokumentech. Nastavení Jupyter kernelu jsme si již představili v sekci Section 3.2. Nyní si představíme, jak můžeme kód vkládat do dokumentů.

5.1 Buňky s kódem

Kód vkládáme do buněk, které ohraničujeme ```. V případě, že chceme pouze vložit kód bez zvýraznění syntaxy, nemusíme nic doplňovat, avšak Quarto, díky Pandocu, podporuje zvýraznění syntaxy pro spoustu programovacích jazyků. Všechny naleznete zde: https://github.com/jgm/skylighting/tree/master/skylighting-core/xml

YAML kód bez specifikace:

````
```
title: "PCA"
```

title: "PCA"
````

YAML kód s specifikací:

````
```yaml
title: "PCA"
```
````

```yaml
   title: "PCA"
```

Pokud chceme i výstup kódu, vložíme jej do stejné buňky, ale specifikovaný programovací jazyk obalíme {}.

Příklad vytvoření buňky s kódem:

````
```{python}
print("Hello world!")
```
````

Výstup:

```
print("Hello world!")
```

```
Hello world!
```

5.2 Nastavení výstupu

Můžeme také upravovat výstup a chování buněk použitím YAML atributů.

Můžeme vytvořit globální podmínky v sekci execute:

```
    execute:
        echo: false
```

Nebo můžeme jednotlivá pravidla psát přímo dovnitř buněk:

````
```{python}
#| echo: false

print("Hello world!")
```
````

Výstup:

```
Hello world!
```

Dalšími užitečnými možnostmi nastavení výstupu jsou:

- `eval: false` - kód se nevykoná, pouze se zobrazí
- `warning: true` - zobrazí se i varovnání
- `error: true` - pokud se v kódu vyskytne chyba, render dokumentu bude pokračovat (jinak by se soubor nevyrenderoval)

5.3 Jednorázová exekuce buňky

Jednotlivé buňky můžeme během tvorby dokumentu spouštět samostatně. Toho docílíme pomocí klávesové zkratky `Ctrl+Enter`. Tímto způsobem si můžeme vyzkoušet, jak se bude kód vykonávat a jak bude vypadat výstup, bez potřeby renderovat celý dokument.

Chapter 6

Tvorba bibliografie

Pro citování bibliografie můžeme využít automatické generování citací. Jako zdrojové soubory pro bibliografii použijeme BibLaTeX (.bib), BibTeX (.bibtex) nebo podobné typy souborů. Tento zdroj specifikujeme pomocí YAML atributu "bibliography". Případně můžeme upravovat i vzhled citací pomocí CSL. Více podrobností naleznete zde: https://quarto.org/docs/authoring/footnotes-and-citations.html.

```
bibliography: references.bib
```

Příklad citece v BibLaTeX:

```
@online{Quarto,
    url={https://quarto.org/},
    journal={Quarto},
    publisher={Posit, PBC},
    author={Scheidegger, Carlos and Teague, Charles and Dervieux, Christophe and Alla
}
```

6.1 Vložení citací k textu

Citace vkládáme pomocí `@název`. Výstupem bude autor a rok vydání, pokud jsou tyto údaje specifikovány.

```
[@Quarto]
```

6.2 Vygenerování zdrojů na konci dokumentu

Bibliografie se vygeneruje automaticky na konci dokumentu. V případě prezentace se vytvoří nový slajd. Pokud chceme citace vypsat jinde, použijeme div s identifikátorem “refs” (viz. ukázka níže) nebo tento identifikátor přidáme rovnou k nadpisu.

```
::: {#refs}
:::
```

```
# Bibliografie {#refs}
```

Chapter 7

Publikování

Pro publikování našich dokumentů máme spostu možností. Pro různé dokumenty je vhodné použít různé způsoy publikace. Jelikož je základním formátem výstupu Quarto dokumentů HTML, můžeme tyto dokumenty zakomponovat přímo do zdrojového kódu webových stránek. Dalším ze základních formátů je PDF, což je velice podporovaný formát, který můžeme použít pro tisk nebo zaslání dokumentu.

Prezentace vytvořené pomocí RevealJS, lze vyexportovat do PDF, ale v případě, že v prezentacích máme zakomponavý kód, tato možnost není ideální. Pro export prezentací do PDF je vhodné použít webový prohlížeč. Prezentaci si nejprve vyrenderujeme do HTML a otevřeme ji v prohlížeči. Poté si přepneme prezentaci do "Print View" pomocí klávesy "E". Následně si otevřeme možnosti tisku (Ctrl+P) a nastavíme si výstup do PDF. Okraje nastavíme na "žádné" a povolíme "Grafiku pozadí".

Jelikož výstupním formátem RevelaJS prezentací je HTML, nabízí se možnost je publikovat na vlastních webových stránkách nebo na GitHub Pages.

7.1 GitHub Pages

GitHub Pages je služba, která umožňuje publikovat webové stránky přímo z GitHubu. Pro publikování na GitHub Pages musíme mít vytvořený Github účet. Následně si vytvoříme nový repozitář a nahrajeme do něj vyrenderovaný HTML kód. V nastavení repozitáře v sekci "Code and automation" najdeme možnost "Pages". Jako zdroj pro vygenreovnání (Source) zvolíme možnost "Deploy from a branch". Dále v části "Branch" zvolíme větev, ve které máme naši prezentaci a jako složku zvolíme "/(root)". GitHub Pages nyní budou hledat v naší složce soubor index.html a vytvoří z něho webovou stránku s doménou ve tvaru "https://jméno.github.io/název-repozitáře". Musíme tedy mít výstupní

soubor pojmenovaný “index.html”. (“Creating a GitHub Pages Site - GitHub Docs,” n.d.)

GitHub Pages a Quarto podporují i další automatizaci. Více informací naleznete zde: https://quarto.org/docs/publishing/github-pages.html.

Chapter 8

Závěr

Quarto je nový a velice jednoduchý nástroj pro tvorbu dokumentů. Díky jednoduchosti a přehlednosti je vhodný jak pro začátečníky, tak pro pokročilejší uživatele, na rozdíl například od LaTeXu, který může být pro začátečníky náročný. Hlavními výhodami může být tvorba všech možných typů dokumentů v jednom prostředí a možnost vkládání a exekuce kódu přímo do dokumentů. Jelikož je Quarto open-source, může se do vývoje zapojit kdokoliv a přidat tak další možnosti a funkce, které tento nástroj může nabídnout.

Bibliografie

"Creating a GitHub Pages Site - GitHub Docs." n.d. https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site.

Scheidegger, Carlos, Charles Teague, Christophe Dervieux, J. J. Allaire, and Yihui Xie. n.d. "Quarto." Posit, PBC. https://quarto.org/.