Markdown i R – zastosowanie i przykłady użycia

Marta Karaś, quantup.pl

2014-02-26

Spis treści

1 Markdown - CO TO?

Z nazwą "Markdown" spotkałam się po raz pierwszy przy tworzeniu dokumentacji do projektu stażowego, w którym brałam udział. Sposób użycia był jasny:

- Tworzymy tekst w prostej, czytelnej formie, przypominającej wiadomość email - formatowanie tekstu sprowadza się do kilku (dosłownie!) znaków, np. na oznaczenie cytatu (`> cytat`) czy linku (`[link]`).
- Piszemy w zwykłym edytorze tekstowym (np. Notepad) i zapisujemy plik z rozszerzeniem `.md` (od "Markdown").
- Na koniec, "dwoma kliknięciami" konwertujemy plik do dokumentu LaTeX, PDF, HTML czy jednego z wielu innych formatów, którego akurat potrzebujemy.
  - Z łatwością umieszczając w tekście: listę, cytat, fragment kodu, link, podział na paragrafy, a także - przy odrobinie uwagi więcej - równianie matematyczne, bibliografię, przypis, etc.
  - Bez konieczności pilnowania się z domykaniem html-owych tagów. Bez wstawiania dziesiątek instrukcji LaTeX przed napisaniem pierwszego słowa właściwego tekstu i kolejnych dziesiątek w środku.

Więc... czym jest Markdown? Z zamysłu twórcy, Johna Grubera, Markdown składa się z dwóch elementów:

- Po pierwsze, to składnia do formatowania zwykłego tekstu - maksymalnie prosta (o czym przekonamy się za chwilę), a ponadto sprawiająca, że na zwykły tekst "dobrze się patrzy" - listy wyglądają jak... listy, tuzin linków dodanych do jednego akapitu nie psuje czytelności treści itd.
- Po drugie, to narzędzie do konwertowania tak sformatowanego tekstu do HTML.

To... kiedyś. Dzisiaj drugie z powyższych zdań należałoby znacznie rozbudować - obecne zastosowania Markdown to dużo więcej, niż "konwersja do HTML".

2 Markdown - DO CZEGO?

Markdown pozwala na czytanie, pisanie i edytowanie tekstu, który ma być zamieszczony w internecie (np. jako HTML) lub - przy użyciu narzędzi takich jak konwertery online czy aplikacja Pandoc - przekonwertowany do większości innych formatów, które w tej chwili przyjdą nam do głowy. Przykładowo, przy użyciu Markdown można tworzyć:

- notatki i wiadomości email w czytelnej formie,
- dokumenty w formatach: procesorów tekstu (Microsoft Word docx, OpenOffice/LibreOffice ODT, OpenDocument XML), ebooków, formatów dokumentacji, formatów TeX, PDF, HTML i innych,
- prezentacje w formacie HTML.

PS: Ten artykuł został napisany w Markdown! Możesz zobaczyć plik źródłowy - jest zamieszczony na końcu artykułu razem z innymi przykładowymi dokumentami, w gotowej do pobrania paczce.

"Reproducible research": dlaczego Markdown jest ważnym narzędziem dla ludzi zajmujących się analizą danych? W dzisiejszym świecie za absolutny standard przy wykonywaniu analiz uważa się powtarzalność przeprowadzanych badań, a więc także powtarzalność ich wyników. W skrócie - chodzi o to, aby łączyć dane i kod analizy tak, aby inna grupa mogła je w całości odtworzyć (i uzyskać takie same rezultaty). O ile większość osób zajmujących się analizą rozumie lub przynajmniej słyszała, że praca "w duchu reproducible research" to rzecz ważna, to już znacznie mniejsza część zna i potrafi posługiwać się narzędziami, które do tego służą.

W środowisku do analiz statystycznych R, Markdown jest podstawowym narzędziem używanym przy prowadzeniu dynamicznych i powtarzalnych badań. Konkretnie, jest wykorzystywany do:

- łączenia analizy danych i jej wyników w jednym dokumencie (raporcie, prezentacji itp.).

3 Markdown - DLACZEGO?

Liczba użytkowników Markdown na świecie rośnie. Dlaczego warto poznać Markdown i dołączyć do tego grona?

Jest **wygodny**.

- Może być użyty do tworzenia dokumentów w różnych formatach (uniwersalność!), a jego składnia jest zdecydowanie prostsza, niż np. HTML czy LaTeX - stąd, pracując z Markdown unikamy błędów i oszczędzamy czas.
- Może być edytowany w dowolnym prostym edytorze obsługującym pliki tekstowe, na dowolnym systemie operacyjnym.
- Rozmiar pliku z rozszerzeniem `.md` jest możliwie mały.

Jest **powszechny**.

- Stał się podstawowym językiem formatowania treści na wielu ważnych stronach internetowych: Stack Overflow, GitHub, etc.
- Staje się standardem w pisaniu dokumentacji.

4 Markdown - JAK?

Poniżej przedstawimy, jak używać Markdown do wymienionych wyżej zastosowań:

1. pokażemy krótki tekst ze składnią Markdown, gotowy do konwersji w edytorze online,
2. napiszemy kod R z podręcznikowym przykładem prostej analizy danych, który wraz z wynikami umieścimy w jednym dokumencie Markdown; dokument ten przekonwertujemy do kilku innych formatów: LaTeX, PDF i HTML;
3. opiszemy, jak w kilka minut stworzyć z dokumentu Markdown prezentację HTML.

4.1 Pierwszy dokument w Markdown

Do stworzenia pliku w formacie Markdown wystarczy mieć do dyspozycji jeden z wielu edytorów online lub dowolny prosty edytor tekstowy.

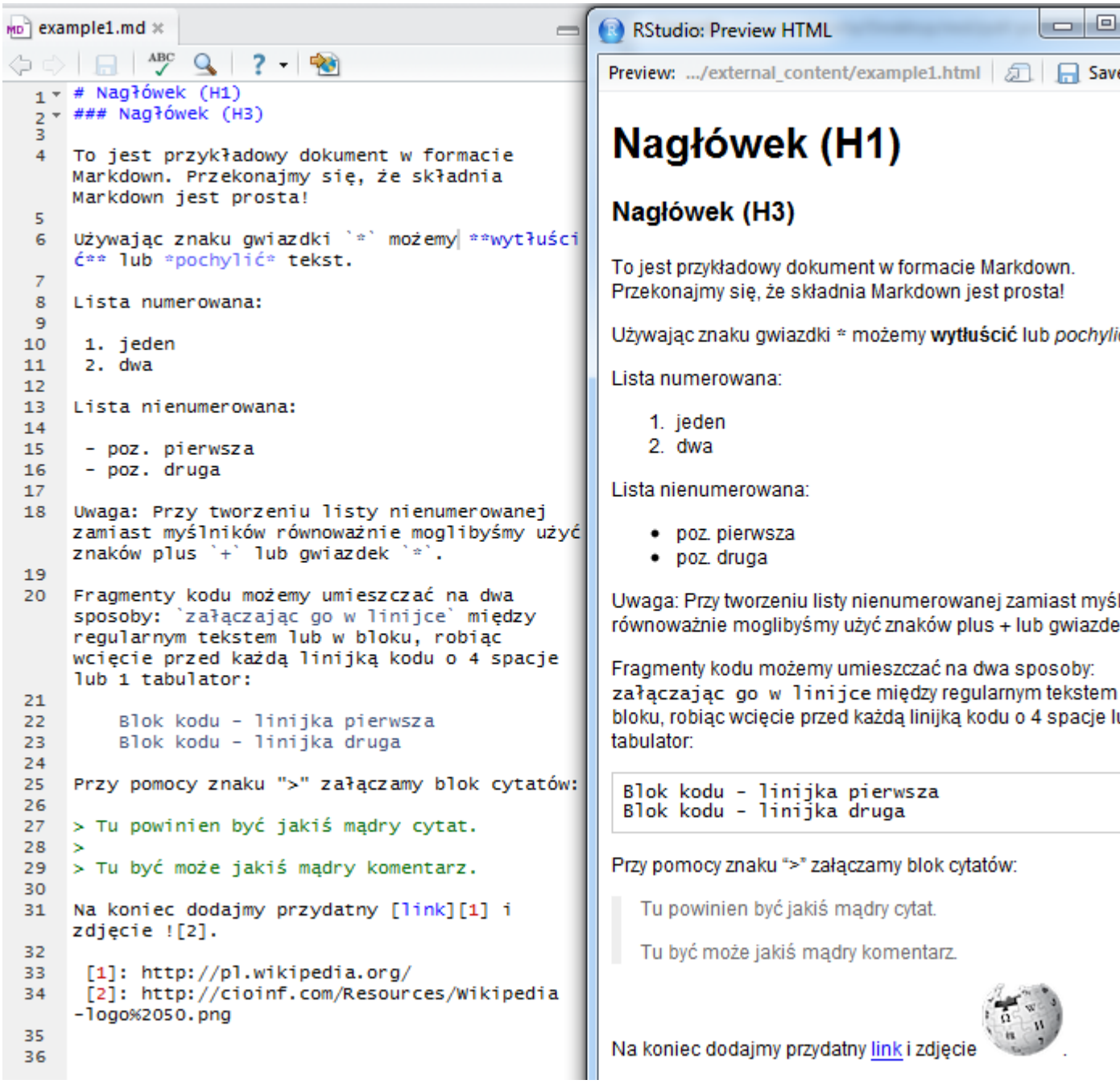
Składnia

Aby "załapać", na czym polega składnia Markdown, wystarczy spojrzeć na przykładowy dokument. Tekst formatujemy przy użyciu minimalnej ilości znaków. Większa część z nich użyta jest w przykładzie widocznym na rysunku poniżej.

Pełną informację o składni Markdown znajdziemy na tej stronie.

MultiMarkdown MultiMarkdown to jedno z wielu istniejących rozszerzeń składni Markdown. Pozwala na umieszczenie w tekście dodatkowych elementów, takich jak: przypis, tabela, bibliografia, metadata dokumentu (autor, tytuł itd.), użycie formuł matematycznych i więcej.

W sieci znajdziemy wiele zarówno skrótowych, jak i pełnych tutoriali dotyczących składni i użycia MultiMarkdown. Przykładowe dokumenty MultiMarkdown dostępne są m.in. w tej galerii.



Rysunek 1: Przykładowy dokument Markdown (po lewej stronie) i jego odpowiednik HTML.

Zapis dokumentu

Dokument tworzymy w prostym edytorze tekstu - np. Notepad pod Windowsem. Zapisujemy plik z rozszerzeniem `.md`. Nie używamy Microsoft Worda! (To procesor tekstu, a nie "prosty" edytor).

Narzędzie do konwersji: Pandoc

Pandoc jest programem typu command-line do konwersji dokumentów z jednego do drugiego formatu. Lista formatów obsługiwanych przez Pandoc jest długa. My użyjemy Pandoc do konwersji pliku Markdown do trzech popularnych formatów: LaTeX, PDF i HTML.

Zakładamy, że Pandoc jest zainstalowany na naszym komputerze (tutaj znajdziemy instrukcję do instalacji). Otwieramy wiersz poleceń i przechodzimy do lokalizacji, w której znajduje się czekający na konwersję plik. Do konwersji używamy poleceń:

- `pandoc example1.md -f markdown -t html -s -o example1.html`
  - konwersja do HTML (output)
- `pandoc example1.md -f markdown -t latex -s -o example1.tex`
  - konwersja do TeX (output)
- `pandoc example1.md -f markdown -t latex -s -o example1.pdf`
  - konwersja do PDF (output - śliczny!)

(Zachęcam do samodzielnego wypróbowania powyższych komend, na przykład na tym pliku Markdown.)

W powyższych poleceniach wskazujemy nazwę pliku, który będzie poddany konwersji (`example1.md`), z jakiego (`-f`) i do jakiego (`-t`) formatu konwersja ma się odbyć, określamy (`-s`), że Pandoc ma dodać do wyjściowego dokumentu odpowiedni nagłówek i stopkę (co jest konieczne, aby w wyniku postał odrębny plik) i wskazujemy (`-o ...`), gdzie zapisać output.

Szczegółowy przewodnik po opcjach konwersji Pandoc znajdziemy w tej zakładce.

Narzędzie do konwersji: edytor online

W sieci dostępnych jest wiele bezpłatnych edytorów Markdown - większość z nich ma wbudowane funkcje pobrania utworzonego tekstu nie tylko w formacie `.md`, ale i innych (HTML, TeX, PDF etc). Godnymi polecenia są:

- Dillinger - synchronizacja z Dropboxem, GitHubem i Google Drive, eksport do Markdown, HTML i PDF, synchronizacja scrollowania tekstu i podglądu HTML, możliwość przystosowania wyglądu edytora,
- Stack Edit - synchronizacja z Dropboxem i Google Drive, bezpośrednie udostępnianie na WordPress, Tumblr, Blogger i inne portale, eksport do Markdown, HTML i PDF, synchronizacja scrollowania tekstu i podglądu HTML, nowoczesny design.

Uwaga: żaden z powyższych edytorów nie stosuje domyślnie systemu kodowania UTF-8 (w przeciwieństwie do Pandoc), więc konieczne (jeśli używamy polskich znaków) jest dodanie następującego fragmentu HTML do tekstu Markdown:

```
<meta charset="utf-8">
```

Inne narzędzia

Do dyspozycji jest szeroki wybór narzędzi do konwersji z Markdown do innych formatów, do pobrania za darmo z sieci. W R, konwersja z Markdown do HTML możliwa jest przy użyciu pakietu `knitr`.

4.2 Markdown w R.

Dynamiczne raporty: R Markdown + knitr

R Markdown to format pliku, w którym oprócz standardowego tekstu Markdown możemy umieszczać bloki kodu R. Przy użyciu R-owego pakietu knitr kod ten jest ewaluowany i umieszczany wraz z uzyskanym wynikiem (wykres, tabela itp.) w dokumencie Markdown lub od razu w pliku HTML, jeśli mamy takie życzenie.

- Dodatkowo, pakiet `knitr` posiada funkcję `pandoc()`, która wykonuje polecenia znanej nam już aplikacji Pandoc; dzięki niej, otrzymany dokument Markdown możemy poddać konwersji do innych formatów, bezpośrednio z R.

- Domyślne ustawienia sposobu prezentacji naszej analizy (kodu i jej wyników) możemy modyfikować - przykładowo, możemy chcieć ukryć kod R, pokazując w końcowym dokumencie same wyniki. Do definiowania tych ustawień służą tzw. chunks - będą omówione w przykładzie poniżej.

Podsumowując, R Markdown daje nam możliwość łatwego budowania raportów, które mogą być automatycznie odświeżane, gdy zmienią się dane użyte w naszej

analizie lub gdy wprowadzimy modyfikacje w użytym kodzie R. Możemy otrzymać wynikowy dokument od razu w formacie HTML - gotowy do publikacji w sieci.

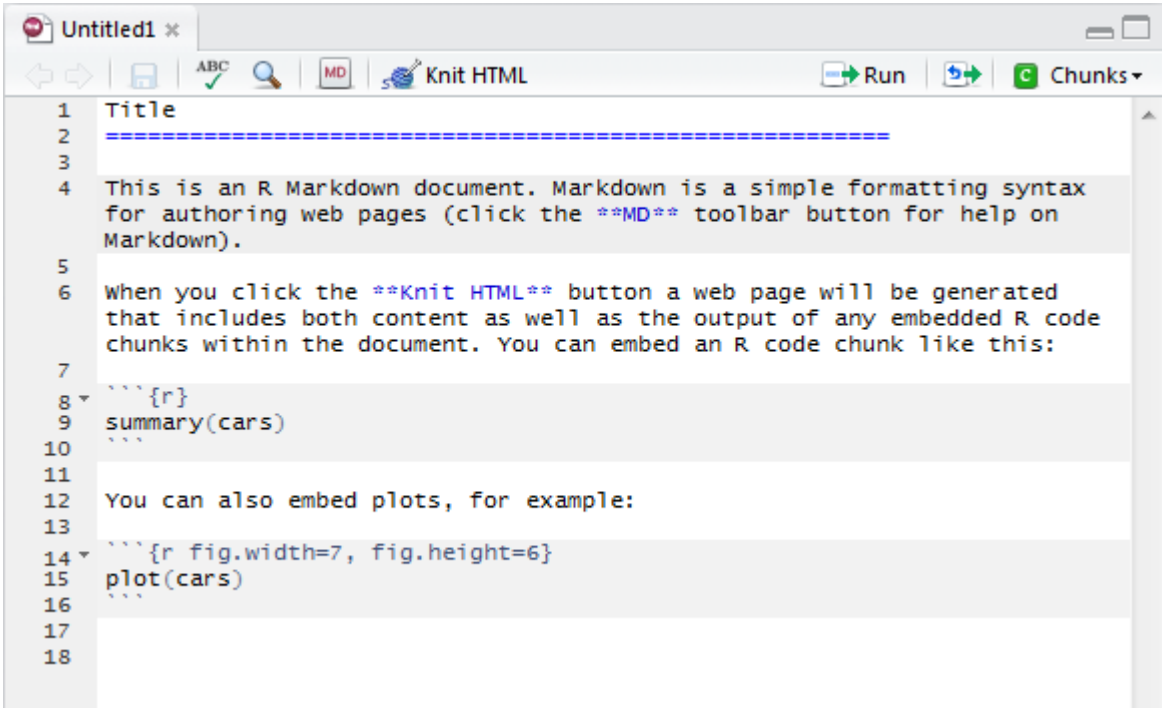
Krok po kroku: prosty raport

Poniżej przedstawiamy prosty przykład, jak wykonać konwersję z R Markdown do HTML przy użyciu pakietu `knitr`. Zakładamy, że mamy zainstalowane R i RStudio.

1. W RStudio instalujemy i załączamy pakiet `knitr`.

```
install.packages("knitr")
library(knitr)
```

2. Otwieramy nowy plik R Markdown: **File -> New -> R Markdown**.



Rysunek 2: Nowy plik Markdown po otwarciu w RStudio.

3. Budujemy "nasz" plik:

- Regularny tekst formatujemy zgodnie ze składnią Markdown.
- Kod R umieszczamy w blokach - tzw. chunks:

  ```` ```{r chunkLabel, chunkOptions} #kod ``` ````

W blokach chunks definiujemy ustawienia (chunk options), według których nasza analiza będzie przedstawiona w wynikowym dokumencie. Konkretnie, ustalamy szczegóły dotyczące sposobu prezentacji kodu, wiadomości z linii komend R pojawiających się przy ewaluacji kodu, otrzymanych wykresów, tabel etc.

Opcje te definiujemy na dwa sposoby: lokalnie (tj. dla konkretnego bloku chunk), uzupełniając odpowiednio pole między wąsami `{...}`, lub globalnie (narzucamy je dla całego dokumentu), używając odpowiedniej funkcji pakietu `knitr` wewnątrz bloku chunk.

Poniżej widzimy fragment dokumentu, gdzie chunk options definiowane są i lokalnie, i globalnie. (Zwróćmy uwagę na praktyczną informację, umieszczoną w tekście! (linijka 22.)).

````
7
8  ```{r settings, echo=FALSE, message=FALSE}
9
10 # LOKALNIE: Ustalamy dla tego bloku kodu, że w wynikowym dokumencie nie
   będzie wyświetlany ani ten kod (echo=FALSE), ani wiadomości pojawiające
   się w konsoli R w czasie jego ewaluacji (message=FALSE).
11
12 require(png)
13 require(XML)
14 require(RCurl)
15
16 options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem",
   package = "RCurl")))
17
18 # GLOBALNIE: ustalamy opcje konwersji (opts_knit$set(...)) dla całego
   dokumentu.
19 opts_knit$set(upload.fun = imgur_upload)
20 ```
21
22 Uwaga: w tym dokumencie zdefiniowaliśmy opcję `opts_knit$set(upload.fun =
   imgur_upload)` - obrazy powstające z ewaluacji kodu R będą uploadowane na
   portal imgur.com. W wynikowym pliku Markdown (lub HTML) zostaną zamieszcz
   one przez linki do imgur, dzięki czemu plik będzie kompletny i samowystar
   czalny.
23
````

Rysunek 3: Użycie chunk options - przykład.

(Dla zainteresowanych: przykłady najczęściej używanych chunk options przedstawione są tutaj. Pełne opracowanie dostępne jest na stronie knitr.)
````

4. Konwertujemy plik R Markdown do formatu HTML przy użyciu pakietu `knitr`. Możemy to zrobić na kilka sposobów (przy każdym uprzednio zapisując plik):

- klikając przycisk **Knit HTML** w RStudio,
- wywołując konwersję z R Markdown do Markdown i z Markdown do HTML:

```
require(knitr)
require(markdown)

# konwersja z .rmd do.md
knit(moja_analiza.rmd", "moja_analiza.md")
# konwersja .md do .html
markdownToHTML("moja_analiza.md", "moja_analiza.html")

browseURL(paste("file://",
file.path(getwd(), "moja_analiza.html"), sep=""))
```

- wywołując konwersję bezpośrednio z R Markdown do HTML:

```
require(knitr)
knit2html("moja_analiza.rmd", "moja_analiza.html")
```

Raport gotowy! (Zobacz użyty plik R Markdown oraz wynikowy HTML).

Ten sam raport, ale w innym formacie: funkcja pandoc()

Funkcja `pandoc()` pakietu `knitr` pozwala na wykonanie z poziomu R wszystkich tych konwersji dokumentu, które umożliwia nam aplikacja Pandoc. Możemy więc najpierw użyć funkcji `knit`, by otrzymać z R Markdown plik Markdown, a następnie `pandoc()`, by dostać plik w pożądanym przez nas formacie.

- Dodatkowo, korzystając z funkcji `pandoc()` możemy zdefiniować opcje konwersji w specjalnym pliku konfiguracyjnym lub zamieścić je w samym pliku Markdown - `knitr` parsuje ten plik konfiguracyjny (/wbudowane konfiguracje) i zamienia w argumenty używane przez aplikację Pandoc. Tym sposobem skracamy długość polecenia, które musielibyśmy wprowadzić w linii komend przy standardowym użyciu aplikacji Pandoc - pozwala to pewną automatyzację przy wielokrotnym konwertowaniu tego samego dokumentu.

Przykłady użycia funkcji `pandoc()` pakietu `knitr` są pokazane na stronie stronie Yihui Xie (nota bene autora pakietu `knitr`). Aby wykonać konwersję pliku Markdown do HTML, autor tworzy następujący plik konfiguracyjny `foo.txt`:

```
t: html
s: <empty>
mathjax: <empty>
number-sections: <empty>
bibliography: foo.bib
o: output.html
```

i używa go, wywołując funkcję `pandoc()`;

```
library(knitr)
pandoc('input.md', format='html', config='foo.txt')
```

Po pełen opis opcji Pandoc należy zajrzeć bezpośrednio na stronę Pandoc - jest długi, co może być zniechęcające. Dla użytkowników nieczujących potrzeby definiowania zaawansowanych opcji konwersji, do dyspozycji pozostaje trywialny sposób użycia:

```
require(knitr)
pandoc('foo.md', format='html')
```

4.3 Prezentacje HTML5 z R Markdown

Błyskawiczne generowanie prezentacji HTML to - zdecydowanie! - moje ulubione zastosowanie Markdown. O jakich prezentacjach mowa?

- Prezentacjach z wykorzystaniem równań matematycznych, rysunków, linków, z wyróżnioną składnią kodu etc.
- Prezentacjach, w których możemy umieścić wyniki analizy zrobionej w R i w razie potrzeby odświeżyć je "dwoma kliknięciami".
- Prezentacjach, których całe wykonanie zajmuje niekiedy mniej, niż napisanie jednego "działającego" slajdu w Beamerze.
- Prezentacjach o nowoczesnym designie, który zachwyci odbiorców (nawet, gdy nie zrobi tego treść ;-)).

Jak budować prezentacje? Ja, w zależności od potrzeb, używam jednego z dwóch narzędzi: aplikacji Pandoc lub pakietu Slidify.

- Użycie aplikacji Pandoc (równoważnie użyciu funkcji `pandoc()` w R) jest generalnie wygodniejsze i stosuję je, gdy prezentację robię z dokumentu Markdown (w szczególności: gdy nie zamieszczam w prezentacji kodu R).
- `slidify` tworzy slajdy zarówno z "czystego" Markdown, jak i bezpośrednio z R Markdown, w którym (jak wspomniano wyżej) mogę zamieścić kod R. `slidify` automatycznie wykona ewaluację tego kodu (np. analizy danych) i umieści (wraz z ewentualnymi wynikami, np. wykresem) w prezentacji.

Poniższe przykłady pokazują jak - krok po kroku - stworzyć prezentację HTML przy użyciu obu tych narzędzi.

Krok po kroku: prezentacja z wykorzystaniem Slidify

Aby stworzyć prezentację:

1. Pobieramy i dołączamy niezbędne biblioteki :

```
library(devtools)
install_github('slidify', 'ramnathv')
install_github('slidifyLibraries', 'ramnathv')

library(slidify)
```

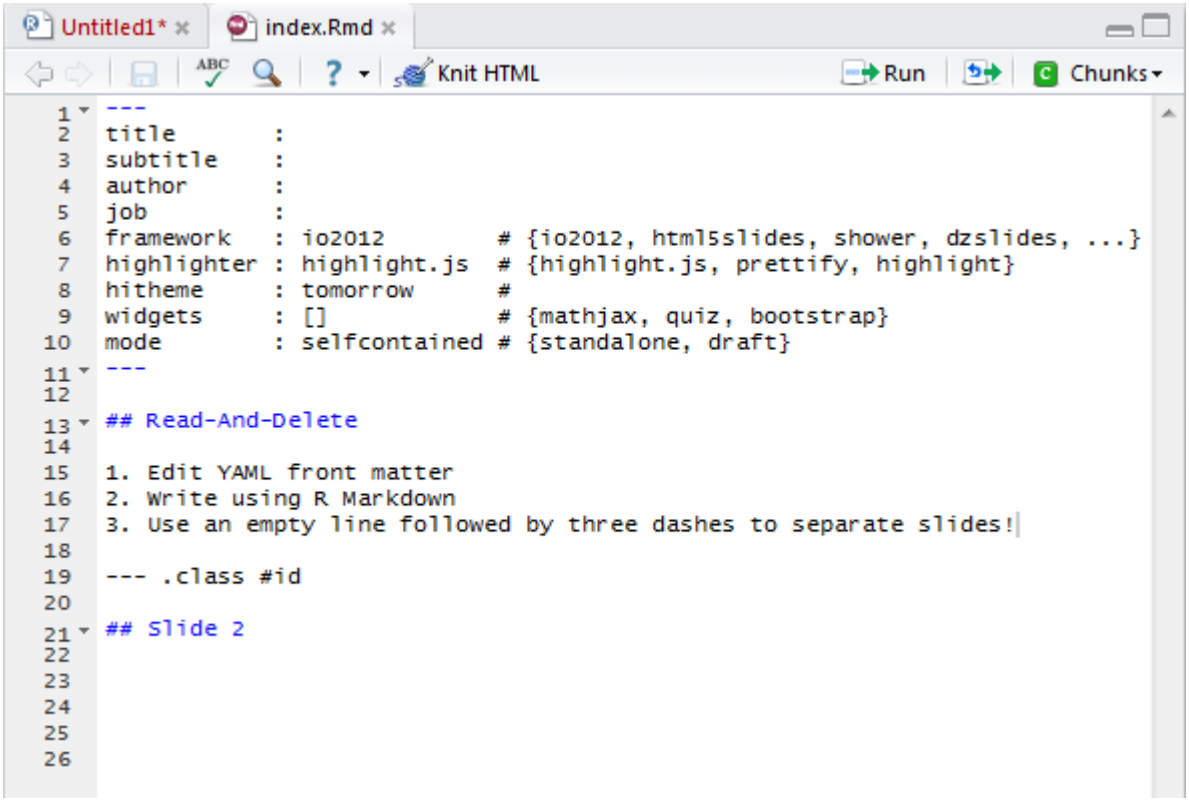
2. Używamy funkcji `author()`:

```
author("mydeck")
```

Funkcja tworzy folder `mydeck` w naszym katalogu roboczym i kopiuje tam "rusztowanie" prezentacji, w tym plik `index.Rmd` - widok pliku na rysunku poniżej.

3. Plik `index.Rmd` to "ciało" naszej prezentacji - uzupełniamy je, pamiętając, że:

- poszczególne slajdy oddzielone są pustą linią (blank line), po której następuje symbol `---`;
- zawartość slajdów wypełniamy tym, czym możemy wypełnić "zwykły" dokument R Markdown - regularnym tekstem sformatowanym zgodnie ze składnią Markdown oraz kodem R, umieszczonym w odpowiednio zdefiniowanych blokach (tzw. chunks);



Rysunek 4: Plik index.Rmd.

- zawartość między pierwszymi dwoma znacznikami `---` jest szczególna - to tzw. metadata, w której definiujemy dane widniejące na slajdzie tytułowym (należy uzupełnić puste pole po oznaczeniach: `title :`, `subtitle :`, `author :`, `job :`) oraz inne ustawienia prezentacji, takie jak wybór jednego z kilku dostępnych `framework`-ów (do generowania slajdów) czy `highlighter`-a (biblioteki służącej do wyświetlania kodu w czytelny sposób).

4. Wykonujemy konwersję do slajdów HTML:

```
slidify("./mydeck/index.Rmd")
```

W jej wyniku tworzą się m.in. dwa nowe dokumenty: `index.md` i `index.html` - nasza prezentacja.

Oczywiście, przy użyciu funkcji `slidify()` możemy tworzyć slajdy HTML ze zwykłych plików Markdown. Gdyby chcieć wykorzystać do tego plik `index.md`, wywołanie funkcji wyglądałoby następująco:

```
slidify("./mydeck/index.md")
```

Uwaga: W tak utworzonych prezentacjach (przy domyślnym frameworku `io2012`), na slajdzie tytułowym polskie znaki diakrytyczne są niepoprawnie wyświetlane (niespójna czcionka - przykład). Dzieje się tak dlatego, że funkcja `slidify()` korzysta z pliku `.css`, w którym zdefiniowane jest użycie na stronie tytułowej czcionek, które nie posiadają polskich znaków. Plik ten powstaje przy pierwszym użyciu funkcji `slidify()`, pod adresem:

`.\mydeck\libraries\frameworks\io2012\css\slidify.css`.

Proste i skuteczne rozwiązanie problemu polega na zmianie kilku linijek we wspomnianym pliku `.css` i jest opisane w tym poście.

Krok po kroku: prezentacja z wykorzystaniem Pandoc

Zauważmy, że przy tworzeniu slajdów HTML z pliku R Markdown (czy Markdown) przy użyciu `Slidify`, niezbędne było użycie specjalnych znaczników (pusta linijka plus znacznik `---`) - jest to pewna uciążliwość. Rozwiązaniem, które pozwala na budowanie prezentacji z pliku Markdown bez używania specjalnych znaczników do oddzielenia slajdów, jest Pandoc.

- Pandoc umożliwia konwersję pliku Markdown do slajdów przy użyciu różnych skryptów: `Slidy`, `reveal.js`, `Slideous`, `S5`, `DZSlides`.

- Wydzielanie slajdów jest proste. Start nowego slajdu jest oznaczony przez wystąpienie nagłówka (#, ##, ### ...) z najwyższego poziomu hierarchii wszystkich nagłówków występujących w dokumencie (gdy bezpośrednio po tym nagłówku występuje zawartość slajdu, a nie np. inny nagłówek z tego poziomu). Oznacza to, że jeśli w dokumencie Markdown używamy znaków ##, ### i #### do oznaczenia nagłówków, to znak ## stanie się oznaczeniem dla nowego slajdu. (Więcej informacji na ten temat znajduje się na tej stronie w sekcji Structuring the slide show.)

Według mnie, zadowalający wynik (tj. powstanie dobrze wyglądających slajdów) daje użycie dwóch skryptów: `Slidy` i `DZSlides`.

Jak tworzymy slajdy?

1. Instalujemy Pandoc (jeśli jeszcze go nie mamy).

2. Konwertujemy dokument Markdown do slajdów HTML (tu: wcześniej wykonujemy także konwersję z R Markdown do Markdown aby pokazać, jak wykresy R są umieszczane w prezentacjach):

```
library(knitr)

# konwersja do formatu Markdown
knit("pandoc_test_slidy.Rmd")
knit("pandoc_test_dzslides.Rmd")

# konwersja do slajdów
pandoc("pandoc_test_slidy.md", format = 'slidy')
pandoc("pandoc_test_dzslides.md", format = 'dzslides')
```

Uwaga: należy pamiętać, aby zdefiniować opcje konwersji Pandoc - robimy to za pomocą osobnego pliku konfiguracyjnego lub wewnątrz pliku Markdown, który jest używany w funkcji `pandoc()`. W przykładzie powyżej użyty został drugi sposób - opcje konfiguracji są umieszczone w plikach `pandoc_test_slidy.md` i `pandoc_test_dzslides.md`. (W naszym przypadku zostały zdefiniowane jeszcze przed konwersją do Markdown, tj. w plikach pandoc_test_slidy.Rmd i pandoc_test_dzslides.Rmd, odpowiednio.) Konfiguracje Pandoc są widoczne na początku każdego z tych dwóch plików, pomiędzy `<!--pandoc` i `-->`.

Slajdy są gotowe!