EDS Egzaminas

Jurgis Samaitis

January 7, 2018

Funkciju failo nuskaitymas.

```
tryCatch(source(str_c(getwd(), "//funkcijos.R"), encoding = "ISO-8859-4"),
         error = source(file = file.choose()), encoding = "ISO-8859-4")
```

1-4 Uzduotys

Naudosiu GitHub API, kad gauciau failu skaiciu ir tikslius pavadinimus.

```
# Visos duomenu "saugyklos" (repository) sha1 kodas - jo reikalauja GitHub
API,
# norint gauti vidinius repository duomenis.
repo_sha <-
  gh(
    'GET /repos/:owner/:repo/contents/:path',
    owner = "vzemlys",
    repo = "eda_2017",
    path = ""
  )[2][1][[1]]$sha

# Failu metadata.
file_info <-
  gh(
    'GET /repos/:owner/:repo/git/trees/:sha',
    owner = "vzemlys",
    repo = "eda_2017",
    path = "data",
    sha = repo_sha
  )$tree

# Pavyzdys
#head(file_info, 1)
```

1. Kiek is viso yra failu?

```
length(file_info)
```

```
## [1] 9896
```

2. Kiek yra skirtingu vietu kuriuose buvo kaupiami duomenys?

Naudodamas metadata, isgaunu failu pavadinimus. Kadangi failo pavadinimas yra miestas_data, tai isgave "miestas", galime pasakyti kiek tokiu vietoviu buvo.

```
# Failu pavadinimai.
file_path <- character()
for (i in 1:length(file_info)) {
  file_path[i] <- file_info[[i]]$path
}

# Miestu pavadinimai.
(file_cities <- str_match(file_path, "\\w*_") %>%
  unique() %>%
  str_sub(end = str_length(.) - 1))

## [1] "ciurlionis" "kaunas"     "klaipeda"   "vilnius"
```

3. Kokiu laikotarpiu buvo kaupiami duomenys?

Dar karta pasinaudoju miestas_data formatu ir isgaunu data. Paverciu i datos formata ir gaunu pradine ir galutine data.

```
# Data is failu vardu.
file_datetime <-
  str_match(file_path, "\\d{4}-\\d{2}-\\d{2}_\\d{2}:\\d{2}:\\d{2}") %>%
  unique() %>%
  ymd_hms(tz = Sys.timezone()) # Jeigu zmogus ne is lietuvos, jo problemos
kad ne ta laiko zona uzrasyta. :)

# Atsakymas i klausima.
(file_datetime_range <- file_datetime %>%
    range())

## [1] "2012-12-17 11:00:01 EET"  "2013-04-01 14:00:01 EEST"
```

4. Kokiu dazniu buvo kaupiami duomenys? (Kiek kartu per para? Kiek paru? Ar duomenys buvo kaupiami nuolatos be pertraukos?)

4. a) Kiek kartu per para?

```
# Suskaiciuoja skirtumus tarp kiekvieno laiko tarpo.
time_differences <- vector()
for (i in (seq_along(file_datetime) - 1)){

  time_differences[i] <- as.numeric(difftime(file_datetime[i],
file_datetime[i+1], units = "mins"))
}
time_differences %>%
  tibble() %>%
  summarise("Vidurkis (min)" = mean(.), "Mediana (min)" = median(.))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.1
```

```
## # A tibble: 1 x 2
##   `Vidurkis (min)` `Mediana (min)`
##              <dbl>           <dbl>
## 1            -51.1           -60.0
```

24 kartus per para pagal mediana, vidutiniskai maziau, nes yra praleistu datu.

4. b) Kiek paru?

Galutine data - pradine data.

```
file_datetime_range[2] - file_datetime_range[1]
```

```
## Time difference of 105.0833 days
```

4. c) Ar duomenys buvo kaupiami nuolatos be pertraukos?

```
# Funkcija kuri tikrina ar yra praleistu valandu. Gali grazinti ir pacias
valandas kaip faila.
missingHours(file_datetime, warnings_print = T)
```

```
## Warning in missingHours(file_datetime, warnings_print = T): Vektoriuje yra
## praleistu datu.
```

```
## # A tibble: 36 x 3
##    start_date          end_date            time_difference
##    <dttm>              <dttm>              <chr>
##  1 2013-02-05 04:00:01 2013-02-05 09:42:55 5.715 hours
##  2 2013-03-18 09:00:01 2013-03-20 07:00:01 1.916667 days
##  3 2013-03-31 00:00:01 2013-03-31 01:00:01 1 hours
##  4 2013-04-01 11:00:01 2012-12-17 12:00:02 -104.9583 days
##  5 2012-12-17 13:00:02 2012-12-18 05:00:02 16 hours
##  6 2012-12-18 05:00:02 2012-12-25 08:00:02 7.125 days
##  7 2012-12-25 08:00:02 2012-12-27 02:00:01 1.749988 days
##  8 2012-12-27 02:00:01 2013-01-15 21:00:04 19.7917 days
##  9 2013-01-15 21:00:04 2013-01-26 05:00:01 10.3333 days
## 10 2013-01-26 05:00:01 2013-01-27 23:00:04 1.750035 days
## # ... with 26 more rows
```

```
## ISSUE: Kazkodel viena atspausdina kur lygiai valanda praejo wtf
```

5-8 Uzduotys

5. Atsitiktinai pasirinkite viena faila is Vilniaus.

Is visu failu pavadinimu, sumatchinu Vilniu, tada atsitinktinai pasirenku 1.

```
set.seed(1401445)

# Matchinimas ir samplinimas.
```

```
(file_random_path <- file_path %>%
    .[which(str_match(file_path, "\\w*_") == "vilnius_")] %>%
    sample(1))
```

```
## [1] "vilnius_2013-03-20_11:00:01"
```

Parsisiunciu pati faila.

```
# GitHub url formatas.
file_random_url <-
  str_c(
    "https://raw.githubusercontent.com/vzemlys/eda_2017/master/data/",
    file_random_path
  )

file_random <- read_html(getURL(file_random_url))
```

6. Siame faile bus HTML lenteles. Kiekviena is ju prasideda ir baigiasi tagu <table> ir baigiasi tagu <\table>. Kiek lenteliu yra siame faile?

Viso lenteliu, kurios yra <table> taguose yra daugiau, taciau tik 10 is ju yra informacija apie orus.

```
# Viso lenteliu yra:
file_random %>%
  html_nodes(xpath = '//table')
```

```
## {xml_nodeset (14)}
##  [1] <table id="wrapper" align="center" border="0" cellpadding="0" cells
...
##  [2] <table align="center" border="0" cellpadding="0" cellspacing="0">\n
...
##  [3] <table class="top-menu" align="right" border="0" cellpadding="0" ce
...
##  [4] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [5] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [6] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [7] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [8] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [9] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
## [10] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
## [11] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
## [12] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
```

```
...
## [13] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
## [14] <table class="footer-menu" align="left" border="0" cellpadding="0"
...
```

```
# Lenteles su duomenimis apie orus.
(file_random_nodes <- file_random %>%
    html_nodes(xpath = '//*[@class="weather_box"]//table'))
```

```
## {xml_nodeset (10)}
##  [1] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [2] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [3] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [4] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [5] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [6] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [7] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [8] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
##  [9] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
## [10] <table class="oraiTable" width="660" cellspacing="0" cellpadding="3
...
```

7. Pasirinkite tas lenteles kuriose yra oru prognoze vienai dienai. Parasykite koda kuris sutvarko viena lentele i mazdaug toki pavidala:

8. Sujunkite visas lenteles i viena lentele ir sukurkite stulpelius kuriuose yra miestas ir laikas kada buvo nuskaityta to miesto oru prognoze.

7-8 padariau kartu, idejes i cikla.

```
# html_table sudeda i lenteles, bet prastai.
file_random_raw <- file_random_nodes %>%
  html_table()

# Rezultatu lentele
orai_table_final <- tibble()

# Sutvarko visas lenteles ir sudeda i viena.
for (i in seq_along(file_random_raw)) {
  temp_tbl <- as.tibble(file_random_raw[[i]])
```

```
  temp_tbl_final <- temp_tbl %>%
    # Pasalina iconeliu stulpeli (debesuota, sauleta, etc.).
    select(-X2) %>%
    # Pervadina stulpelius atitinkamais pavadinimais.
    `colnames<-`(sapply(slice(., 2), enc2native)) %>%
    # Pasalina eilute, kuria naudojom pavadinimams sudeti.
    slice(3:n()) %>%
    # Prideda laiko eilutes.
    separate(Laikas, into = c("Pradzia", "Pabaiga"), sep = " - ") %>%
    mutate(Pradzia = as.integer(str_sub(Pradzia, end = 2)),
               Pabaiga = as.integer(str_sub(Pabaiga, end = 2)))

  # Prideda sutvarkyta lentele.
  orai_table_final <- rbind(orai_table_final, temp_tbl_final)
}

# Galutine lentele.
orai_table_final
```

```
## # A tibble: 40 x 6
##    Pradzia Pabaiga Temperatura Vejas     Slegis     Krituliai
##      <int>   <int> <chr>       <chr>     <chr>      <chr>
##  1       5      11 -7°         <U+008A>R 5.4m/s 1004.2 hPa 0.3 mm
##  2       8      11 -7°         <U+008A>R 6.1m/s 1005.7 hPa 0 mm
##  3      11      17 -6°         <U+008A>R 5.7m/s 1007.5 hPa 0.2 mm
##  4      17      23 -5°         <U+008A>R 4.4m/s 1010.8 hPa 0.1 mm
##  5      23       5 -9°         <U+008A>R 3.6m/s 1013.9 hPa 0 mm
##  6       5      11 -11°        <U+008A>R 3.2m/s 1015.8 hPa 0 mm
##  7      11      17 -6°         <U+008A>R 3.2m/s 1018.2 hPa 0 mm
##  8      17      23 -4°         <U+008A>R 3.6m/s 1019.1 hPa 0 mm
##  9      23       5 -8°         <U+008A>R 2.3m/s 1021.5 hPa 0 mm
## 10       5      11 -9°         <U+008A>R 4.2m/s 1021.8 hPa 0 mm
## # ... with 30 more rows
```

9-14 Uzduotys

9. Panaudokite parasyta funkcija vienam failui skaityti, nuskaityti visoms Vilniaus miesto prognozems. Kiek eiluciu yra gautoje galutineje lenteleje?

Naudojama funkcija yra (beveik) viskas kas buvo virsuj sujungta i viena.

```
weather_vilnius <- getWeatherTable("Vilnius", file_path)
head(weather_vilnius)
```

```
## # A tibble: 6 x 8
##   Data       Pradzia Pabaiga Temperatura Vejas     Slegis     Kritu~ Mies~
##   <date>       <int>   <int> <chr>       <chr>     <chr>      <chr>  <chr>
## 1 2012-12-17       5      11 -10°        PR 7.9m/s 1022.3 hPa 1.3 mm viln~
## 2 2012-12-17       8      11 -10°        PR 8.1m/s 1022.6 hPa 0 mm   viln~
```

```
## 3 2012-12-17      11      17 -11°        PR 7.6m/s 1023.3 hPa 1.2 mm viln~
## 4 2012-12-17      17      23 -11°        PR 7.3m/s 1023.1 hPa 0.4 mm viln~
## 5 2012-12-17      23       5 -11°        PR 6.1m/s 1023.6 hPa 0.3 mm viln~
## 6 2012-12-18       5      11 -11°        PR 5.6m/s 1023.7 hPa 0.9 mm viln~
```

```
nrow(weather_vilnius)
```

```
## [1] 93799
```

10. Pritaikykite Vilniaus miesto duomenims nuskaityti skirta funkcija nuskaityti Kauno ir Klaipedos duomenimis. Sudekite viska i viena lentele. Kiek yra duomenu kiekvienam miestui?

Ta pacia funkcija naudoju kiekvienam miestui, po to apjungiu i viena.

```
# Kauno duomenys.
weather_kaunas <- getWeatherTable("kaunas", file_path)
nrow(weather_kaunas)
```

```
## [1] 94032
```

```
# Klaipedos duomenys.
weather_klaipeda <- getWeatherTable("klaipeda", file_path)
nrow(weather_klaipeda)
```

```
## [1] 96319
```

```
# Bendrai.
weather_final <- rbind(weather_vilnius, weather_kaunas, weather_klaipeda)
```

11. Parasykite funkcija kuri skaito Ciurlionio hidrometereologijos stoties duomenis. Jos rezultatas turi buti mazdaug toks:

Beveik identiskas formatas pries tai funkcijai. Nuo Ciulionio failu numetu nereikalingus simbolius su regexpr ir paverciu JSON formatu.

```
weather_ciurlionis <- getWeatherCiurlionis(file_path)
```

12. Pritaikykite sia funkcija nuskaityti visiems Ciurlionio stoties duomenims. Kokia yra vidutine vidurdienio temperatura duomenu fiksuotu laikotarpiu? Atsitiktinai pasirinkite 21 dienu laikotarpi. Suskaiciuokite vidutine dienos ir nakties temperatura.

```
# Vidutine temperatura.
weather_ciurlionis %>%
  summarize(mean(Temp))
```

```
## # A tibble: 1 x 1
##   `mean(Temp)`
##          <dbl>
## 1        -3.61
```

```
# Pasirenkame atsitiktine diena is duomenu.
set.seed(1401445)
start_date <- sample(weather_ciurlionis$Date, 1)
```

```r
end_date <- start_date + days(20)

# Pasirenkame duomenis. Dienos perioda laikysime nuo 8:00 iki 20:00, nakties
- kas lieka.
weather_ciurlionis_dn <- weather_ciurlionis %>%
  # Atenkame duomenis patenkancius i laiko intervala.
  filter((Date >= start_date) & (Date <= end_date)) %>%
  # Pridedame diena/naktis eilute
  mutate(Hour = as.integer(Hour),
         `diena/naktis` = ifelse((Hour >= 8) & (Hour <= 20), "diena",
"naktis"))

# Skaiciuojame vidurkius
weather_ciurlionis_dn %>%
  group_by(`diena/naktis`) %>%
  summarize("Vidurkis" = mean(Temp))
```

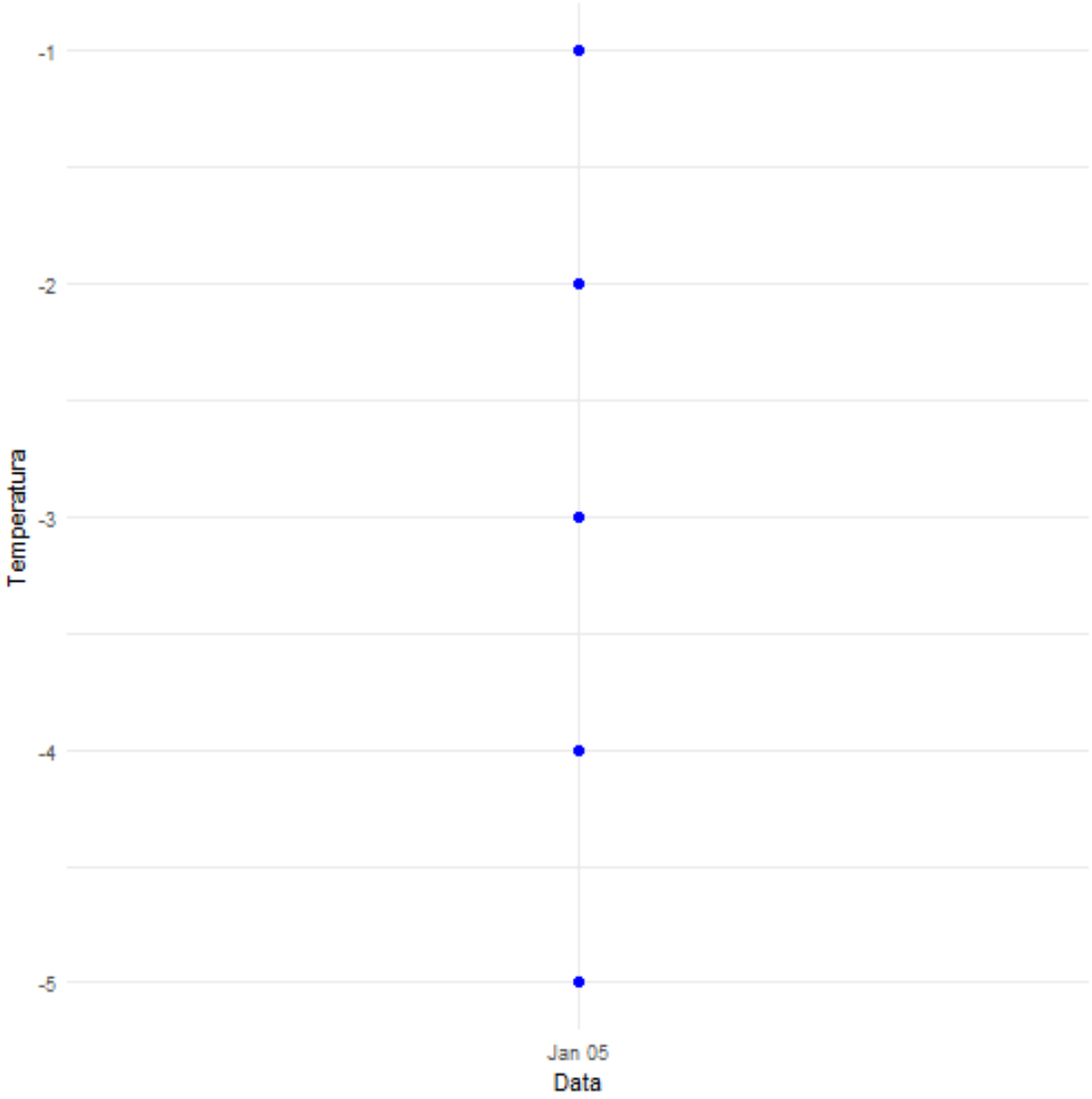
```
## # A tibble: 2 x 2
##   `diena/naktis` Vidurkis
##   <chr>             <dbl>
## 1 diena             -1.02
## 2 naktis            -5.07
```

13. Parašykite funkcija kuri nubrėžia pasirinktos dienos, valandos ir vietos visas surinktas prognozes kaip laiko eilute(s). Atsitiktinai pasirinkite dieną, valandą ir miestą bei nubrėžkite grafiką.

```r
# Atsitiktinai pasirenka parametrus.
set.seed(1401445)
city_rnd <- sample(file_cities, 1)
date_rnd <- weather_final$Data %>% sample(1)
hour_rnd <- sample(0:23, 1)

# I¹filtruoja duomenis pagal miestą, datą ir valandą.
weather_filter <- weather_final %>%
  filter(Miestas == city_rnd,
         Data == date_rnd,
         Pradzia <= hour_rnd,
         Pabaiga >= hour_rnd)

# Jeigu filtravimas nieko nerado, vadinasi pasirinkta valanda yra labai
ankstus rytas (0-2 valanda).
if (nrow(weather_filter) == 0) {
weather_filter <- weather_final %>%
  filter(Miestas == city_rnd,
         Data == date_rnd,
         Pradzia >= 18,
         Pabaiga >= hour_rnd & Pabaiga < 10)
}
```

```r
# Jeigu ir antras filtravimas nieko nerado, vadinasi pasirinkta valanda yra
vėlus vakaras (22+ valanda).
if (nrow(weather_filter) == 0) {
weather_filter <- weather_final %>%
  filter(Miestas == city_rnd,
         Data == date_rnd,
         Pradzia <= hour_rnd & Pradzia >= 18,
         Pabaiga < 10)
}

# Paverčia temperatūrą į skaičių ir nupaišo grafiką.
weather_filter %>%
  mutate(Temperatura = as.numeric(str_match(Temperatura,".\\d{1,2}"))) %>%
  ggplot() +
  geom_point(mapping = aes(Data, Temperatura), color = "blue", size = 2) +
  theme_minimal()
```

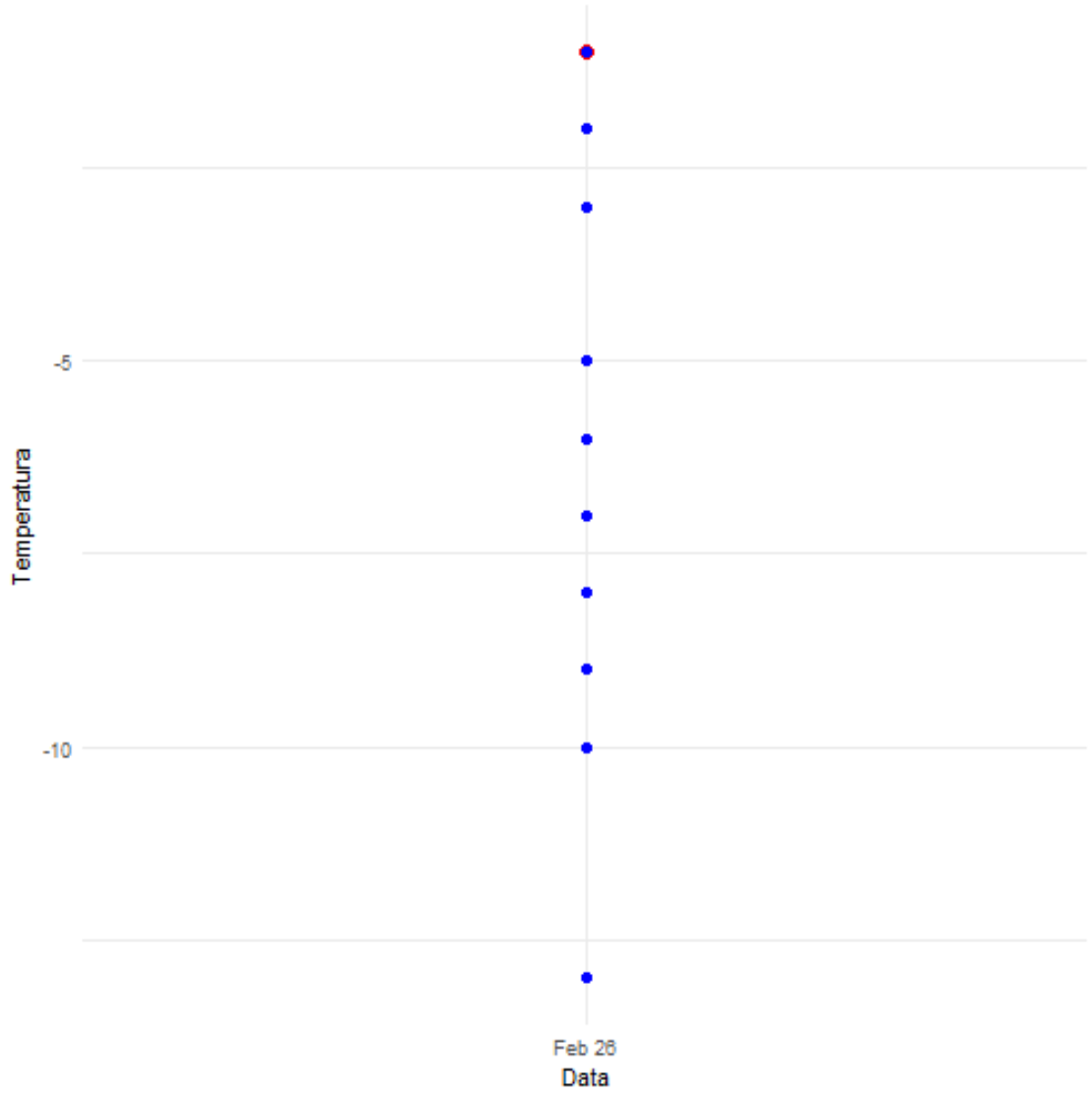
```
## Warning: Removed 81 rows containing missing values (geom_point).
```



14. Pasirinkite atsitiktinai dieną, valandą ir nubrėžkite surinktų Vilniaus prognozių grafiką bei atidėkite tos dienos ir valandos užfiksuotą realią temperatūrą iš Čiurlionio stoties.

```
# Atsitiktinai parenka parametrus (be miesto).
set.seed(1401445)
date_rnd <- weather_final$Data %>% sample(1)
hour_rnd <- sample(0:23, 1)

# Išfiltruoja vilniaus duomenis, tokiu pačiu principu kaip ir 13 užduotyje.
# NUO ČIA KODAS KARTOJASI.

# Išfiltruoja duomenis pagal miestą, datą ir valandą.
weather_filter <- weather_final %>%
  filter(Miestas == "vilnius",
         Data == date_rnd,
         Pradzia <= hour_rnd,
         Pabaiga >= hour_rnd)
```

```r
# Jeigu filtravimas nieko nerado, vadinasi pasirinkta valanda yra labai
ankstus rytas (0-2 valanda).
if (nrow(weather_filter) == 0) {
weather_filter <- weather_final %>%
  filter(Miestas == city_rnd,
         Data == date_rnd,
         Pradzia >= 18,
         Pabaiga >= hour_rnd & Pabaiga < 10)
}

# Jeigu ir antras filtravimas nieko nerado, vadinasi pasirinkta valanda yra
vėlus vakaras (22+ valanda).
if (nrow(weather_filter) == 0) {
weather_filter <- weather_final %>%
  filter(Miestas == city_rnd,
         Data == date_rnd,
         Pradzia <= hour_rnd & Pradzia >= 18,
         Pabaiga < 10)
}

weather_plot <- weather_filter %>%
  mutate(Temperatura = as.numeric(str_match(Temperatura,".\\d{1,2}"))) %>%
  ggplot() +
  geom_point(mapping = aes(Data, Temperatura), color = "blue", size = 2) +
  theme_minimal()

# NUO ČIA NEBESIKARTOJA KODAS.
# Atrenka atitikamus Čiurlionio stoties duomenis.
weather_ciurlionis_filtered <- weather_ciurlionis %>%
  filter(Date == date_rnd,
         Hour == hour_rnd)

# Prideda naują informaciją prie seno grafiko.
weather_plot + geom_point(data = weather_ciurlionis_filtered, mapping =
aes(Date, Temp), color = "red", shape = 1, size = 3)
```


-5
-10
Temperatura
Feb 26
Data