KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

IBE entuzjaści edukacji

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Monitorowanie losów absolwentów uczelni
z wykorzystaniem danych administracyjnych
Zakładu Ubezpieczeń Społecznych

Dokumentacja użytkownika

Pakietu do generowania raportów MLAK

Raport przygotowany przez Pracownię Ewaluacji Jakości Kształcenia na Uniwersytecie Warszawskim w ramach projektu systemowego Badanie jakości i efektywności edukacji oraz instytucjonalizacja zaplecza badawczego, współfinansowanego przez Unię Europejską ze środków Europejskiego Funduszu Społecznego, realizowanego przez Instytut Badań Edukacyjnych

Warszawa, 21 Kwietnia 2015

Autor:
Mateusz Żółtak

Wydawca:
Instytut Badań Edukacyjnych
ul. Górczewska 8
01-180 Warszawa
tel. (22) 241 71 00
www.ibe.edu.pl

Publikacja opracowana w ramach projektu systemowego „Badanie jakości i efektywności edukacji oraz instytucjonalizacja zaplecza badawczego”, współfinansowanego przez Unię Europejską ze środków Europejskiego Funduszu Społecznego, realizowanego przez Instytut Badań Edukacyjnych.

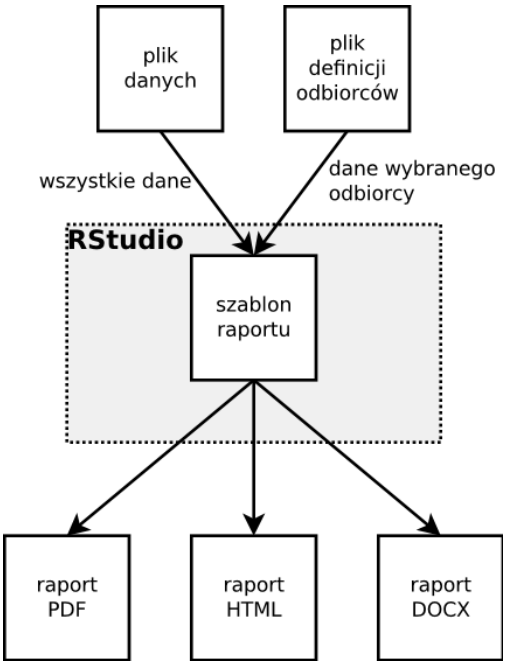
Spis Treści

1. Wstęp 5
2. Instalacja 6
2.1. Korzystanie z programu RStudio 7
Tworzenie nowego szablonu raportu 8
Zapisywanie pliku szablonu 9
Otwieranie zapisanego na dysku pliku szablonu 9
Generowanie raportu na podstawie szablonu 9
Zmiana opcji eksportu raportu do danego formatu 10
3. Plik danych 10
3.1. Formaty danych 10
3.2. Ograniczenia nazw zmiennych (kolumn) 11
3.3. Wczytywanie pliku danych w szablonie raportu 11
4. Plik definicji odbiorców 11
4.1. Formaty danych 12
4.2. Wczytywanie pliku danych odbiorcy w szablonie raportu 12
4.3. Wyświetlanie w raporcie wartości z pliku definicji odbiorców 12
5. Szablon raportu 13
5.1. Formatowanie tekstu 14
Przykład 15
5.2. Obliczanie statystyk 16
Przykład 18
5.3. Obliczanie statystyk dla podgrup 19
Przykład 20
5.4. Tabele 21
Pipe tables 21
Multiline tables 22
Przykład 23

5.5. Wykresy 24
Przykład 25
5.6. Zawartość warunkowa 28
Przykład 29
6. Automatyczne generowanie wielu raportów 30
6.1. Przykład 31
7. Interaktywne raporty WWW 32
7.1. Szablon formularza 32
Funkcja inputPanel() 34
Funkcja renderUI() 35
7.2. Podsumowanie 36

1. Wstęp

Przed przystąpieniem do korzystania z pakietu *MLAK* warto zapoznać się z ogólnym przebiegiem procesu generowania raportów (rysunek 1).



Rysunek 1. Przebieg generowania raportów z użyciem pakietu *MLAK*

Raport generowany jest na podstawie:

- **Pliku danych**, który zawiera informacje o jednostkach analizy (typowo: studentach). Na podstawie tych danych wyliczane są w raporcie statystyki określone w szablonie raportu. Plik ten opisano dokładnie w rozdziale 3.

- **Pliku definicji odbiorców**, zawierającego informacje o poszczególnych odbiorcach raportów (nazwie itp.) oraz wszelkie identyfikatory potrzebne do dopasowania w pliku danych rekordów właściwych dla danego odbiorcy (np. identyfikator kierunku studiów, itp.). Plik ten opisano dokładnie w rozdziale 4.

- **Szablonu raportu**, który opisuje niezmienną treść raportu, formatowanie tekstu, wyliczane statystyki i rysowane wykresy, nie zawiera jednak żadnych danych ani wyliczonych wartości. Tworzenie szablonów raportów, jak również sposób generowania pojedynczego raportu opisano w rozdziale 5.

Przygotowanie plików danych i definicji odbiorców odbywać się może w dowolnym programie, do edycji szablonu raportu i generowania raportów wykorzystywany jest natomiast program *RStudio*. Generowanie raportów polega na wypełnieniu szablonu wartościami obliczonymi na podstawie pliku danych i/lub odczytanymi z pliku definicji odbiorców i zapisaniu raportu w określonym formacie wyjściowym (PDF, HTML lub DOCX) na dysku.

Rozdzielenie szablonu raportu, danych oraz opisu odbiorców pozwala na:

- Korzystanie z tego samego szablonu niezależnie od danych i listy odbiorców. Np. żeby w kolejnym roku wygenerować ten sam raport dla kolejnego rocznika studentów, nie trzeba dokonywać żadnych zmian w szablonie raportu, a jedynie podmienić plik danych.

- Korzystanie z tego samego szablonu i pliku danych niezależnie od listy odbiorców, dla jakich mają być wygenerowane raporty, tzn. zmiana listy odbiorców, nie powoduje konieczności dokonywania modyfikacji w zbiorze danych ani szablonie raportu.

Przedstawiony na rysunku 1 proces może zostać automatycznie powtórzony dla wszystkich odbiorców opisanych w pliku definicji odbiorców (patrz rozdział 6).

Istnieje jeszcze wariant, w którym raport prezentowany jest jako interaktywna strona WWW. W takim wypadku dane z pliku definicji odbiorców zastępowane są przez wartości wybrane przez użytkownika na stronie WWW i na tej podstawie generowany jest odpowiedni raport w formacie HTML (patrz rozdział 7).

2. Instalacja

Aby móc generować raporty z użyciem *MLAK*, niezbędne jest zainstalowanie wymienionych poniżej programów. Wszystkie potrzebne programy są darmowe oraz dostępne zarówno dla Windows, MacOS, jak i Linuxa.

- *R* dostępny jest pod adresem:
    - dla Windows http://r.meteo.uni.wroc.pl/bin/windows/base/
    - dla MacOs http://r.meteo.uni.wroc.pl/bin/macosx/
    - dla Linuxa http://r.meteo.uni.wroc.pl/bin/linux/ lub w paczkach poszczególnych dystrybucji.
- *RStudio* dostępny jest pod adresem http://www.rstudio.com/products/rstudio/download/ (dla wszystkich platform).
- *Pandoc* dostępny jest pod adresem https://github.com/jgm/pandoc/releases#Downloads (dla wszystkich platform, pod Linuxem także w paczkach poszczególnych dystrybucji).
- Dowolna dystrybucja LaTeX-a, np.:
    - pod Windows MiKTeX – http://miktex.org/download
    - pod MacOs MacTeX – https://tug.org/mactex/
    - pod Linuxem TeX Live – dostępny w paczkach dystrybucji.

Dodatkowo wymagane jest zainstalowanie pakietu *R devtools* oraz samego pakietu *MLAK*. W tym celu w konsoli programu *RStudio* (patrz następny rozdział – opis lewej szpalty) wpisujemy komendy:

```
install.packages('devtools')
devtools::install_github('zozlak/MLAK')
```

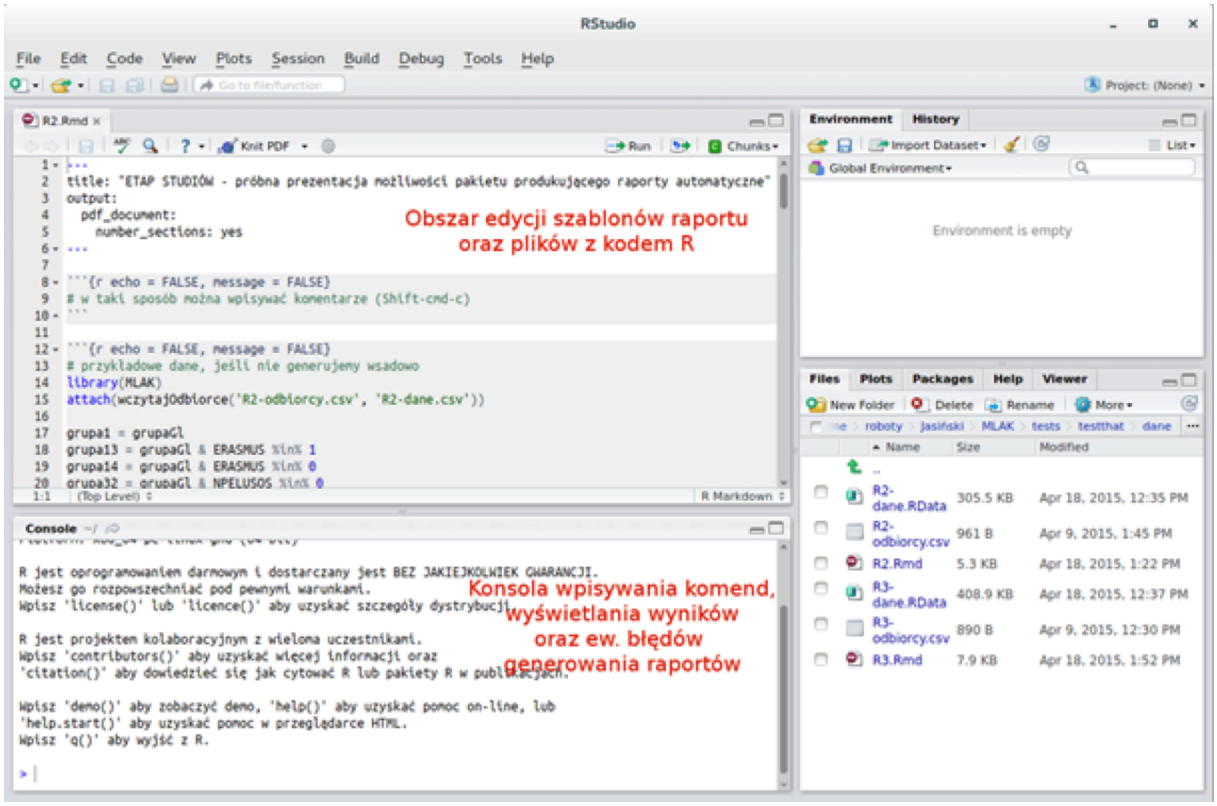
Uwagi:

- Jeśli użytkownik ma już zainstalowane na komputerze programy *R, RStudio* czy *Pandoc*, warto upewnić się, czy są one w aktualnych wersjach, a jeśli nie, dokonać aktualizacji.

- Aby uniknąć problemów z brakującymi czcionkami dystrybucji LaTeX-a, najlepiej dokonać instalacji kompletnej dystrybucji. Wymaga to 2 do 3 GB wolnego miejsca na dysku, ale uwalnia od konieczności ręcznego znajdowania i instalowania ewentualnych brakujących pakietów LaTeX-a.

2.1. Korzystanie z programu *RStudio*

Korzystanie z pakietu *MLAK* jest możliwe w dowolnym środowisku programistycznym języka *R*, np. w interpreterze uruchamianym w linii komend, środowisku graficznym instalowanym razem z programem *R* i innych. Najlepszym środowiskiem programistycznym do korzystania z pakietu *MLAK* wydaje się jednak *RStudio*, gdyż zapewnia ono bardzo dobrą integrację z funkcjonalnością pakietu *rmarkdown*, np. umożliwia wygenerowanie raportu z szablonu jednym kliknięciem, automatycznie formatuje kod *R* i *markdown* itp. Z uwagi na to w niniejszym podręczniku **zakłada się, że użytkownik będzie korzystać właśnie z *RStudio***.



Rysunek 2. Okno programu *RStudio* z zaznaczonymi dwoma obszarami najważniejszymi podczas pracy z pakietem *MLAK*

Okno programu RStudio (rysunek 2) podzielone jest na dwie szpalty:

- Lewą:

- W jej górnej części odbywa się edycja szablonów raportu (rozdział 5) i plików z kodem *R* (np. rozdziały 6 i 7).

- W dolnej części znajduje się konsola do wprowadzania komend w języku *R* (wykorzystujemy ją np. do instalacji pakietów – patrz instrukcja instalacji), wyświetlania wyników ich działania, jak również ewentualnych błędów generowania raportu.

Podczas przygotowywania raportów z użyciem pakietu *MLAK* przez zdecydowaną większość czasu będziemy korzystać właśnie z tej szpalty.

Uwaga! Przy pierwszym uruchomieniu *RStudio* (jak również w wypadku, gdy zamkniemy wszystkie otwarte pliki szablonów raportów, kodu *R* i danych) widoczna jest tylko dolna część lewej szpalty (konsola). Górna część pojawia się w momencie otwarcia bądź utworzenia nowego pliku szablonu raportu lub kodu *R*.

- Prawą, która również podzielona jest na dwie części:

  - Górną, w której wyświetlana jest lista aktualnie wczytanych obiektów (zakładka Environment) oraz historia komend wydanych w konsoli (zakładka History).

  - Dolną, w której wyświetlane są:

    - lista plików (zakładka Files);

    - wygenerowane poprzez ręczne wykonywanie komend wykresy (zakładka Plots);

    - lista dostępnych pakietów *R* (zakładka Packages);

    - przeglądarka plików pomocy (zakładka Help).

  Z zawartości prawej szpalty będziemy korzystać znacznie rzadziej, ale może być ona przydatna np. do:

  - przeglądania stron pomocy (aby wyszukać pomoc dla danej funkcji, wystarczy wpisać fragment jej nazwy w polu wyszukiwania w zakładce *Help*);

  - sprawdzania efektów wykonania *wstawek R* umieszczonych w szablonie raportu bez generowania całego raportu (zakładki *Environment* czy *Plots*).

Tworzenie nowego szablonu raportu

Aby utworzyć nowy szablon raportu, należy:

- wybrać menu *File* → *New File* → *R Markdown…*,

- w wyświetlonym okienku wybrać *Document*, podać nazwę oraz, opcjonalnie, autora raportu, po czym zatwierdzić wybór przyciskiem *OK* (wybór formatu wyjściowego spośród HTML/PDF/DOCX nie ma na tym etapie znaczenia – można go potem zmienić w każdej chwili),

- wybrać menu *File* → *Save* (lub skorzystać z kombinacji klawiszy *CTRL+S*), aby zapisać szablon raportu na dysku.

  Uwaga! Szablon najlepiej zapisać w pliku z rozszerzeniem *.Rmd* – wtedy zostanie on automatycznie skojarzony z programem *RStudio*.

W wyniku utworzenia nowego szablonu raportu zostanie on otwarty w górnej części lewej szpalty programu *RStudio*. Nowy szablon zawsze zawiera krótką przykładową treść, można więc od razu spróbować wygenerować raport na jego podstawie (patrz niżej).

Zapisywanie pliku szablonu

Zapis pliku szablonu jest możliwy poprzez:

- kombinację klawiszy *CTRL+S*,
- menu *File* → *Save*,
- kliknięcie ikonki dyskietki w lewej części paska znajdującego się nad polem edycji szablonu.

Uwaga! Szablon raportu jest też automatycznie zapisywany przy każdym generowaniu raportu.

Otwieranie zapisanego na dysku pliku szablonu

- Jeśli szablon raportu został zapisany w pliku z rozszerzeniem *.Rmd*, wystarczy kliknąć na nim dwukrotnie – zostanie automatycznie otwarty w programie *RStudio*.
- Otwarcie szablonu jest również możliwe z poziomu programu *RStudio* – w tym celu należy wybrać menu *File* → *Open File*…, a następnie wskazać plik szablonu na dysku.

Uwaga! *RStudio* zapamiętuje ostatnio edytowane pliki i automatycznie otwiera je przy ponownym uruchomieniu, często więc ręczne otwieranie szablonu raportu nie jest w ogóle potrzebne – wystarczy uruchomić *RStudio*.

Generowanie raportu na podstawie szablonu

Aby wygenerować raport na podstawie szablonu, należy:

- otworzyć plik szablonu w programie *RStudio*,
- kliknąć przycisk *Knit HTML* (ew. *Knit PDF* lub *Knit Word*) w pasku nad obszarem edycji szablonu raportu.

Po chwili oczekiwania raport zostanie wygenerowany i otwarty w odpowiednim programie.

Uwaga! Jeśli chcemy zmienić format generowanego raportu, należy kliknąć na małą strzałkę w prawym rogu przycisku *Knit HTML* (ew. *Knit PDF* lub *Knit Word*) i wybrać pożądany format.

Uwaga! Jeśli szablon raportu zawierał błędy, które uniemożliwiły wygenerowanie raportu, komunikaty informujące o tych błędach zostaną wyświetlone w dolnej części lewej szpalty programu *RStudio* (w zakładce *R Markdown*).

Zmiana opcji eksportu raportu do danego formatu

Aby zmienić opcje eksportu raportu do danego formatu (np. ustalić domyślny rozmiar wykresów, wybrać, czy nagłówki powinny być numerowane, itp.), należy:

- otworzyć szablon raportu w programie *RStudio*,
- kliknąć na ikonkę zębatki w pasku nad polem edycji szablonu raportu,
- w liście rozwijalnej w górnej części wyświetlonego okienka wybrać interesujący format,
- ustawić opcje i zatwierdzić przyciskiem *OK*.

Uwaga! Wybrane opcje zostaną zapisane w nagłówku szablonu raportu, a więc zawartość kilku pierwszych linijek szablonu raportu może ulec zmianie.

3. Plik danych

Plik danych zawiera informacje o analizowanych w raporcie jednostkach obserwacji (typowo: studentach). Aby plik danych mógł być wykorzystany do wygenerowania danego raportu, musi zawierać wszystkie zmienne, które są wykorzystywane w szablonie danego raportu (rozdział 5), a które nie zostaną wczytane z pliku definicji odbiorców (rozdział 6).

3.1. Formaty danych

Wspierane są dwa formaty danych:

- **CSV** zapisany w sposób zgodny z MS Excel przy polskich ustawieniach językowych:
  - separator pola: średnik,
  - separator dziesiętny: przecinek,
  - separator tekstu: cudzysłów,
  - kodowanie znaków: Windows-1250.

  Jest to format właściwy przy imporcie danych z zewnętrznych programów – praktycznie każdy program statystyczny i arkusz kalkulacyjny obsługuje eksport do formatu CSV, w formacie tym zapisywane są również dane eksportowane z systemów IRK i USOS.

- **RData** (format zapisu danych programu *R*) z zapisaną dokładnie jedną ramką danych.

  Jest to format właściwy do przygotowania większych zbiorów danych, które mają być wykorzystywane w raportach interaktywnych WWW (rozdział 7). Dane zapisane w tym formacie wczytują się znacznie szybciej niż z formatu CSV, co wymiernie skraca czas potrzebny na odświeżenie raportu.

  Aby skonwertować dane z formatu CSV do RData, należy w konsoli (dolna część lewej szpalty okna *RStudio*) wydać komendy:

```
library(MLAK)
dane = wczytajCSV('ścieżkaDoPliku.csv')
save(dane, file = 'ścieżkaZapisuPliku.RData')
```

3.2. Ograniczenia nazw zmiennych (kolumn)

Aby nie kolidować z dozwolonymi nazwami zmiennych w *R*, nazwy zmiennych (nagłówki kolumn w pliku CSV) powinny zawierać jedynie:

- małe i duże litery alfabetu łacińskiego (bez polskich znaków);
- cyfry (z wyjątkiem pierwszego znaku);
- podkreślenia i kropki.

Uwaga! *R* rozróżnia wielkość liter, a więc np. jeśli zmienna w pliku danych nosi nazwę *MojaZmienna*, to w szablonie raportu odwoływać się do niej trzeba przez *MojaZmienna*, a odwołania *MOJAZMIENNA, mojazmienna,* itp. nie będą poprawne.

3.3. Wczytywanie pliku danych w szablonie raportu

Najłatwiejszym sposobem załadowania danych w szablonie raportu (patrz rozdział 5) jest wywołanie w umieszczonej na samym początku szablonu raportu wstawce *R* funkcji `zaladujOdbiorce()`, np.:

````
---
title: "Mój raport"
output:
  pdf_document
---
```{r}
library(MLAK)
zaladujOdbiorce('mójPlikOdbiorcow.csv', 'mójPlikDanych.csv')
```
````

Wszystkie zmienne (kolumny) z załadowanego w ten sposób zbioru danych staną się dostępne w szablonie raportu. Jeśli np. zbiór danych zawierał zmienne (kolumny) ERASMUS i DATADYP, to będzie się teraz można do nich odwołać wprost przez te nazwy.

4. Plik definicji odbiorców

Typowo plik danych (rozdział 3) zawiera informacje zbiorcze, np. dla całej uczelni (albo nawet ogólnopolskie), raporty zaś są generowane w podziale na pewne podzbiorowości (np. wydziały, kierunki, lata studiów itp.). Plik definicji odbiorców przechowuje wszystkie dane niezbędne do:

- wygenerowania raportów dla poszczególnych podzbiorowości, a niezawarte w pliku danych (np. nazwę wydziału czy kierunku itp.);
- odfiltrowania w pliku danych tylko tych rekordów, które należą do danej podzbiorowości (np. identyfikator wydziału czy kierunku używany w pliku danych).

Uwaga! Przy automatycznym generowaniu raportów dla wielu odbiorców (rozdział 6) pierwsza zmienna (kolumna) w pliku odbiorców zostanie użyta jako nazwy plików dla generowanych raportów.

4.1. Formaty danych

Obsługiwane są dokładnie takie same formaty danych jak w wypadku pliku danych (patrz rozdział 3.2), obowiązują również te same ograniczenia nazw zmiennych (patrz rozdział 3.3).

4.2. Wczytywanie pliku danych odbiorcy w szablonie raportu

Najłatwiejszym sposobem wczytania danych konkretnego odbiorcy w szablonie raportu (patrz rozdział 5) jest wywołanie w umieszczonej na samym początku szablonu raportu wstawce *R* funkcji `zaladujOdbiorce()`, np.:

```
---
title: "Mój raport"
output:
  pdf_document
---
```{r}
library(MLAK)
zaladujOdbiorce('mójPlikOdbiorcow.csv', 'mójPlikDanych.csv')
```
```

W ten sposób utworzone zostaną zmienne o nazwach identycznych jak nazwy zmiennych (kolumn) w pliku definicji odbiorców i wartościach takich jak wartości dla pierwszego odbiorcy zdefiniowanego w pliku.

Istnieje również możliwość wczytania danych dowolnego odbiorcy – w takim wypadku należy skorzystać z opcjonalnego trzeciego argumentu funkcji `zaladujOdbiorce()`, w którym można przekazać numer odbiorcy do załadowania. Np. żeby wczytać piątego odbiorcę z pliku:

```
---
title: "Mój raport"
output:
  pdf_document
---
```{r}
library(MLAK)
zaladujOdbiorce('mójPlikOdbiorcow.csv', 'mójPlikDanych.csv', 5)
```
```

4.3. Wyświetlanie w raporcie wartości z pliku definicji odbiorców

Umieszczenie w raporcie personalizowanych danych odbiorców przechowywanych w zmiennych (kolumnach) pliku definicji odbiorców możliwe jest poprzez użycie w szablonie raportu (rozdział 5) tzw. *wstawek R*. Wstawka taka ma postać `` `r nazwaZmiennejZplikuDefinicjiOdbiorców` ``.

Np. żeby uzyskać nagłówek raportu parametryzowany celownikiem nazwy odbiorcy, gdy dysponujemy plikiem definicji odbiorców w postaci:

NazwaOdbiorcy	IdOdbiorcy	NazwaOdbiorcyCel
odbiorca X	1	odbiorcy X
odbiorca Y	2	odbiorcy Y

szablon raportu powinien wyglądać następująco:

```
---
title: "Mój raport"
output:
  pdf_document
---
```{r}
library(MLAK)
zaladujOdbiorce('mójPlikOdbiorcow.csv', 'mójPlikDanych.csv', 5)
```
Raport dla odbiorcy `r NazwaOdbiorcyCel`

Dalsza część raportu…
```

Uwaga! Aby odwoływanie się do zmiennych z pliku definicji odbiorców działało w opisany powyżej sposób, szablon raportu musi rozpoczynać się od *wstawki R*, która wywołuje funkcję `zaladujOdbiorce()`, tak jak w przykładzie powyżej i jak opisano w rozdziale 4.2.

5. Szablon raportu

Szablon raportu opisuje zawartość generowanego raportu. W tym celu używany jest prosty język składu – markdown.

W języku tym pewne ustalone sekwencje znaków zamieniane są na konkretne formatowania w wynikowym dokumencie, np.

- sekwencja: `# To jest nagłówek` spowoduje wygenerowanie nagłówka pierwszego poziomu o treści *To jest nagłówek*,
- sekwencja `## To jest inny nagłówek` – nagłówka drugiego poziomu o treści *To jest inny nagłówek*,
- a sekwencja `Ten tekst nie jest pogrubiony, **a ten jest**` – akapitu, którego ostatnie trzy wyrazy będą zapisane pogrubioną czcionką.

Sposób uzyskiwania poszczególnych formatowań, tabel, itp. opisany został w rozdziałach 5.1 oraz 5.4.

Dodatkowo w szablonie raportu można osadzać dowolny kod języka *R*, co pozwala na umieszczenie w wygenerowanym raporcie zawartości tworzonej dynamicznie – czy to na podstawie pliku definicji odbiorców, czy pliku danych; np. zakładając, że w pliku danych istnieje zmienna *OCENA*, możemy wstawić w raporcie średnią ocen za pomocą sekwencji `` `r E(OCENA)` ``. Ta funkcjonalność opisana została w rozdziałach 5.2, 5.3, 5.5 i 5.6.

Uwaga! Tworzenie, zapisywanie i otwieranie szablonów raportów w programie *RStudio*, jak również uruchamianie generowania raportu na podstawie szablonu opisane zostało w rozdziałach 2.1.1–2.1.5.

Każdy szablon raportu generowanego z użyciem pakietu *MLAK* rozpoczyna się:

- Nagłówkiem opisującym tytuł raportu oraz ostatnio wybrany format eksportu:

```
---
title: "Tytuł raportu"
output: pdf_document
---
```

Pozostałe pola nagłówka (np. dodawane automatycznie przy tworzeniu nowego szablonu pola `author` czy `date`) są nieobowiązkowe i mogą zostać usunięte). Nagłówek opisuje również specyficzne dla poszczególnych formatów wyjściowych opcje eksportu – ich dostosowanie opisano w rozdziale 2.1.5.

- *Wstawką R*, która załaduje pakiet *MLAK* oraz wczyta dane z pliku danych (patrz rozdział 3) i przykładowego odbiorcę z pliku definicji odbiorców (patrz rozdział 4):

```
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('ścieżkaDoPlikuOdbiorców', 'ścieżkaDoPlikuDanych'))
```
```

Dalsza część szablonu raportu opisuje już jego treść.

5.1. Formatowanie tekstu

- Styl znaków:
  - `*kursywa*, _kursywa_`
  - `**pogrubienie**, __pogrubienie__`
  - `^indeks górny^`
  - `~indeks dolny~`
  - `~~przekreślenie~~`
- Akapity:
  - pusta linia oddziela akapity, np.:

```
Akapit 1.
W dalszym ciągu akapit 1.

Akapit 2.
```

  - wymuszone łamanie wiersza – dwie spacje na końcu linii :

```
Linia z wymuszonym{spacja}{spacja}
podziałem wiersza
```

- Nagłówki:
  - `# 1. poziomu`

- `## 2. poziomu`
- ltd.

- Listy:
  - Nienumerowane – punkt sygnalizowany przez `*`, każdy następny poziom oznaczany 4 spacjami, np.:

```
* punkt 1
    * punkt 1.1
* punkt 2
```

  - Numerowane – punkt sygnalizowany numerem, każdy następny poziom oznaczany 4 spacjami, np.:

```
1. punkt 1
    1. punkt 1.1
2. punkt 2
```

- Statyczne rysunki: `![](ścieżkaDoRysunku)`
- Linki: `http://adres`, `[tekst wyświetlany](adres)`
- Linia pozioma `---`, `***`
- Przypis dolny:
  - `[^tekst]` w miejscu, które odnosi do przypisu;
  - `[^tekst]: treść przypisu` w miejscu, w którym określona jest treść przypisu.

Kompletna i aktualna dokumentacja wszystkich dostępnych sekwencji formatowania jest dostępna pod adresem http://rmarkdown.rstudio.com/authoring_pandoc_markdown.html.

Przykład

Przykładowy szablon raportu zawierającego jedynie statyczny tekst (a więc niezależnie od zawartości pliku danych i pliku definicji odbiorców generujący zawsze taki sam raport):

````
---
title: "Tytuł raportu"
output: pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('ścieżkaDoPlikuOdbiorców', 'ścieżkaDoPlikuDanych'))
```

# Pierwsza część raportu
````

```
Treść akapitu z wymuszonym
łamaniem linii.

Inny akapit

1. Prosta lista numerowana
2. Na pierwszym poziomie
    * i nienumerowana na poziomie drugim
        * i trzecim
3. Ostatni punkt listy

## Style znaków

**Pogrubienie**, *kursywa*, **_pogrubiona kursywa_**, **pogrubienie _z fragmentem
kursywy_**, ~~przekreślenie~~

H~2~0, 10^6^, przypis dolny[^1]

[^1]: treść przypisu dolnego
```

5.2. Obliczanie statystyk

Wyliczanie i umieszczanie w raporcie wartości statystyk obliczonych na podstawie pliku danych i/lub pliku definicji odbiorców możliwe jest dzięki tzw. *wstawkom R*. Jest to kod w języku *R* (w szczególności funkcje pakietu *MLAK*), którego wynik zostanie umieszczony w raporcie w miejscu, gdzie wstawka znajduje się w szablonie raportu.

Składnia *wstawki R* to: `` `r kodR` ``, np.:

```
Średnia ocena studentów: `r E(OCENA)`
```

Pakiet *MLAK* udostępnia następujące funkcje do obliczania statystyk:

- `N(zmienna)` – liczba obserwacji zmiennej niebędących brakami danych;
- `N(zmienna, wartość)` – liczba obserwacji zmiennej posiadających określoną wartość:
    - aby otrzymać liczbę braków danych, za `wartość` należy podstawić `NA`;
    - `wartość` może być również zbiorem, np. `N(OCENA, c(2, 3, NA))` – zwróć liczbę obserwacji, dla których zmienna OCENA ma wartość 2, 3 lub jest brakiem danych.
- `E(zmienna)`, `E(zmienna, dokl = n)` – średnia wartość zmiennej:
    - domyślnie zaokrąglona do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).
- `Me(zmienna)`, `Me(zmienna, dokl = n)` – mediana danej zmiennej:

- domyślnie zaokrąglona do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).

- `Q(zmienna, kwantyl, liczbaKwantyli)`,
  `Q(zmienna, kwantyl, liczbaKwantyli, dokl = n)` – dowolnie określony kwantyl danej zmiennej, np. `Q(zmienna, 1, 4)` – 1. kwartyl;

  - domyślnie zaokrąglony do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).

- `P(zmienna, wartość)`,
  `P(zmienna, wartość, dokl = n)`,
  `P(zmienna, wartość, znakProcent = FALSE)`,
  `P(zmienna, wartość, dokl = n, znakProcent = FALSE)` – wyrażony w procentach odsetek obserwacji o zadanej wartości zmiennej:

  - za podstawę procentowania przyjmowana jest liczba wszystkich obserwacji danej zmiennej (**włączając braki danych**);

  - `wartość` może być również zbiorem, np. `P(OCENA, c(2, 3, NA))` – zwróć odsetek obserwacji, dla których zmienna OCENA ma wartość 2, 3 lub jest brakiem danych;

  - domyślnie wynik zaokrąglany jest do jednego miejsca po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`);

  - domyślnie do wyniku dołączany jest znak %, np. `43,1%`; opcjonalnie można usunąć znak % ze zwracanej wartości (argument `znakProcent = FALSE`).

- `Pw(zmienna, wartość)`,
  `Pw(zmienna, wartość, dokl = n)`,
  `Pw(zmienna, wartość, znakProcent = FALSE)`,
  `Pw(zmienna, wartość, dokl = n, znakProcent = FALSE)` – wyrażony w procentach odsetek obserwacji o zadanej wartości zmiennej:

  - za podstawę procentowania przyjmowana jest liczba obserwacji danej zmiennej, które **nie są brakami danych**;

  - `wartość` może być również zbiorem, np. `Pw(OCENA, c(2, 3))` – zwróć odsetek obserwacji, dla których zmienna OCENA ma wartość 2 lub 3;

  - domyślnie wynik zaokrąglany jest do jednego miejsca po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`);

  - domyślnie do wyniku dołączany jest znak %, np. `43,1%`; opcjonalnie można usunąć znak % ze zwracanej wartości (argument `znakProcent = FALSE`).

- `R(zmienna1, zmienna2)`,
  `R(zmienna1, zmienna2, dokl = n)` – współczynnik korelacji liniowej (Pearsona) pomiędzy zadanymi zmiennymi;

- domyślnie zaokrąglona do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).

- `R2(zmienna1, zmienna2)`, `R2(zmienna1, zmienna2, dokl = n)` – kwadrat współczynnika korelacji liniowej (Pearsona) pomiędzy zadanymi zmiennymi;

  - domyślnie zaokrąglona do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).

- `Tau(zmienna1, zmienna2)`, `Tau(zmienna1, zmienna2, dokl = n)` – współczynnik korelacji rangowej Kendalla pomiędzy zadanymi zmiennymi;

  - domyślnie zaokrąglona do dwóch miejsc po przecinku, opcjonalnie można ustawić dowolną precyzję zaokrąglenia (argument `dokl`).

Uwaga! Funkcje pakietu *MLAK*, które obliczają statystyki, automatycznie anonimizują wynik; jeśli liczba analizowanych obserwacji była zbyt mała, zamiast wyniku zwrócony zostanie znak -.

Przykład

Załóżmy, że dysponujemy prostym zbiorem danych postaci:

Przedmiot	Ocena
1	2
1	3
1	3
1	2
1	4
2	3
2	4
2	5
2	5
2	3
2	Brak danych
2	Brak danych

zapisanym w pliku *dane.csv* oraz zbiorem odbiorców:

Odbiorca
X

zapisanym w pliku *odbiorcy.csv*.

Dla takich danych możemy przygotować szablon raportu ilustrujący wykorzystanie opisanych powyżej funkcji obliczających statystyki:

```
---
title: "Przykłady obliczania statystyk"
output: pdf_document
---
```

```
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv'))
```

Liczba obserwacji: `r N(Przedmiot)`
Liczba obserwacji dla przedmiotu 1: `r N(Przedmiot, 1)`
Liczba obserwacji z oceną 2 lub brakiem danych: `r N(Przedmiot, c(2, NA))`

Procent obserwacji z oceną 2 lub brakiem danych: `r P(Ocena, c(2, NA))`
Procent obserwacji z oceną 5, procentowany względem obserwacji o znanej ocenie, zaokrąglony do pełnych procentów: `r Pw(Ocena, 5, dokl = 0)`
Procent obserwacji z oceną 2 lub brakiem danych, zaokrąglony do pełnych procentów, bez dołączonego znaku procenta: `r P(Ocena, c(2, NA), dokl = 0, znakProcent = FALSE)`

Średnia ocena: `r E(Ocena)`
Średnia ocena zaokrąglona do liczby całkowitej: `r E(Ocena, dokl = 0)`

Pierwszy kwartyl zmiennej Ocena: `r Q(Ocena, 1, 4)`
Drugi kwartyl (mediana) zmiennej Ocena: `r Q(Ocena, 2, 4)`
Trzeci kwartyl zmiennej Ocena: `r Q(Ocena, 3, 4)`
Mediana zmiennej Ocena: `r Me(Ocena)`
Piąty stanin zmiennej Ocena: `r Q(Ocena, 5, 9)`

Korelacja rangowa pomiędzy przedmiotem a oceną: `r Tau(Przedmiot, Ocena)`
```

5.3. Obliczanie statystyk dla podgrup

Opisane w poprzednim rozdziale statystyki obliczane były na całym zbiorze danych, często jednak zachodzi potrzeba odfiltrowania danych dla pewnych podgrup, np. ze względu na odbiorcę raportu, opisywaną zbiorowość lub kombinację ich obu.

Każdą zmienną odfiltrować można za pomocą składni: `zmienna[warunek]`.

`Warunek` może przyjmować kilka różnych postaci:

- `zmienna operator wartość` – wybiera podgrupę, która spełnia zadaną relację, gdzie dostępne operatory to:
  - `%in%` – zawieranie się w zbiorze (jeśli zbiór wartości jest pojedynczą wartością, jest on równoważny porównaniu), np. `Ocena[Rok %in% 2010]`, `Ocena[rok %in% c(2010, 2011)]`;
  - `>, >=, <, <=` - większe, większe bądź równe, mniejsze, mniejsze bądź równe, np. `rok >= 2010`;
- `G(zmiennaFiltrująca, kwantyle, liczbaKwantyli)` – wybiera podgrupę wyróżnioną przez zadany kwantyl ze względu na wskazaną zmienną filtrującą, np.:

`Erasmus[G(Ocena, 1, 4)]` – pozostaw tylko te obserwacje zmiennej `Erasmus`, które należą do pierwszej grupy kwartylowej ze względu na zmienną `Ocena`.

- `G(zmiennaFiltrująca, kwantyle, liczbaKwantyli, innyWarunek)` – jak powyżej, ale zarówno przy obliczaniu kwantyli, jak i przy filtrowaniu zmiennej brane są pod uwagę jedynie obserwacje wskazywane przez `innyWarunek`, np.:

  `Erasmus[G(Ocena, 1, 4, Rok %in% 2010)]` – pozostaw tylko te obserwacje zmiennej `Erasmus`, dla których zmienna `Rok` ma wartość 2010 i które należą do pierwszego kwartyla wartości zmiennej `Ocena` dla roku 2010.

- Powyższe konstrukcje **można ze sobą łączyć** za pomocą operatorów & (i) oraz | (lub) oraz nawiasów, np.: `Ocena[Rok %in% 2010 & KierunekStudiow %in% c(1, 2)]`.

Ponieważ większość *warunków* jest używana wielokrotnie, a wpisywanie ich przy każdym użyciu jest niewygodne i czyni szablon raportu nieczytelnym, warto zapamiętać je na początku szablonu raportu. Służy do tego konstrukcja:

```
```{r}
warunek1 = definicjaWarunku1
warunek2 = definicjaWarunku2
warunek3 = warunek2 & dodatkoweWarunki
(…itd…)
```
```

Zapisane w ten sposób *warunki* można potem łatwo stosować w dalszej części szablonu raportu, np.:

```
E(Ocena[warunek1])
P(Ocena[warunek1 & Stypendium %in% 0])
```

Przykład

Załóżmy, że dysponujemy prostym zbiorem danych postaci:

Przedmiot	Ocena
1	2
1	3
1	3
1	2
1	4
1	2
1	3
1	3
1	3
1	5
2	3
2	4
2	5
2	5
2	3
2	Brak danych

2	Brak danych

zapisanym w pliku *dane.csv* oraz zbiorem odbiorców:

KodPrzedmiotu	NazwaPrzedmiotu
1	Statystyka
2	Wstęp do socjologii

zapisanym w pliku *odbiorcy.csv*.

Dla takich danych chcemy przygotować szablon raportu podsumowujący wyniki z przedmiotu określonego w zbiorze definicji odbiorców.

Oznacza to, że każdą statystykę obliczać chcemy jedynie dla obserwacji związanych z danym przedmiotem. Zapamiętamy więc filtr `Przedmiot %in% KodPrzedmiotu`, który odfiltruje obserwacje w zbiorze danych zgodnie z wartością zmiennej `KodPrzedmiotu` wczytaną z pliku definicji odbiorców.

````
---
title: "Przykłady obliczeń w podgrupach"
output: pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv'))
```
```{r}
fOdbiorca = Przedmiot %in% KodPrzedmiotu
```
Średnia ocena z przedmiotu `r NazwaPrzedmiotu`:
  `r E(Ocena[fOdbiorca])`

Liczba studentów, którzy nie zaliczyli przedmiotu:
  `r N(Ocena[fOdbiorca & Ocena %in% 2])`

Średnia ocena studentów z wynikiem poniżej mediany:
  `r E(Ocena[G(Ocena, 1, 2, fOdbiorca)])`
````

5.4. Tabele

Język *markdown* udostępnia kilka wariantów opisu tabel. Poniżej opisano dwa najważniejsze z nich. Pełny, najbardziej aktualny opis znaleźć można w dokumentacji pakietu rmarkdown.

Uwaga! Żaden dostępny w języku *markdown* sposób opisu tabel nie umożliwia scalania komórek (ściśle rzecz biorąc, jest to ograniczenie programu *Pandoc*).

Pipe tables

Najprostszym sposobem definiowania tabel są tzw. *pipe tables*, w których:

- każda linia oznacza jeden wiersz tabeli;
- separatorem kolumny jest pionowa kreska: `|`;

- nagłówek od treści tabeli oddzielany jest specjalną linią, która pozwala określić wyrównanie kolumn (wyrównanie wartości samych komórek w ramach szablonu nie ma żadnego znaczenia dla ich wyrównania w wygenerowanym raporcie – patrz przykład poniżej).

Przykładem tego typu tabeli może być:

```
kolumna wycentrowana | wyrównana do lewej | wyrównana do prawej
:--------------------:|:-------------------|--------------------:
                   1 |                  2 |               tekst
                   2 |  10                |  inny tekst
```

Uwaga! W wypadku tego typu tabel zawartość komórek musi zmieścić się w jednym wierszu; w wypadku, gdy potrzebna jest tabela z komórkami zawierającymi wiele linijek, należy skorzystać z opisanych w dalszej części rozdziału *multiline tables*;

Uwaga! Poszczególne kolumny w kolejnych wierszach nie muszą na siebie nachodzić – zupełnie poprawny (choć niepolecany z uwagi na niską czytelność szablonu raportu) jest np. zapis:

```
kolumna1 | kolumna2
:-:|-:
1 | 2
234 | 4598
```

W tabeli można wstawiać obliczone wartości statystyk – nie różni się to niczym od wstawiania ich poza tabelą, np.:

```
kolumna wyśrodkowana | wyrównana do lewej | wyrównana do prawej
:--------------------:|:-------------------|--------------------:
      `r E(zmienna)` |                  2 | `r Q(zmienna, 1, 4)`
      `r N(zmienna)` |  10                |  inny tekst
```

Multiline tables

Składnia *multiline tables* umożliwia tworzenie tabel, w których zawartość komórek będzie podzielona na wiele wierszy. W tym wypadku:

- Tabela rozpoczyna się linią składającą się z dywizów.
- Dalej następuje nagłówek tabeli – może się on składać z wielu linii (podział na kolumny wyznacza linia kończąca nagłówek – patrz niżej). Położenie tekstu nagłówka względem linii podziału kolumn decyduje o wyrównaniu kolumn (patrz przykład poniżej).
- Nagłówek kończy linia składająca się z dywizów rozdzielonych spacjami – miejsca występowania spacji oznaczają granice kolumn.
- Później następuje opis zawartości tabeli – separatorem wiersza jest pusta linia. Każdy wiersz tabeli może się składać z wielu linii (jednak w wynikowej tabeli połamane zostaną one automatycznie, niekoniecznie zgodnie z łamaniem w szablonie). Podział na kolumny następuje na pozycjach wskazywanych przez spacje w wierszu oddzielającym nagłówek tabeli od jej zawartości.

- Tabela kończy się linią składającą się z dywizów i **następującą po niej pustą linią**.

W tabeli można wstawiać obliczone wartości statystyk – nie różni się to niczym od wstawiania ich poza tabelą.

Przykładem tego typu tabeli może być:

```
-------------------------------------------------------------
 Kolumna       Wyrównana                              do prawej
 wyśrodkowana  do lewej
-------------- ------------ ----------------------------------
`r E(zmienna)`     2            `r Q(zmienna, 1, 4)`

`r N(zmienna)`        10         komórka, której zawartość
                                 może zajmować wiele linii
                                 zarówno w szablonie raportu,
                                 jak i w wygenerowanym raporcie
-------------------------------------------------------------
```

Uwaga! Dla poprawnego sformatowania *multiline tables* kluczowe znaczenie ma odpowiednie wyrównanie treści tabeli względem linii podziału kolumn.

Przykład

Załóżmy, że dysponujemy prostym zbiorem danych postaci:

Przedmiot	Ocena
1	2
1	3
1	3
1	2
1	4
2	3
2	4
2	5
2	5
2	3
2	Brak danych
2	Brak danych

zapisanym w pliku *dane.csv* oraz zbiorem odbiorców:

Odbiorca
X

zapisanym w pliku *odbiorcy.csv*.

Dla takich danych możemy przygotować szablon raportu ilustrujący wykorzystanie tabel, wykorzystujący jednocześnie funkcje obliczające statystyki (rozdział 5.2) w podziale na grupy (rozdział 5.3):

```
---
```

```
title: "Przykłady tabel"
output: pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv'))
```
```{r}
fPrz1 = Przedmiot %in% 1
fPrz2 = Przedmiot %in% 2
```
Przedmiot	Liczba obserwacji	Średnia ocena
Przedmiot 1	`r N(Przedmiot[fPrz1])`	`r E(Przedmiot[fPrz1])`
Przedmiot 2	`r N(Przedmiot[fPrz2])`	`r E(Przedmiot[fPrz2])`

 Przedmiot   Liczba obserwacji   Średnia ocena
------------ ------------------------ ------------------------
Przedmiot 1   `r N(Przedmiot[fPrz1])`   `r E(Przedmiot[fPrz1])`

Przedmiot 2   `r N(Przedmiot[fPrz2])`   `r E(Przedmiot[fPrz2])`
------------ ------------------------ ------------------------
```

5.5. Wykresy

Pakiet *MLAK* umożliwia łatwe rysowanie trzech rodzajów wykresów:

- **Wykresów rozkładu** wartości zmiennej (histogramu):

  `wykresHistogram(zmienna, liczbaPrzedziałów, tytuł, tytułOsiX, tytułOsiY)`

- **Wykresów słupkowych:**
  - skumulowanych:

    `wykresSlupkowy(dane, FALSE, tytuł, tytułOsiX, tytułOsiY, sufiksY)`

  - nieskumulowanych:

    `wykresSlupkowy(dane, TRUE, tytuł, tytułOsiX, tytułOsiY, sufiksY)`

  parametr `sufiksY` umożliwia dostosowanie etykiet osi Y, np. gdy wyświetlane są wartości procentowe (patrz ostatni wykres w przykładzie).

- **Wykresów kołowych**:
  - gdy dane w legendzie mają dokładnie odpowiadać przekazanym wartościom:

    `wykresKolowy(dane, tytuł)`

- gdy dane w legendzie mają odpowiadać wystandaryzowanym wartościom:

  ```
  wykresKolowyNorm(dane, tytul = tytuł)
  wykresKolowyNorm(dane, dokł, tytuł)
  ```

- gdy na wykresie mają zostać zaprezentowane częstości występowania przekazanej zmiennej:

  ```
  wykresKolowyZlicz(zmienna, tytul = tytuł)
  wykresKolowyZlicz(zmienna, etykiety, tytuł)
  ```

  za pomocą parametru `etykiety` można nadać etykiety wartościom prezentowanej zmiennej (patrz przykład).

Odpowiednią funkcję należy osadzić w raporcie jako:

```
```{r, fig.height = wysokośćWykresuWCalach, fig.width = szerokośćWykresuWCalach}
wywołanieFunkcjiRysującejWykres()
```
```

przy czym jeśli dany wymiar wykresu nie zostanie podany, przyjęta zostanie wartość domyślna, np.:

```
```{r. fig.height = 2.5}
wykresHistogram(Ocena, 4, 'Rozkład ocen', 'Ocena', 'Częstość')
```
```

Uwaga! W wypadku funkcji `wykresHistogram()` oraz `wykresKolowyZlicz()` jako pierwszy argument typowo przekazywana będzie zmienna z pliku danych, natomiast w wypadku pozostałych funkcji rysujących wykresy na ogół będzie to zbiór już obliczonych wartości statystyk (patrz przykład poniżej).

Uwaga! Różnice pomiędzy działaniem różnych funkcji rysujących wykresy kołowe najlepiej prześledzić na przykładzie.

Uwaga! Każda z funkcji rysujących wykresy może również opcjonalnie przyjąć argument `opcjeWykresu = opcje`, gdzie `opcje` to dowolne funkcje pakietu ggplot2, np. `wykresSlupkowy(c(1, 2), opcjeWykresu = ggplot2::theme_grey())`.

Uwaga! Inne rodzaje wykresów można uzyskać, korzystając z odpowiednich pakietów języka *R* (np. funkcje z pakietu podstawowego czy pakietu ggplot2).

Przykład

Załóżmy, że dysponujemy prostym zbiorem danych postaci:

Przedmiot	Ocena
1	2
1	3
1	3
1	2
1	4
2	3
2	4

2	5
2	5
2	3
2	Brak danych
2	Brak danych

zapisanym w pliku *dane.csv* oraz zbiorem odbiorców:

Odbiorca
X

zapisanym w pliku *odbiorcy.csv*.

Na ich podstawie chcemy wygenerować wykresy rozkładu częstości występowania poszczególnych ocen:

- w postaci histogramu;
- w postaci wykresu kołowego:
  - z niestandaryzowanymi wartościami legendy (wartości częstości obliczone za pomocą funkcji `N()`);
  - ze standaryzowanymi wartościami legendy (wartości częstości obliczone za pomocą funkcji `N()`);
  - obliczone wprost ze zmiennej Ocena przez funkcję `wykresKolowyZlicz()`;
- w postaci skumulowanego wykresu słupkowego;
- w postaci nieskumulowanego wykresu słupkowego.

````
---
title: "Przykłady obliczania statystyk"
output: pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv'))
```

# Histogramy

```{r, fig.height = 2}
wykresHistogram(Ocena, 4, 'Rozkład ocen - 4 grupy', 'Ocena', 'Częstość')
```

```{r, fig.height = 2}
wykresHistogram(Ocena, 3, 'Rozkład ocen - 3 grupy', 'Ocena', 'Częstość')
```

# Wykresy kołowe
````

Liczba wystąpień policzona funkcją N()

```{r, fig.height = 3}
wykresKolowy(
  c('ndost.' = N(Ocena, 2), 'dost.' = N(Ocena, 3), 'db.' = N(Ocena, 4), 'bdb.' = N(Ocena, 5)),
  'Częstość występowania ocen'
)
```

Odsetki policzone funkcją Pw()

```{r, fig.height = 3}
wykresKolowy(
  c('ndost.' = Pw(Ocena, 2), 'dost.' = Pw(Ocena, 3), 'db.' = Pw(Ocena, 4), 'bdb.' = Pw(Ocena, 5)),
  'Częstość występowania ocen'
)
```

Liczba wystąpień policzona funkcją N(), znormalizowana przez wykresKolowyNorm()

```{r, fig.height = 3}
wykresKolowyNorm(
  c('ndost.' = N(Ocena, 2), 'dost.' = N(Ocena, 3), 'db.' = N(Ocena, 4), 'bdb.' = N(Ocena, 5)),
  tytul = 'Częstość występowania ocen'
)
```

Wykres jest identyczny, jak przy obliczeniu odsetków funkcją Pw()

Odsetki policzone bezpośrednio ze zmiennej

```{r, fig.height = 3}
wykresKolowyZlicz(Ocena, tytul = 'Częstość występowania ocen')
```

Wynik jest ten sam, jak na poprzednich wykresach, brak jednak etykiet wartości zmiennej.

```{r, fig.height = 3}
wykresKolowyZlicz(
  Ocena,
  c('2' = 'ndost. ', '3' = 'dost.', '4' = 'db.', '5' = 'bdb.'),
  tytul = 'Częstość występowania ocen'
)
```

````
Z dodanymi etykietami.

# Wykresy słupkowe

## Liczba wystąpień - wykres nieskumulowany

```{r, fig.height = 3}
wykresSlupkowy(
  c('ndost.' = N(Ocena, 2), 'dost.' = N(Ocena, 3), 'db.' = N(Ocena, 4), 'bdb.' =
N(Ocena, 5)),
  FALSE,
  tytul = 'Częstość występowania ocen'
)
```

## Procent wystąpień - wykres skumulowany

```{r, fig.height = 3}
wykresSlupkowy(
  c('ndost.' = Pw(Ocena, 2), 'dost.' = Pw(Ocena, 3), 'db.' = Pw(Ocena, 4), 'bdb.' =
Pw(Ocena, 5)),
  TRUE,
  tytul = 'Częstość występowania ocen',
  sufiksY = '%'
)
```

```{r, fig.height = 3, fig.width = 3}
wykresSlupkowy(
  c('ndost.' = Pw(Ocena, 2), 'dost.' = Pw(Ocena, 3), 'db.' = Pw(Ocena, 4), 'bdb.' =
Pw(Ocena, 5)),
  TRUE,
  tytul = 'Częstość występowania ocen',
  sufiksY = '%'
)
```

Dodanie znaku % w etykietach osi Y i zmniejszenie szerokości wykresu.
````

5.6. Zawartość warunkowa

Czasami zachodzi konieczność, aby dany fragment raportu wyświetlić tylko pod pewnymi warunkami, np. tylko dla niektórych odbiorców. Można to osiągnąć za pomocą składni:

````
```{r}
if(warunek){
      wstawTresc('kod szablonu')
}
```
````

gdzie `warunek` skonstruowany jest dokładnie w ten sam sposób jak w wypadku obliczania statystyk dla podgrup (rozdział 5.3), a `kod szablonu` to fragment szablonu, który ma zostać wyświetlony tylko w wypadku spełnienia `warunku`.

W wypadku, kiedy wstawiany warunkowo fragment raportu jest obszerny, zamiast explicite podawać jego kod jako argument funkcji `wstawTresc()`, można zapisać go w oddzielnym pliku szablonu raportu, a do funkcji `wstawTresc()` przekazać ścieżkę do tego pliku (patrz przykład).

Przykład

Załóżmy, że dysponujemy prostym zbiorem danych postaci:

Przedmiot	Ocena
1	2
1	3
1	3
1	2
1	4
1	2
1	3
1	3
1	3
1	5
2	3
2	4
2	5
2	5
2	3
2	Brak danych
2	Brak danych

zapisanym w pliku *dane.csv* oraz zbiorem odbiorców:

KodPrzedmiotu	NazwaPrzedmiotu
1	Statystyka
2	Wstęp do socjologii

zapisanym w pliku *odbiorcy.csv*.

Dla tak przygotowanych danych chcemy przygotować raport, w którym wiersze tabeli opisujące kwantyle ocen wyświetlane będą jedynie dla przedmiotu o kodzie przedmiotu równym 1:

````
---
title: "Przykłady obliczania statystyk"
output: pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv', 1))
```
```{r}
filtr = Przedmiot %in% KodPrzedmiotu
````
````

```
Statystyka  |                  Wartość
:----------:|-------------------------:
Średnia     |       `r E(Ocena[filtr])`
```{r}
if(KodPrzedmiotu %in% 1){
wstawTresc('Q1          | `r Q(Ocena[filtr], 1, 4)`
Q2          | `r Q(Ocena[filtr], 2, 4)`
Q3          | `r Q(Ocena[filtr], 3, 4)`')
}
```
```

Zmieniając numer odbiorcy do wczytania w 7 linii szablonu i generując raport, można zauważyć, że w zależności od odbiorcy tabela ma 1 lub 4 wiersze.

6. Automatyczne generowanie wielu raportów

Automatyczne generowanie raportów dla wielu odbiorców odbywa się przy użyciu funkcji `generujRaporty()` pakietu *MLAK*. Aby z niej skorzystać, niezbędne jest stworzenie krótkiego skryptu w języku *R*, który załaduje pakiet *MLAK*, a następnie wywoła funkcję `generujRaporty()` z odpowiednimi argumentami. W tym celu należy:

- przygotować szablon raportu, plik danych oraz plik definicji odbiorców, tak jak w wypadku generowania pojedynczego raportu;
- utworzyć nowy skrypt *R* – w *RStudio*:
  - wybrać menu *File* → *New File* → *R Script*,
  - wybrać menu *File* → *Save* i zapisać skrypt w wybranej lokalizacji, nadając mu nazwę z rozszerzeniem *.R* (najlepiej w tym samym katalogu, w którym znajdują się plik szablonu, danych oraz definicji odbiorców);
- wpisać kod skryptu, odpowiednio dostosowując ścieżki do plików:

```
library(MLAK)
generujRaporty(
    'ścieżkaDoPlikuSzablonuRaportu',
    'ścieżkaDoPlikuDanych',
    'ścieżkaDoPlikuDefinicjiOdbiorców',
    'katalogDoKtóregoZostanąZapisaneRaporty'
)
```

- ustawić katalog roboczy na katalog skryptu – w *RStudio* wybrać menu *Session* → *Set Working Directory* → *To Source File Location*;
- wykonać skrypt – w *RStudio* wykonać jedną z czynności:
  - wybrać w menu *Code* → *Run Region* → *Run All,*

- nacisnąć kombinację klawiszy *CTRL+ALT+R*,
- zaznaczyć cały skrypt i nacisnąć kombinację klawiszy *CTRL+ENTER*.

W wyniku wykonania skryptu we wskazanym katalogu zapisane zostaną raporty dla wszystkich odbiorców opisanych w pliku definicji odbiorców.

Uwaga! Za nazwy plików dla poszczególnych odbiorców przyjęte zostaną wartości z pierwszej zmiennej (kolumny) pliku definicji odbiorców. W wypadku, jeśli zmienna ta nie będzie miała unikalnych wartości, wtedy dla duplikatów raporty wygenerowane później nadpiszą te wygenerowane wcześniej.

Uwaga! Jako że funkcja `generujRaporty()` sama kolejno wczytuje właściwych odbiorców, wszystkie wywołania funkcji `wczytajOdbiorce()` zostaną automatycznie pominięte. Nie trzeba więc usuwać ich z szablonu raportu przed automatycznym wygenerowaniem raportów dla wielu odbiorców.

6.1. Przykład

Załóżmy, że w tym samym katalogu przygotowaliśmy:

- Szablon raportu *szablon.Rmd*:

````
---
title: "Mój raport"
output:
  pdf_document
---
```{r, echo = FALSE}
library(MLAK)
attach(wczytajOdbiorce('odbiorcy.csv', 'dane.csv'))

filtr = Id == IdOdbiorcy
```
Średnia ocen dla `r NazwaOdbiorcyCel`: `r E(Ocena[filtr])`
````

- Plik danych *dane.csv*:

Id	Ocena
1	2
1	3
1	3
1	2
1	4
2	3
2	4
2	5
2	5
2	3

- Plik definicji odbiorców *odbiorcy.csv*:

NazwaOdbiorcy	IdOdbiorcy	NazwaOdbiorcyCel
odbiorca X	1	odbiorcy X
odbiorca Y	2	odbiorcy Y

- Skrypt do automatycznego generowania raportów:

```
library(MLAK)
generujRaporty(
    'szablon.Rmd',
    'dane.csv',
    'odbiorcy.csv',
    'raporty'
)
```

Wykonanie skryptu spowoduje utworzenie katalogu *raporty*, w którym znajdować się będą pliki *odbiorca X.pdf* oraz *odbiorca Y.pdf* – raporty dla wszystkich odbiorców zdefiniowanych w pliku definicji odbiorców.

7. Interaktywne raporty WWW

Z pomocą pakietu *rmarkdown* możliwe jest zastąpienie pliku definicji odbiorców interaktywnym formularzem na stronie WWW. Jeśli użytkownik zmieni wartości formularza, wyświetlany raport zostanie automatycznie odświeżony na podstawie wartości aktualnie wybranych w formularzu. Aby skorzystać z tej możliwości, należy:

- przygotować szablon raportu i plik danych, tak jak w wypadku generowania pojedynczego raportu;
- przygotować oddzielny szablon formularza, który zastąpi plik definicji odbiorców. Budowa tego pliku zostanie dokładnie opisana poniżej;
- wygenerować interaktywną stronę WWW poprzez naciśnięcie przycisku *Run Document* w otwartym szablonie formularza (przycisk *Run Document* znajduje się na pasku nad polem edycji szablonu formularza).

7.1. Szablon formularza

Szablon formularza jest to plik *.Rmd*, który zawiera następujące elementy:

- nagłówek informujący o tym, że jest to szablon formularza interaktywnego:

```
---
title: "Tytuł strony WWW "
runtime: shiny
---
```

- *wstawkę R*, a w niej:

- definicję pól formularza poprzez wywołanie funkcji `inputPanel()`;
- opis sposobu wygenerowania raportu na podstawie wybranych przez użytkownika wartości pól formularza poprzez wywołanie funkcji `renderUI()`.

Przygotowanie formularza najłatwiej będzie omówić na przykładzie:

- Załóżmy, że dysponujemy szablonem raportu oraz plikiem danych, jak w przykładzie w rozdziale 6.1.
- Chcemy zastąpić plik definicji odbiorców poprzez danie użytkownikowi możliwości samodzielnego wyboru odbiorcy z listy rozwijalnej. Listę taką tworzymy poprzez wykonanie poleceń z pliku *formularz.Rmd* (przykład poniżej).
- Do poprawnego wygenerowania raportu potrzebujemy również pozostałych danych z pliku definicji odbiorców – wartości zmiennych *IdOdbiorcy* oraz *NazwaOdbiorcyCel* dla odbiorcy wybranego przez użytkownika w formularzu. Słowniki wartości tych zmiennych będziemy musieli na sztywno umieścić w kodzie wywołania funkcji `renderUI()`.

Kompletny plik *formularz.Rmd* będzie miał postać:

````
---
title: "ETAP STUDIÓW"
runtime: shiny
---
```{r, echo = FALSE}
library(MLAK)

inputPanel(
  selectInput(
    'NazwaOdbiorcy',
    label = 'Nazwa odbiorcy',
    choices = c('odbiorca X', 'odbiorca Y'),
    selected = 1
  )
)

renderUI({
  slownikNazwaOdbiorcyCel = c('odbiorca X' = 'odbiorcy X', 'odbiorca Y' = 'odbiorcy
Y')
  slownikIdOdbiorcy = c('odbiorca X' = 1, 'odbiorca Y' = 2)

  NazwaOdbiorcyCel = slownikNazwaOdbiorcyCel[input$NazwaOdbiorcy]
  IdOdbiorcy = slownikIdOdbiorcy[input$NazwaOdbiorcy]

  attach(wczytajDane('dane.csv'))

  rmarkdown::render(
    input = 'szablon.Rmd',
    output_format = html_document(),
    output_file = 'tmp.html'
  )
  raport = HTML(readLines('tmp.html'))
  unlink('tmp.html')
  return(raport)
````
````

```
})
```

Pierwsza część to opisany już nagłówek informującym o tym, że jest to formularz interaktywny:

```
---
title: "ETAP STUDIÓW"
runtime: shiny
---
```

Dalej następuje *wstawka R*, w której ładowany jest pakiet *MLAK*, a w dalszej kolejności wywoływane są dwie funkcje: `inputPanel()` – opisująca kontrolki formularza oraz `renderUI()` – opisująca sposób wygenerowania raportu:

```
```{r, echo = FALSE}
library(MLAK)

inputPanel(
  (…)
)

renderUI({
  (…)
})
```
```

Funkcja inputPanel()

Funkcja `inputPanel()` przyjmuje tyle argumentów, ile jest kontrolek formularza. Każdy z tych argumentów jest wywołaniem funkcji tworzącym kontrolkę pożądanego typu (patrz http://shiny.rstudio.com/tutorial/lesson3/). W naszym przypadku jest to funkcja `selectInput()`, która tworzy listę rozwijalną. Przyjmuje ona jako kolejne argumenty:

- nazwę pola formularza – na tej podstawie nazwana zostanie zmienna dostępna w funkcji `renderUI()`, w której zostanie zapisana wartość wybrana w tym polu formularza przez użytkownika;
- etykietę – tekst, który zostanie wyświetlony obok listy rozwijalnej;
- wektor wartości, które będą wyświetlone na liście;
- numer domyślnie wybranej pozycji listy.

Tak więc kod:

```
inputPanel(
  selectInput(
    'NazwaOdbiorcy',
    label = 'Nazwa odbiorcy',
    choices = c('odbiorca X', 'odbiorca Y'),
    selected = 1
  )
)
```

tworzy formularz składający się z jednej listy rozwijalnej, z wartościami *odbiorca X* oraz *odbiorca Y*, z których domyślnie wybrana jest pierwsza. Obok listy rozwijalnej wyświetlona będzie etykieta *Nazwa odbiorcy*, a wartość wybrana przez użytkownika z listy dostępna będzie w funkcji `renderUI()` w zmiennej `input$NazwaOdbiorcy`.

Funkcja renderUI()

Funkcja `renderUI()` ma za zadanie zwrócić wygenerowany raport.

Uwaga! Wszystkie wartości wybrane przez użytkownika w formularzu są w niej dostępne na liście `input`, a więc jeśli nazwaliśmy pole formularza `NazwaOdbiorcy` (jak w omawianym przykładzie), to jego wartość dostępna jest w zmiennej `input$NazwaOdbiorcy`.

Ponieważ użytkownik wybiera jedynie nazwę odbiorcy, zaś do wygenerowania raportu potrzebne są jeszcze identyfikator odbiorcy w zmiennej oraz nazwa odbiorcy w celowniku w zmiennej, wewnątrz funkcji musimy ustawić wartości tych zmiennych. W tym celu:

- Tworzymy słowniki przechowujące wartości zmiennych `NazwaOdbiorcyCel` oraz `IdOdbiorcy` dla poszczególnych odbiorców:

```
slownikNazwaOdbiorcyCel = c('odbiorca X' = 'odbiorcy X', 'odbiorca Y' =
'odbiorcy Y')
slownikIdOdbiorcy = c('odbiorca X' = 1, 'odbiorca Y' = 2)
```

- Na podstawie wartości wybranej przez użytkownika, przekazanej w zmiennej `input$NazwaOdbiorcy`, wybieramy ze słowników właściwe wartości i przypisujemy je do zmiennych `NazwaOdbiorcyCel` oraz `IdOdbiorcy`:

```
NazwaOdbiorcyCel = slownikNazwaOdbiorcyCel[input$NazwaOdbiorcy]
IdOdbiorcy = slownikIdOdbiorcy[input$NazwaOdbiorcy]
```

Kolejne kroki są już zawsze takie same (zmieniają się ewentualnie tylko ścieżki do pliku danych i szablonu raportu):

- Wczytujemy plik danych:

```
attach(wczytajDane('dane.csv'))
```

- Mając ustawione wszystkie zmienne opisujące odbiorcę oraz wczytany plik danych, generujemy na podstawie szablonu raportu raport w formacie HTML. Używamy do tego funkcji `render()` pakietu *rmarkdown*:

```
rmarkdown::render(
  input = 'szablon.Rmd',
  output_format = html_document(),
  output_file = 'tmp.html'
)
```

- Wygenerowany raport zapisany jest w pliku *tmp.html*. Ostatni krok to odczytanie raportu z tego pliku, usunięcie pliku i zwrócenie treści raportu jako wyniku działania funkcji `renderUI()`:

```
raport = HTML(readLines('tmp.html'))
unlink('tmp.html')
return(raport)
```

7.2. Podsumowanie

Raporty w postaci interaktywnych stron WWW umożliwiają osiągnięcie istotnych korzyści:

- Dają możliwość samodzielnego generowania indywidualnie sparametryzowanych raportów bez konieczności zdobywania specjalistycznej wiedzy (wystarczy umiejętność obsługi przeglądarki internetowej).

- Jednocześnie zapewniają pełną separację osoby generującej raport od danych jednostkowych, na podstawie których jest on generowany.

Stworzenie raportu w postaci interaktywnej strony WWW nie jest trudne, wymaga jedynie zapoznania się ze sposobem opisywania kontrolek formularza, którymi użytkownik będzie parametryzował raport, oraz przygotowania kodu w języku *R*, który na podstawie wybranych wartości przygotuje zmienne normalnie wczytywane z pliku definicji odbiorców.