

Duomenų analizės įvadas

1. Dalis

Justas Mundeikis

2019 m. vasario 26 d.

1. Dalies turinys

1 Įvadas į duomenų analizę

- Duomenų analizės menas
- Duomenų analizės epiciklai
- Duomenų analizės klausimai
- Duomenys, matavimo skalės

2 Command line interface

- Intro
- Direktorijos
- CLI komandos

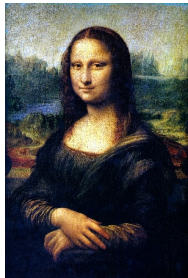
3 Git ir GitHub

- Intro
- Darbas su Git
- Git branch
- Markdown sintaksė

4 Google Scholar

Duomenų analizės menas

1 pav.: To paties matymas kitaip



Duomenų analizės menas

2 pav.: Taip atrodo "duomenys"

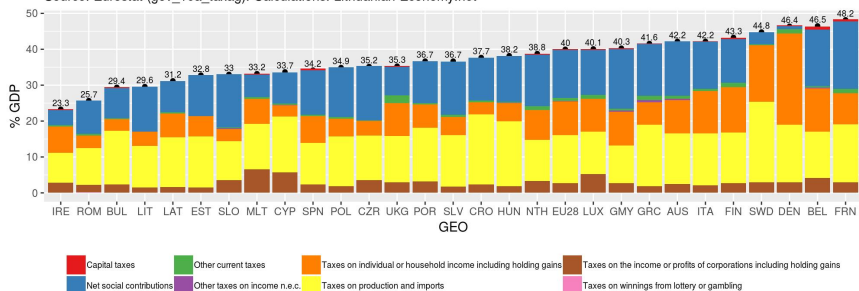
[illegible]

Duomenų analizės menas

3 pav.: Taip atrodo, ką galima padaryti su duomenimis

Main national accounts tax aggregates, % GDP, 2017

Source: Eurostat (gov_10a_taxag). Calculations: Lithuanian-Economy.net



Trumpa diskusija: ar Lietuvos Tax/GDP (29.6 %) yra gerai, ar blogai? Gal verta perimti Airijos modelį?

Duomenų analizės menas

- *"Science is knowledge which we understand so well that we can teach it to a computer; and if we don't fully understand something, it is an art to deal with it."*
Donald Knuth (1974) ([Knuth: Computer Programming as an Art](#))
- Neegzistuoja jokie formalūs aprašymo, kaip reikia atlikti "duomenų analizę"
- Nors yra žinomi tam tikri įrankiai, statistiniai, ekonometriniai metodai, kuriais galima naudotis...
- Kiekvieno "tyrėjo" (ekonomisto, duomenų analitiko, studento...) asmeninių pasirinkimų aibė nulemia atliekamos analizės kokybę bei naudą

Mokslinio tyrimo žingsniai

- **Labai daug skaityti** (Savaitiniai skaitiniai: 3-6 straipsnius per savaitę, žinių (testo) dalis!)
- Išvystyti klausimą / hipotezę
- Nuspręsti kokia metodika bus taikoma
- Parengti duomenų surinkimo procesą (tyrimo protokolas)
- Surinkti duomenis
- Atlikti tiriamąją statistiką
- Atlikti aprašomąją statistiką
- Modeliuoti, atlikti prognozes
- Interpretuoti rezultatus
- Aprašyti tyrimo eigą bei rezultatus

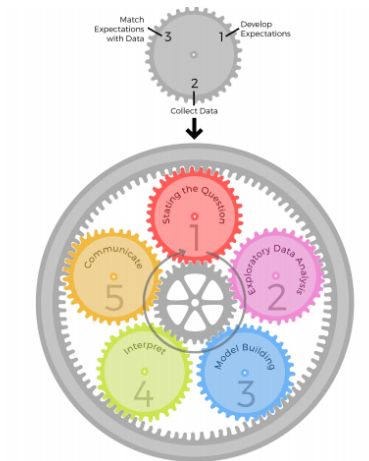
Duomenų analizės žingsnių epikiklai

1 lentelė: The Art of Data Science (Roger D. Peng & Elizabeth Matsui)

Epycles of analysis	Set expectations	Collect information	Revise expectations
Question	Question is of interest to audience	Literature search / Experts	Sharpen question
EDA	Data are appropriate for question	Make exploratory plots of data	Refine question or collect more data
Formal modeling	Primary model answers question	Fit secondary models, sensitivity analysis	Revise formal model to include more predictors
Interpretation	Interpretation of analyses provides a specific & meaningful answer to the question	Interpret totality of analyses with focus on effect sizes & uncertainty	Revise EDA and / or models to provide specific & interpretable answer
Communication	Process & results of analysis are understood, complete & meaningful to audience	Seek feedback	Revise analyses or approach to presentation

Duomenų analizės žingsnių epiklai

4 pav.: The Art of Data Science (Roger D. Peng & Elizabeth Matsui)



6 Klausimų tipai

Remiantis R.Peng ir J.Leek ([Science 2015](#)) egzistuoja 6 klausimų tipai:

- Aprašomieji
- Tiriamieji
- Inferenciniai
- Progozuojamieji
- Pražastinių ryšių
- Mechanistiniai

6 Klausimų tipai

Aprašomieji klausimai:

- Kuriais siekiama gauti duomenų aprašymą, arba charakteristikų santraukas
- Nedaromos jokios išvados ar prognozės, nes patys rezultatai yra išvados per se
- Pvz., Moterų ir vyrų dalis tyrimo imtyje, vidutinis tiriamųjų amžius, vidutinė metinė infliacija, medianinės pajamos ir t.t.
- **LSD šalies rodikliai**
- **LSD statistika vizualiai**

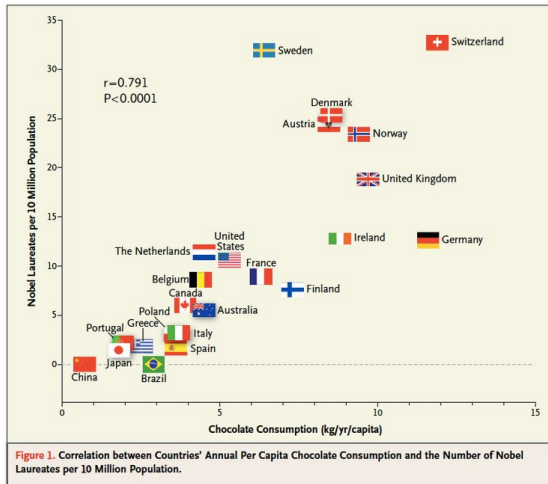
6 Klausimų tipai

Tiriamieji klausimai

- Klausimai, kuriais siekiama nustatyti sąsajas bei trendus
- Padeda rasti kelią kuriuo galima judėti tyrime pirmyn, pvz., generuoti hipotezes
- Dažniausiai tokių klausimų atsakymui braižomi grafikai, padedantys surpasti duomenis
- Tačiau "Correlation does not imply causation" (žr sekanti skaidrė!!!)

6 Klausimų tipai

5 pav.: Šokolado vartojimas ir Nobelio prizai (Franz H. Messerli, M.D., 2012)



6 Klausimų tipai

Inferenciniai klausimai:

- Klausimai, kuriais siekiama atsakyti klausimus apie bendrą populiaciją, tiriant tik imtį
- Pvz., Ekonomikos kurso 1 gr. baigiamasis pažymys 8. Ar visas 1 kursas gavo 8?
- Taikant inferencinę analizę siekiama nustatyti dominantį kiekį bei su prognoze susijusią paklaidą

6 Klausimų tipai

Prognozuojamieji klausimai

- Klausimai, kuriais siekiama "atspėti" ateitį
- Naudojant turimą informaciją apie tam tikrus objektus prognozuoti reikšmes kitiems objektams
- Svarbu: Jeigu X prognozuoja Y nereiškia, kad X iššaukia Y
- Prognozavimo taiklumas priklauso nuo teisingo matuojamų kintamųjų pasirinkimo
- Kuo daugiau duomenų ir kuo paprastesnis modelis!
- <https://fivethirtyeight.com/>
- AMAZON IBM E570

6 Klausimų tipai

Priežastinių ryšių klausimai:

- Klausimai, norint sužinoti, ar pakeitus vieną faktorių, kinta kitas faktorius
- How does a lack of sleep impact memory, problem solving and critical thinking skills amongst college students?
- Reikalingos randomizuotos studijos
- Ekspertimentų galimybė ekonomikos šakoje ribota
- Yra būdų kaip tai apeiti (ekonometrika magistre / PhD)
- Dažniausiai gaunami vidutiniai efektai
- Siekiama atsakyti "ar" bet ne "kaip"

6 Klausimų tipai

Mechanistiniai klausimai

- Klausimai, kuriais siekiama nustatyti "kaip"
- Kaip ir kokie būtent pokyčiai vieno kintamojo keičia daro įtaką kitiems kintamiesiems (fizikos/inžinerijos sritys)

6 Klausimų tipai

- Svarbu suprasti, jog pvz., iškėlus prognozuojamąjį klausimą, tyrimo eigoje bus atsakyti ir į aprašomuosius, tiriamuosius, inferencinius klausimus

Koks yra geras klausimas?

Geras klausimas pasižymi šiomis savybėmis:

- ➊ Klausimas turi būti įdomus tikslinei auditorijai
- ➋ Klausimas dar neturi būti atsakytas
- ➌ Klausimas turi būti logiškas / prasmingas (pagrįstas teorija)
- ➍ Klausimas turi būti atsakomas (netinka: "Kokia yra gyvenimo prasmė? / Ar egzistuoja dievas?"), kitaip tariant, turi egzistuoti duomenys ir metodikos, kurių pagalba būtų galima atsakyti į klausimą
- ➎ Klausimas turi būti labai konkretus
 - Blogas klausimas: ar sveika mityba skatina ilgesnį gyvenimą
 - Geras klausimas: ar 250gr daržovių kasdien suaugusiam asmeniui padidina tikėtiną gyvenimo trukmę 10 metų?
 - Blogas klausimas: kas ekonomikos nuosmukio laikotarpiu nukenčia labiausiai
 - Geras klausimas: Kurioms iš soc grupių: bedarbiai, pensininkai, daugiavaikės šeimos per ekonominę 2008-2009 krizę labiausiai padidėjo rizika patirti santykinį skurdą

Duomenys, matavimo skalės

- Duomenys yra faktai arba skaičiai, kurie yra renkami, analizuojami bei apibendrinami pristatymo ar interpretavimo tikslais
- Duomenys surinkti tam tikro tyrimo metu vadinami duomenų set'u arba duomenų masyvu
- **Elementai** - subjektai, apie kurios renkami duomenys
- **Kintamasis** - elemento charakteristika
- Tyrimo metu surinkti **matavimai** apie visus dominančius elementus ir jų kintamuosius ir yra duomenys / duomenų set'as
- Duomenų set'as vieno elemento vadinamas **obzervacija**

Duomenys, matavimo skalės

Kintamųjų tipas apibrėžia informacijos kiekį slypinti duomenyse, bet kartu ir apriboja galimus taikyti statistinius metodus jų analizei.

- **Kategoriniai** kintamieji:

- **Nominalūs** kintamieji: Lytis, Spalva
Galima tik suskaičiuoti vienetus
- **Ranginiai** kintamieji: Dydžiai S,M,L; Kredito reitingai F - AAA
Juos galima prasmingai suranguoti!

- **Kiekybiniai** kintamieji:

- **Intervaliniai** kintamieji: pažymiai (neturi 0)
+,-, yra prasmingi, bet daugyba, dalyba nėra prasmingi
- **Santykiniai** kintamieji: svoris, ūgis, atstumas (turi 0)
+,-, daugyba, dalyba yra prasmingi

Duomenys, matavimo skalės

- 'Tarpseksiniai' duomenys (angl.: cross-sectional data): vienu ar panašiu metu užfiksuoti skirtingų elementų matavimai: pvz Europos šalių 2018m. BVP €
- Laiko eilučių duomenys (angl.: Time series data): Matavimai surinkti per du ar daugiau laikotarpių vienam elementui
- 'Tarpseksinės' laiko eilutės (n elementų, t laikotarpių, taigi $n \times t$ matavimų)

Big Data

- Lietuvoje dauguma įmonių nelabai supranta ką reiškia "big-data"
- Big-data be AI perteklinis duomenų kaupimas
- Problema su AI - niekas nesupranta AI
- Tačiau su laiku AI keis ir ekonomikos mokslą:
- [Video: AEA AFA Joint Luncheon - The Impact of Machine Learning on Econometrics and Economics](#)
- John Tukey: "The data may not contain the answer. The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data"

Apibendrinant:

- Svarbiausias duomenų analizės / tyrimo aspektas - klausimas!
- Antras pagal svarbumą - duomenys
- Dažnai duomenys apribos arba išlaisvins Jus, bet tik duomenys be klausimo, neišgelbės :D



Command Line interface (CLI)

Kiekviena operacinė sistema turi CLI:

- Windows: Git Bash (), CMD
- Mac/ Linux: Terminal'as

Su CLI galima:

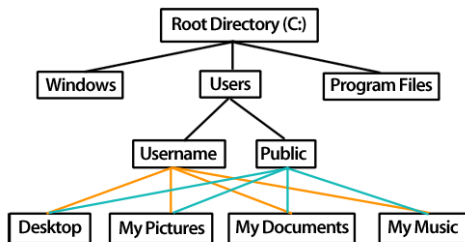
- Naviguoti tarp aplankų (folder'ių)
- Kurti, keisti, naikinti: failus, aplankus, programas
- Startuoti programas

Intarpas GIT Bash instaliavimas

- Nors darbiniai kompiuteriai turi instaliuotą Git Bash, tiems kas neturi:
- <https://git-scm.com/>
- Windows 32/64, Linux žr. komandą
- Perimti teikiamus standartinius siūlymus, nebent antrame Setup lange pasirinkti, jog Git Bash rodytų ir "Additional icons: On the desktop"
- Startuojam Git Bash
- Nuspaudus dešinį pelės mygtuką atsidaro meniu, einame ant *Options..*
 - Looks: pasirenkame Curser - Block (bent jau pradžiai, vėliau pasikeikite į *line*)
 - Keys: Ctrl+Shift+letter shortcuts pastarasis apsirinkimas leidžia daryti naudoti Ctrl+C, Ctrl+V tačiau reikia kartu nuspausti ir Shift!

Direktorijos

- *Directory* yra tiesiog kitas pavadinimas žodžiui aplankas
- Direktorijos kompiuteryje organizuotos kaip medžio šakos
- CLI padeda naviguoti tarp šių direktorijų
- "/" yra *root directory* Linux,
- Windows *root directory* yra C:
- *Root directory* talpina visas kitas direktorijas



Direktorijos

- Startavus matosi daug maž toks tekstas (priklausomai nuo kompiuterio jis gali skirtis!)

```
1 USER@PC MINGW64 ~  
2 $
```

- \$ ženklas (angl.: prompt) reiškia: "gali rašyti komandą"
- Tipinis įrašas susideda iš: "command flag argument"
- Komandos pvz: komanda liepianti atspausdinti kurioje direktorijoje esama: `pwd`

```
1 USER@PC MINGW64 ~  
2 $ pwd  
3 /c/Users/USER
```

Direktorijos

Universiteto kompiuteriuose pwd atsako:
`/c/Users/studentas`

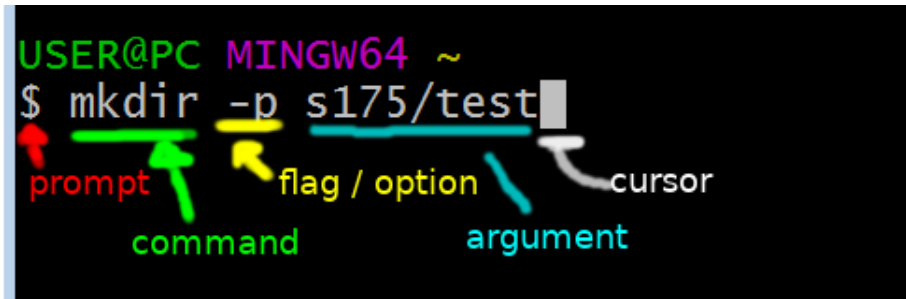
6 pav.: Kur mes esame direktorijų medyje

CLI komandos

- Komanda gali būti iššaukiama su tam tikra programa
- `git init` šitos komandos dabar nenaudot!
- `python get-pip.py` šitos komandos dabar nenaudot!
- flag: tam tikri nustatymai, galimi priklausomai nuo komandos ir visada su "-"
- argument - kiti nustatymai, pakeitimai ar panašūs dalykai
- `git commit -m "this is the initial commit"` šitos komandos dabar nenaudot!
- jeigu flag yra žodis , tada naudojami du brūkšniai --
- `git reset --hard HASH` šitos komandos dabar nenaudot!

CLI komandos

7 pav.: komanda



CLI komandos - echo

- Pirmosios komandos

```
1 USER@PC MINGW64 ~  
2 $ echo "hello world"  
3 hello world  
4 $ echo 'hello world'  
5 hello world  
6 $ echo hello world  
7 hello world  
8 $ echo "hello world"  
9 >
```

- arba šiuo atveju padėti ", tada Enter arba
- CTRL+C
- ESC
- q

Klaviatūros trumpiniai ir CLI komandos

```
1 USER@PC MINGW64 ~  
2 $ echo "hello world, what a beautiful day it is"
```

- Ctrl+A peršoka į eilutės pradžią (HOME)
- Ctrl+E peršoka į eilutės pabaigą (END)
- Ctrl+U ištrina viską į kairę nuo *cursor*
- `clear` arba Ctrl+L išvalo langą
- arba Ctrl+D išjungia CLI

CLI komandos - cd

- `cd` reiškia *change directory*
- `cd` be argumentų sugrąžins į *home directory*
- `cd..` pakels viena direktoriją aukščiau
- su komanda `pwd` pasitikriname kur esame ir periname į *Desktop*
`cd Desktop`. Su komanda `pwd` įsitikiname, kad esame ant *Desktop*

```
1 USER@PC MINGW64 ~
2 $ pwd
3 /c/Users/USER
4
5 USER@PC MINGW64 ~
6 $ cd Desktop
7
8 USER@PC MINGW64 ~/Desktop
9 $ pwd
10 /c/Users/USER/Desktop
```

CLI komandos >

- echo delfi antraštę

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "Seimas proposes a reduction of MPs"
3 Seimas proposes a reduction of MPs
```

- Po argumento naudojant redirektoriaus simbolį > ir parašome į kur pirmą komandą nusiųsti, šiuo atveju sukuriame tekstinį failą pavadinimu delfi.txt

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "Seimas proposes a reduction of MPs" > delfi.txt
```

Ant *Desktop* atsiranda delfi.txt failas (atsidarom failą su *Sublime*)

- Jeigu padarysim taip, perrašysime delfi.txt failą

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "There are more news" > delfi2.txt
```

(atsidarom failą su *Sublime*)

CLI komandos » ir cat

- Jeigu norime ne perrašyti failą, o prisegti vieną eilutę, naudojame `>>`

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "Seimas proposes a reduction of MPs" > delfi.txt
3 $ echo "A topic heatedly discussed in public" >> delfi.txt
```

- `cat` (conCATenate) parodo failo turinį arba apjungia kelis failus

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "There are more news" > delfi2.txt
3
4 USER@PC MINGW64 ~/Desktop
5 $ cat delfi.txt
6 Seimas proposes a reduction of MPs
7 A topic heatedly discussed in public
8
9 USER@PC MINGW64 ~/Desktop
10 $ cat delfi.txt delfi2.txt
11 Seimas proposes a reduction of MPs
12 A topic heatedly discussed in public
13 There are more news
```

CLI komandos - Listing

- `ls` nurodo visus failus ir folderius esančius direktorijoje
- `ls -a` nurodo visus matomus ir paslėptus failus ir folderius
- `ls -al` nurodo visų matomų ir paslėptų failų ir folderių detales
- `ls -rtlh` tas pats kaip `ls -r -t -l -h`
- `ls *.txt`
- `ls --help`

CLI komandos - nematomi failai

- Kai kurie failai yra "nematomi", tai gali būti sisteminiai failai, arba nustatymo failai.
- Pasigaminam paslėptą failą `.gitignore` ir palyginame rezultatus

```
1 USER@PC MINGW64 ~/Desktop
2 $ echo "secrets" > .gitignore
3
4 USER@PC MINGW64 ~/Desktop
5 $ ls
6 ...
7
8 USER@PC MINGW64 ~/Desktop
9 $ ls -a
10 ...
```

CLI komandos - mkdir

- su `rm` ištriname ant Desktop sukurtus failus
- Pasitikriname ar nepalikome nieko ant *Desktop* su `ls -a`

```
1 USER@PC MINGW64 ~/Desktop
2 $ rm delfi.txt
3
4 USER@PC MINGW64 ~/Desktop
5 $ rm delfi2.txt
6
7 USER@PC MINGW64 ~/Desktop
8 $ rm .gitignore
9
10 USER@PC MINGW64 ~/Desktop
11 $ ls -a
12 ...
```


CLI komandos - mkdir

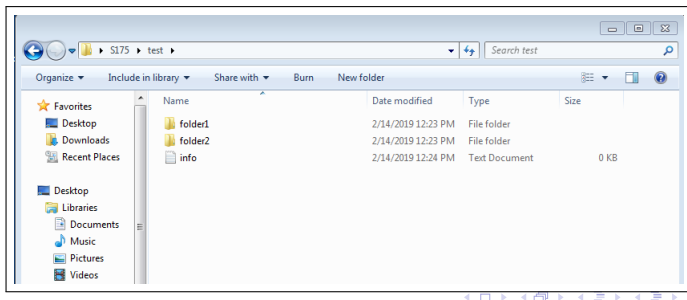
- `mkdir` *make directory* sukuria direktoriją / folderį pvz: "S175"
- `rmdir` *remove directory* ištrina, bet tik, jeigu folderis yra tuščias
- `mkdir -p s175/test` sukuria direktoriją s175 (parent) ir jos viduje direktoriją test.
- Su komanda `cd` pereiname į sukurtą subfolderį test ir su komanda `pwd` įsitikiname ar tikrai esame ten :D

```
1 USER@PC MINGW64 ~/Desktop
2 $ mkdir -p S175/test
3
4 USER@PC MINGW64 ~/Desktop
5 $ cd S175/test
6
7 USER@PC MINGW64 ~/Desktop/S175/test
8 $ pwd
9 /c/Users/USER/Desktop/S175/test
```

CLI komandos - touch

- `touch` sukuria failą
- `touch info.txt`
- Sukuriam dvi direktorijas folder1 ir folder2
- `mkdir folder1`
- `mkdir folder 2`

8 pav.: Windows Explorer



CLI komandos - cp

- `cp` kopijuoja failą

```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ cp info.txt info2.txt
3
4 USER@PC MINGW64 ~/Desktop/S175/test
5 $ cp info.txt folder1
6
7 USER@PC MINGW64 ~/Desktop/S175/test
8 $ cp -r folder1 folder2
9
10 USER@PC MINGW64 ~/Desktop/S175/test
11 $ cp -r folder1 /c/Users/USER/Desktop/S175
```

- `cp -r folder folder` arba `cp -r folder directory` -r reiškia, jog kartu kopijuojamas ir folderio turinys
- Dabar folderyje S175 turime: folder1 ir test

CLI komandos - rm

- `rm` *remove* trina (ne į šiukšliadėžę!!!)
- `rm -r` su direktorijos pavadinimu viskam kas direktorijoje

```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ rm info2.txt
3
4 USER@PC MINGW64 ~/Desktop/S175/test
5 $ rm -r folder2
6
7 USER@PC MINGW64 ~/Desktop/S175/test
8 $ rm -r /c/Users/USER/Desktop/S175/folder1
```

CLI komandos - mv

- `mv` perkelia failą (Cut+Paste), arba pervadina failą

```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ mv info.txt folder1
3
4 USER@PC MINGW64 ~/Desktop/S175/test
5 $ mkdir folderx
6
7 USER@PC MINGW64 ~/Desktop/S175/test
8 $ mv folderx folder2
```

CLI editoriai

- Geriausia naudoti *Sublime*
- *Windows* turi standartinį *Notepad* (tragedija!)

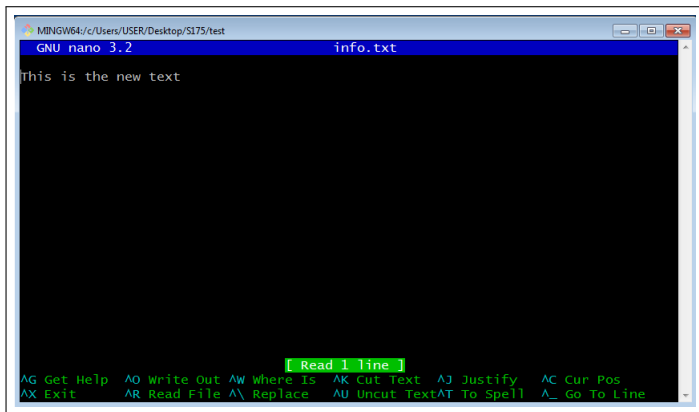
```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ echo "This is the new text" > info.txt
3
4 USER@PC MINGW64 ~/Desktop/S175/test
5 $ notepad info.txt
```

- Kol neuždarytas Notepad , GitBash "laukimo" būsenoje! tad norint dirbti toliau, pirma reikia uždaryti editorių

CLI editoriai

- GitBash turi *Nano*, bet reikia labai įprasti!

9 pav.: Nano editorius



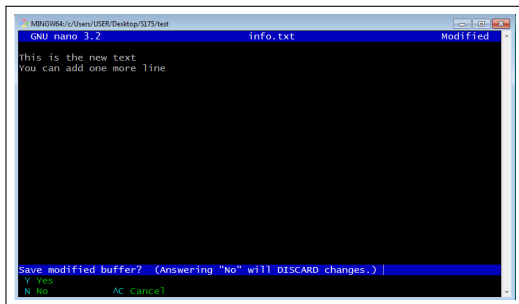
```
MINGW64: c:/Users/USER/Desktop/S175/test
GNU nano 3.2 info.txt
This is the new text

[ Read 1 line ]
AG Get Help  AO Write Out  AW Where Is  AK Cut Text  AJ Justify  AC Cur Pos
AX Exit      AR Read File  AL Replace  AU Uncut Text  AT To Spell  AL_ Go To Line
```

CLI editoriai

- GitBash turi *Nano*, bet reikia labai įprasti!
- Atidarome su Nano
- Įrašome papildomą eilutę
- Uždarome su Ctrl+X, Nano klausia: "Save modified buffer? .."
- Spaudžiama Y klavišą

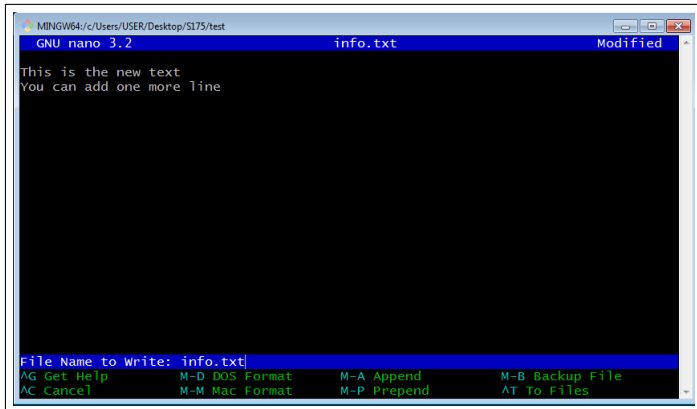
10 pav.: Nano editorius



CLI editoriai

- Nano klausia, ar pavadinimas lieka tas pats?
- Spaudžiam *Enter* klavišą

11 pav.: Nano editorius



Wrap-up

- Iki dabar išmokome pagrindinių komandų, kuriomis galime naviguoti sistemoje
- Taip pat išmokome svarbiausias komandas, kurios leidžia dirbti su failais ir folderiais
- Tam kad būtų švarus S175/test, belieka tik ištrinti test esančius failus:

```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ rm -r folder1
3 USER@PC MINGW64 ~/Desktop/S175/test
4 $ rm -r folder2
5 USER@PC MINGW64 ~/Desktop/S175/test
6 $ rm info.txt
```

VCS - Version Control System

- Kai darbuojatės ir rašote dokumentus, darote juose pakeitimus, dažnai turite 10-20 dokumentų pvz., `bakalaurinis.doc`, `bakalaurinis2.doc`, `bakalaurinis2019_01_02.doc`, `bakalaurinis2019_01_02(1).doc` ir t.t.
- dar panašiai tiek pat `.xls` failų, kiek mažiau skirtingų `.ppt` failų
- Galų gale tai veda link chaoso!
- Todėl labai svarbu, ypač dirbant su tekstiniais dokumentais (ne Word'iniais), pvz., R skriptais, LaTeX failais, turėti vieną failą, bet būti išsisaugojus kartu visas jo ankstesnes formas ir galėti atstatyti ankstesnes jų **versijas**
- Bendradarbiaujant su kitais, taip pat svarbu galėti dalintis turimu dokumentu, kodu, leisti kitiems jį keisti ir galų galų integruoti pakeitimus į motininį failą

Trumpas įvadas į Git

"Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision-control system, it is aimed at speed, data integrity, and support for distributed, non-linear workflows"

<https://en.wikipedia.org/wiki/Git>

Git

"Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency"

<https://git-scm.com/>

- Sukurta Linux kurėjo Linus Torvalds
- Populiariausia VCS
- Viskas išsaugoma lokaliai
- GIT naudojamas naudojant CLI, nors Windows yra ir GUI
- [Download GIT](#)

Git pagrindiniai nustatymai

- Kiekvienas išsaugojimas bus susietas su išsaugotu "user.name" ir "user.email"
- Tai reikia padaryti tik vieną kartą (dirbant su savo PC), arba pasikeisti kaskart prisėdus prie svetimo PC
- Bendradarbiaujant tai padeda atpažinti kas padarė kokius pakeitimus
- Konfiguruokite Git su savo vardu ir pavarde, bei universiteto email

```
1 $ git config --global user.name "Justas Mundeikis"
2 $ git config --global user.email justas.mundeikis@evaf.vu.lt
3 $ git config --global core.pager cat
4 $ git config -l
5 $ git config --global -l
```

GitHub

"GitHub is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project"

<https://en.wikipedia.org/wiki/GitHub>

- "push" ir "pull" tarp lokalių ir internetinių repozitorijų
- suteikia homepage vartotojo repozitorijoms
- GitHub atlieka back-up funkciją lokalioms repozitorijoms
- Leidžia bendradarbiauti, dalintis projektais, gerinti kitų kodą ir t.t.
- GitHub paskyros susikūrimas su VardasPavard (rekomenduotina) ir VU email

Git

- Pirma pasitikriname ar esame .../S175/test ir ar direktorija tuščia
- tada Inicializuojame git

```
1 USER@PC MINGW64 ~/Desktop/S175/test
2 $ pwd
3 /c/Users/USER/Desktop/S175/test
4
5 USER@PC MINGW64 ~/Desktop/S175/test
6 $ ls -a
7 ./ ../
8
9 USER@PC MINGW64 ~/Desktop/S175/test
10 $ git init
11 Initialized empty Git repository in C:/Users/USER/Desktop/S175
    ↪ /test/.git/
12
13 USER@PC MINGW64 ~/Desktop/S175/test (master)
14 $ ls -a
15 ./ ../ .git/
```

- direktorijoje sukuriamas nematomas failas .git

Git = foto sesija

- Git daro "pasirinktos direktorijos nuotraukas"
- **Inicializavimas**, tai tarsi foto kambario pasirinkimas, kuriame gali būti daug "veikėjų"
- `git add failname` yra tarsi "veikėjo" užvedimas ant foto scenos
- `git commit` yra pačios nuotraukos darymas
- Tam kad nuotraukoje nebūtų tam tikrų asmenų galima naudoti sąrašą kuris slepiasi ".gitignore" faile

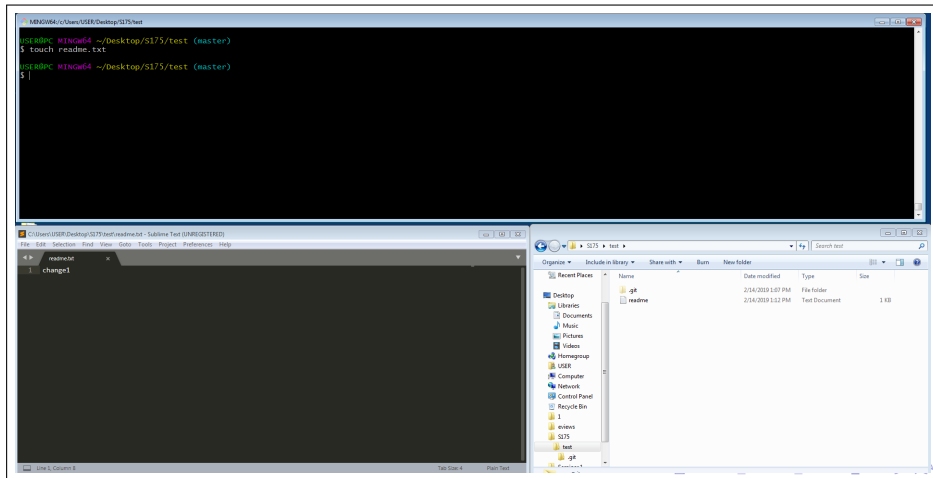


Git

- Su komanda `touch` sukuriame failą `readme.txt`
- `touch readme.txt`
- Su *Sublime* atsidarome `readme.txt`
- Įrašome "change1" ir išsaugome failą "Ctrl+S" (uždaryti *Sublime* nereikia)

Viskas turėtų atrodyti taip

12 pav.: Pirmas žingsnis



Git

- Šis failas egzistuoja *repo* (nes jau sekame folderį o to kai parašėme komandą `git init`), tačiau nėra "sekamas"

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git status
3 On branch master
4
5 No commits yet
6
7 Untracked files:
8   (use "git add <file>..." to include in what will be
9     ↪ committed)
10
11     readme.txt
12
13 nothing added to commit but untracked files present (use "git
14     ↪ add" to track)
```

- Taigi failas `readme.txt` yra "untracked" kitaip tariant ne ant "scenos"

Git

- su `git add readme.txt` įkeliame `readme.txt` į *staging area*
- patikriname statusą:

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git add readme.txt
3
4 USER@PC MINGW64 ~/Desktop/S175/test (master)
5 $ git status
6 On branch master
7
8 No commits yet
9
10 Changes to be committed:
11   (use "git rm --cached <file>..." to unstage)
12
13       new file:   readme.txt
```

- Taigi failas `readme.txt` yra *staged* bet dar ne *committed*

Git

- Su Sublime prirašome vieną eilutę "change2", tada Ctrl+S
- Ką rodo `git status`?

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git status
3 On branch master
4
5 No commits yet
6
7 Changes to be committed:
8   (use "git rm --cached <file>..." to unstage)
9
10      new file:   readme.txt
11
12 Changes not staged for commit:
13   (use "git add <file>..." to update what will be committed)
14   (use "git checkout -- <file>..." to discard changes in
15       ↪ working directory)
16
17      modified:   readme.txt
```

Git

- Norint išsaugoti naujausią versiją turime ją pervesti į *staging area*
- `git add readme.txt`

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git status
3 On branch master
4
5 No commits yet
6
7 Changes to be committed:
8   (use "git rm --cached <file>..." to unstage)
9
10      new file:   readme.txt
```

- "changes to be committed" reiškia, kad galime fotografuoti

Git

- norint perduoti failą repozitorijai (= fotografuoti)

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git commit -m "sukurtas readme.txt, papildytas change1 ir
   ↳ change2"
3 [master (root-commit) ca5eddb] sukurtas readme.txt, papildytas
   ↳ change1 ir change2
4 1 file changed, 2 insertions(+)
5 create mode 100644 readme.txt
6
7 USER@PC MINGW64 ~/Desktop/S175/test (master)
8 $ git status
9 On branch master
10 nothing to commit, working tree clean
```

- "working tree clean" reiškia, jog esame išsaugoję nuotruk su visais naujausiais pakeitimais

Git commit -m "..."

- `git commit -m "..."`
- -m reiškia "message"
- Turime apie 72 ženklus, bet galima ir daugiau
- Aprašas turi būti aiškus, jog būtų prasmingas

13 pav.: <https://m.xkcd.com/1296/>



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.

Git

Toliau keičiame naudojamą failą:

- readme.txt prirašome "change3"
- `git add readme.txt`
- `git commit -m "pridetas change 3"`
- Taigi jau padarėme 2 *commit* (= 2 nuotraukas)

Git log

- Norint žinoti kas, kada ir kaip keitė failą:

- `git log`

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git log
3 commit 030210bb490f45400a706ea69385e8fccc18c65a (HEAD ->
   ↪ master)
4 Author: Justas Mundeikis <mundeikis@gmx.de>
5 Date: Thu Feb 14 13:28:03 2019 -0800
6
7     pridetas change3
8
9 commit ca5eddb5fa492038cb3c3d07f483542502bb1e90
10 Author: Justas Mundeikis <mundeikis@gmx.de>
11 Date: Thu Feb 14 13:24:27 2019 -0800
12
13     sukurtas readme.txt, papildytas change1 ir change2
```

Git log --oneline

- Norint žinoti kas, kada ir kaip keitė failą, bet gauti mažiau info
- `git log --oneline`

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git log --oneline
3 030210b (HEAD -> master) pridetas change3
4 ca5eddb sukurtas readme.txt, papildytas change1 ir change2
```

Git

- Dabar sukursime naują failą *basic.R* ir *readme.txt* pridėsime change4
- `touch basic.R`
- *readme.txt* įrašome "change4", išsaugome
- `git status` parodo, jog pakeistas *readme.txt* failas, ir untracked *basic.R* failas
- su komanda `git add .` stagine visus failus
- galimi kiti variantai : `git add -A` , `git add -u` (pastarasis naudojamas tik update'inti jau sekamus failus)
- tada `git commit -m "pridedamas change4 ir sukuriamas failas basic.R"`

Git

- Kartais yra tam tikri failai, kurių nenorime sekti (pvz duomenys, nereikalingi LaTeX failai ir t.t.)
- Sukuriame duomenų failą
- `touch data.csv`
- `git status` parodo failą kaip *untracked*
- todėl sukuriame failą `touch .gitignore`, kuriame galima surašyti failų pavadinimus, arba failų tipą/galūnes, kurių nenorime trackinti
- *Sublime* įrašome: `*.csv`
- `*` reiškia bet kokį pavadinimą, po kurio seka taškas ir csv, alternatyviai galima specifiikuoti konkretų failą "data.csv"
- "git status" neberodo failo "data.csv" bet rodo ".gitignore", todėl stagine ir commitiname pakeitimus
- `git add .` ir `git commit -m "sukurtas gitignore sarasas"`

Git branch'inimas

- Bazinis scenarijus:
 - A kuria projektą, B nori prisidėti, tačiau ir A dirba tuo pat metu...
 - B atsiskelia atšaką (branch'ina) A projektą, padaro savo pakeitimus ir pateikia A juos sujungti
 - A peržiūri pakeitimus, priima/atmeta

Git branch'inimas

- `git branch NewBranch` sukuria naują atšaką pavadinimu *NewBranch*
- `git checkout NewBranch` išmeta iš "master" į *NewBranch*
- `git checkout master` visada sugrąžina atgal į *master* šaką

```
1 USER@PC MINGW64 ~/Desktop/s175/test (master)
2 $ git branch NewBranch
3
4 USER@PC MINGW64 ~/Desktop/s175/test (master)
5 $ git checkout NewBranch
6 Switched to branch 'NewBranch'
```

- dabar visi pakeitimai vyks tik šioje atšakoje ir nepaveiks *master* šakos

Darbas Git atšakoje

Tarkime dabar asmuo B darbuojasi NewBranch atšakoje tobulindamas projektą

- `touch advanced.R`
- `notepad readme.txt` įrašome papildomą eilutę "change5", tada Ctrl+S
- staginam ir commitinam pakeitimus:
 - `git add .`
 - `git commit -m "sukuriamas advanced.R ir pridemas change5"`
- sugrįžtame į master atšaką su `git checkout master`
- Rezultatas: trūksta *advanced.R* failo, *readme.txt* turi tik 4 įrašus!
- Norint sujungti naują atšaką į *master* `git merge NewBranch`

Git merge

- `git merge NewBranch`

- `git log --oneline`

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git merge NewBranch
3 Updating b8940f7..407d27c
4 Fast-forward
5   advanced.R | 0
6   readme.txt | 3 ++
7   2 files changed, 2 insertions(+), 1 deletion(-)
8   create mode 100644 advanced.R
9
10 USER@PC MINGW64 ~/Desktop/S175/test (master)
11 $ git log --oneline
12 407d27c (HEAD -> master, NewBranch) sukuriamas advanced.R ir
    ↳ pridedmas change5
13 b8940f7 sukurtas gitignore sarasas
14 ebff561 pridedamas change4 ir sukuriamas failas basic.R
15 030210b pridetas change3
16 ca5eddb sukurtas readme.txt, papildytas change1 ir change2
```

Merge problemos

- Kartais nutinka taip, kad atšakoje esantys failai ne visai atitinka su master šakoje esančiais failais, todėl juos jungiant kyla problemų, kurias reikia pašalinti "ranka"
- *master* atšakoje *readme.txt* sukuriame eilutę "change5", Ctrl+S, stage'iname ir commit'iname
- Nueiname į atšaką *NewBranch* ir ten esančiame *readme.txt* sukuriame "change7", stage'iname ir commit'iname
- grįžtame į *master* atšaką `git checkout master`
- ir bandome sujungti `git merge NewBranch`

Merge problemos

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git merge NewBranch
3 Auto-merging readme.txt
4 CONFLICT (content): Merge conflict in readme.txt
5 Automatic merge failed; fix conflicts and then commit the
   ↪ result.
```

- Git negalėjo automatiškai sutvarkyti failų (kaip pirmą kartą), nes dabar toje pačioje readme.txt failo vietoje vienur yra "change6" kitur "change7". Todėl atsidarome readme.txt failą ir tvarkome patys

Konfliktinio failo tvarkymas

- Atsidarius readme.txt matome

14 pav.: Merge problema

The screenshot shows a Windows environment. The top window is a terminal with the following text:

```
USER@PC MINGW64 ~/Desktop/s175/test (master)
$ git merge NewBranch
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
USER@PC MINGW64 ~/Desktop/s175/test (master|MERGING)
$ |
```

The bottom window is the Sublime Text editor, open to the file `readme.txt`. The content of the file is as follows:

```
1 change1
2 change2
3 change3
4 change4
5 change5
6 <<<<<< HEAD
7 change6
8 =====
9 change7
10 >>>>> NewBranch
11
```

The editor's interface shows two tabs: `readme.txt` and `gitignore`. The right sidebar displays the 'Recent Places' and 'Include in library' sections, with the 'Name' column listing `git`, `advanced.R`, `basic.R`, `data.csv`, and `readme`.

- `<<<<<< HEAD` yra tai kas yra aktyvioje atšakoje
- `>>>>> NewBranch` yra kas ateina iš sujungiamos atšakos
- atskirta `=====`

Konfliktinio failo tvarkymas

- Sutvarkome failą, taip kaip norime

```
1 change1
2 change2
3 change3
4 change4
5 change5
6 change6
7 change7
```

- Išsaugome pakeitimus Ctrl+S, tada

- `git commit -a -m "sujungtas failas is NewBranch bei pasalintas konfliktas"`

- `git commit -a -m "..."` galima naudoti, jeigu nėra sukurtų naujų failų

- Yra papildomų įrankių, kurie padeda atlikti merge'inimo darbus, nes dažniausiai konfliktų visada bus

Git atsatatymas 1

- Imituojame, jog dirbome ir sukūrėme gerą failą ir paskui jį papildėme blogu:

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ echo "sukuriamas geras failas" > svarbus.txt
3 $ git add .
4 $ git commit -m "sukurtas svarbus.txt"
5 $ echo "12 nakties pridirbau nesamonių" >> svarbus.txt
6 $ git commit -a -m "pridirbta nesamone"
7 $ git log --oneline
8 78839b7 (HEAD -> master) pridirbta nesamone
9 9547650 sukurtas svarbus.txt
10 2a982aa merge klaida panaikinta, change6 ir change7
11 26e6ae0 (NewBranch) sukurtas change7
12 8562e69 sukurtas change6
13 407d27c sukuriamas advanced.R ir pridedamas change5
14 b8940f7 sukurtas gitignore sarasas
15 ebff561 pridedamas change4 ir sukuriamas failas basic.R
16 030210b pridetas change3
17 ca5eddb sukurtas readme.txt, papildytas change1 ir change2
```

Git atsatatymas

- Norint susigrąžinti failą į praėjusią "gerą" būseną, reikia susirasti "blogo" commit hashą mano atveju (žr. viršuje) 78839b7
- `git log --oneline`
- komanda `git revert HASH` atstato pasirinktą versiją, versijos su "12 nakties..." nebėra!
- komanda `git revert 78839b7`
- po įvedimo atsiranda *message* langas (atitinka -m "..."), nes revert'inimas yra naujas commit

Perspėjimas

- Priklausomai nuo to, koks yra standartinis editorius priskirtas git, atsidaro arba Nano editorius, arba VIM editorius
- Kadangi VIM papildomas ir nebūtinai reikalingas apsunkinimas, šioje vietoje pridedu tik nuorodą į youtube video (8min) kaip dirbti su VIM

- Mano siūlymas:

- `git config --global core.editor nano.exe`

- tada revertinant atsidarys Nano, arba

- `git config --global core.editor absadr` kur vietoj absadr įrašomas `sublime_text.exe` su absoliučiu adresu

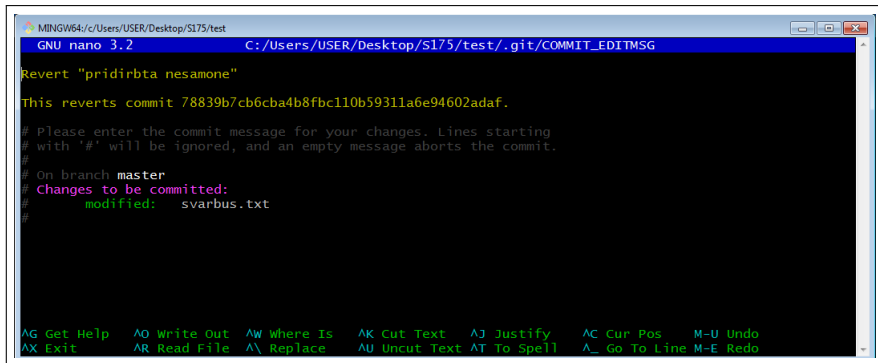
- PVZ

```
git config --global core.editor /c/Users/USER/Download/sublime/sublime_text.exe
```

- tada atsidaro Sublime langas!

Git atstatymas

15 pav.: Revert atidaro message su nano



The screenshot shows a terminal window with the title bar "MINGW64/c:/Users/USER/Desktop/S175/test". The terminal content is as follows:

```
GNU nano 3.2 C:/Users/USER/Desktop/S175/test/.git/COMMIT_EDITMSG
Revert "pridirbta nesamone"

This reverts commit 78839b7cb6cba4b8fbc110b59311a6e94602adaf.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   svarbus.txt
#

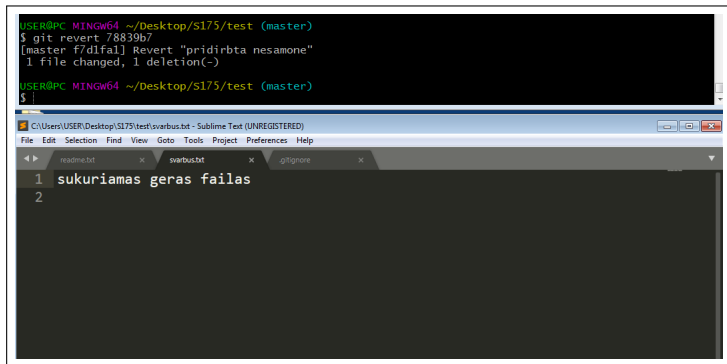
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo
```

- Spaudžiame Ctrl+X ir vėl atsiduriame normaliaime Git

Git atstatymas

- Tik jau blogojo įrašo *Sublime* neberodo

16 pav.: Po revert



The screenshot shows a terminal window and a Sublime Text editor. The terminal window displays the following commands and output:

```
USER@PC MINGW64 ~/Desktop/s175/test (master)
$ git revert 78839b7
[master f7d1fa1] Revert "pridirbta nesamone"
1 file changed, 1 deletion(-)

USER@PC MINGW64 ~/Desktop/s175/test (master)
$
```

The Sublime Text editor window shows the file `svarbus.txt` with the following content:

```
1 sukuriamas geras failas
2
```

Spaudžiame Ctrl+X ir vėl atsiduriame normaliaame Git

Git reset

- Atsargiai!
- Tarkime jus labai daug darbavotės, turite n commit padarę ir supratote, kad pirmas commit buvo geras, o po to viskas ne.
- `git revert HASH` palieka buvusias versijas, bet šiuo atveju, jos nebereikalingos
- `git reset --hard HASH` komanda padaro hard-reset, t.y. resetina į pasirinktą commitą, tačiau ištrina viską, kas buvo daryta po to!
- revert'inti `git reset --hard HASH` neįmanoma, priešingai nei paties revert.
- Taigi geriau su hard-reset nesišaist :D

Git branch ištrynimas

- Mums nebereikia NewBranch atšakos
- Kiek šakų turime patikrinti galima su `git branch`. * parodo kur esame
- Nereikalingą šaką ištrinti galime su komanda `git branch -d Newbranch`
-d = delete

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ git branch
3   NewBranch
4 * master
5
6 USER@PC MINGW64 ~/Desktop/S175/test (master)
7 $ git branch -d NewBranch
8 Deleted branch NewBranch (was 26e6ae0).
9
10 USER@PC MINGW64 ~/Desktop/S175/test (master)
11 $ git branch
12 * master
```

Git remote

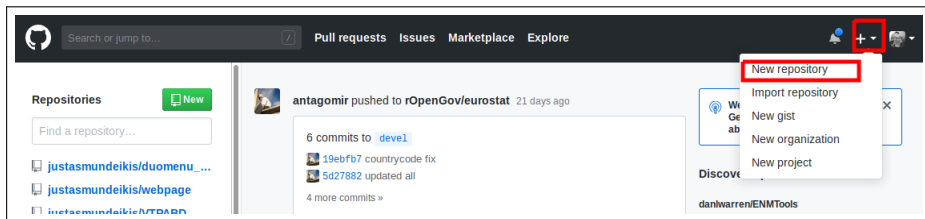
- norint žinoti, ar lokali repozitorija yra susieta su nuotoline repozitorija (pvz Github, Bitbucket ar pan)
- `git remote`
- Jeigu komanda neišmeta jokios informacijos, reiškia lokali *repo* nesusieta su nuotoline *repo*

Git remote

- Nueiname kiekvienas į savo GitHub ir ten susikuriame repo:
 - pavadinimas: test
 - Description paliekam tučią
 - NE inicializuojame su README!!!!
- keliaujame į savo "test" direktoriją
- `git remote add origin HTTPS`
- `git push -u origin master`

Git remote

17 pav.: Nuotolinės repo kūrimas @GitHub: spaudžiam and + ir tada "New repository"



Git remote

18 pav.: Nuotolinės repo kūrimas @GitHub: pavadinimas :test, Initialize paliekam tuščią ir spaudžiam "Creat repository"

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history.

Owner: justasmundeikis

Repository name *: test

Great repository names are short and memorable. Need inspiration? How about [studious-octo-funicular?](#)

Description (optional)

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

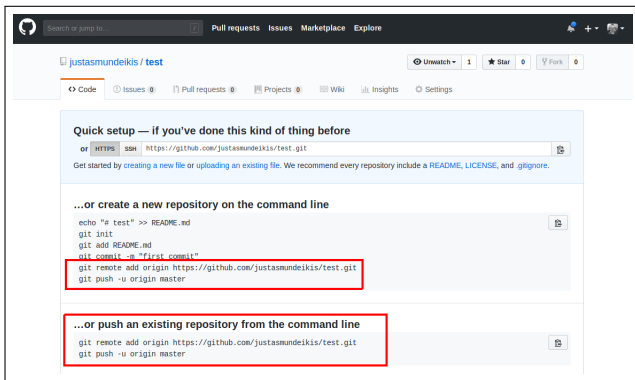
☒ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

Git remote

19 pav.: Nuotolinės repo kūrimas @GitHub: šis langas pasako, kaip lokalinę repo sujungti su nuotoline...



Git remote

- po `git push` reikės įrašyti savo email (GitHub) ir pop-up lange savo passwordą
- po `git remote -v` parodo daugiau info apie remote repo

```
1 $ git remote add origin https://github.com/justasmundeikis/  
   ↪ test.git  
2 $ git remote -v  
3 origin https://github.com/justasmundeikis/test.git (fetch)  
4 origin https://github.com/justasmundeikis/test.git (push)  
5 $ git push origin master  
6 Username for 'https://github.com': vardas.pavarde@stud.vu.lt  
7 Enumerating objects: 32, done.  
8 Counting objects: 100% (32/32), done.  
9 Delta compression using up to 2 threads  
10 Compressing objects: 100% (21/21), done.  
11 Writing objects: 100% (32/32), 2.55 KiB | 55.00 KiB/s, done.  
12 Total 32 (delta 8), reused 0 (delta 0)  
13 remote: Resolving deltas: 100% (8/8), done.  
14 To https://github.com/justasmundeikis/test.git  
15 * [new branch]      master -> master
```

Git remote

20 pav.: Su F5 refresh'inus GitHub puslapį matome, jog vietinio folderio turinys visas perkeltas į GitHub

The image shows two side-by-side windows. The left window is a GitHub repository page for 'justasmundeikis / test'. It displays 11 commits, 1 branch, and 0 releases. The 'master' branch is selected. The commit history shows a series of changes to files like .gitignore, advanced.R, basic.R, data.csv, readme.txt, and svarbus.txt. The right window is a local file explorer showing the contents of the 'test' folder, which matches the files listed in the GitHub commit history.

GitHub Repository: justasmundeikis / test

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Latest commit 775f1a3 34 minutes ago

File	Commit Message	Time
.gitignore	sukurtas .gitignore sąrašas	an hour ago
advanced.R	sukuriamas advanced.R ir pridėdmas change5	an hour ago
basic.R	pridedamas change4 ir sukuriamas failas basic.R	an hour ago
readme.txt	merge klaida panaikinta, change6 ir change7	an hour ago
svarbus.txt	Revert "pridėbta nesamone"	34 minutes ago

Local File Explorer: test

Name	Date modified	Type	Size
git		File folder	
advanced.R	2/14/2019 2:50 PM	Text Document	1 KB
basic.R	2/14/2019 1:35 PM	R File	0 KB
data.csv	2/14/2019 1:54 PM	R File	0 KB
readme	2/14/2019 1:33 PM	CSV File	0 KB
readme.txt	2/14/2019 2:10 PM	Text Document	1 KB
svarbus	2/14/2019 2:23 PM	Text Document	1 KB

- Sveikinu su pirmu išsiuntimu savo darbo į internetą
- Suprantama, galima pirma sukurti repo @GitHub ir tada sukurtą repo klonuoti į savo PC tai, sekantis žingsnis
- Klonuoti galima ir "svetimas" repo

Git repo klonavimas

- Kiekvienas į savo GitHub paskyroje susikuriame repo:
 - Pavadinimas: test2
 - Description paliekam tuščią
 - inicializuojame su Readme.md

21 pav.: Sukuriama nauja repo "test2"

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history.

Owner: justasmundeikis

Repository name: test2

Great repository names are short and memorable. Need inspiration? How about [vigilant-spork?](#)

Description (optional)

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☒ Initialize this repository with a README
We will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

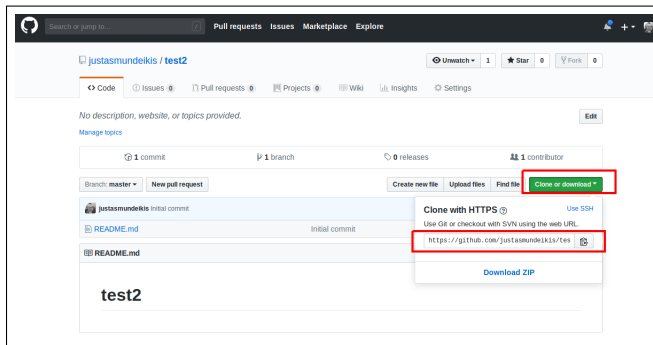
Add .gitignore: None Add a license: None

Create repository

Git repo klonavimas

- GitHub nusikopijuojuame sukurtos repo HTTPS adresą

22 pav.: Nukopijuojamas repo HTTPS



Git repo klonavimas

- Git Bash lange pakylame viena direktorija aukščiau, įsitikiname ar esame S175 folderyje ir klonuojame nusikopijuotą adresą:

```
1 USER@PC MINGW64 ~/Desktop/S175/test (master)
2 $ cd ..
3
4 USER@PC MINGW64 ~/Desktop/S175
5 $ pwd
6 /c/Users/USER/Desktop/S175
7
8 USER@PC MINGW64 ~/Desktop/S175
9 $ git clone https://github.com/justasmundeikis/test2.git
10 Cloning into 'test2'...
11 remote: Enumerating objects: 3, done.
12 remote: Counting objects: 100% (3/3), done.
13 remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
14 Unpacking objects: 100% (3/3), done.
```


Git repo klonavimas

- Klonuotoje repo, sukursime failą, ir jį nusiųsime atgal į GitHub"
- pareikalavus įrašome github username ir userpassword
- GitHub atnaujina (F5) ir voila, failas info.txt yra remote repozitorijoje (&&) apjungia dvi komandas vienoje eilutėje

```
1 USER@PC MINGW64 ~/Desktop/S175
2 $ pwd
3 /c/Users/USER/Desktop/S175
4
5 USER@PC MINGW64 ~/Desktop/S175
6 $ cd test2
7
8 USER@PC MINGW64 ~/Desktop/S175
9 $ echo "test2 klonavimas ir siuntimas i Github" >about.txt
10
11 USER@PC MINGW64 ~/Desktop/S175
12 $ git add . && git commit -m "sukurtas about.txt"
13
14 USER@PC MINGW64 ~/Desktop/S175
15 $ git push
```

Git repo klonavimas

- Sveikinu, sukūrėte GitHub repo
- Ją klovanote
- Pakeitėte vietinėje repo failus
- Ir juos pushinote į Github!
- Atnaujine Github, pamatysite savo naujai sukurtą failą!

Git repo klonavimas

- Dabar klonuokite dar vieną folderį:
- Tačiau pirma įsitikinite, ar esate S175 folderyje, jeigu ne, pereikite (žr sekanti skaidrė)
- ```
git clone https://github.com/justasmundeikis/duomenu_analizes_ivadas.git
```
- Sveikinu, atsisiuntėte mano paskaitų medžiagą
- Norint žinoti ar nėra jokių paskaitos medžiagų atnaujinimo, galima nuėjus į patį folderį įrašyti komandą
- ```
git pull
```

Git repo klonavimas

```
1 USER@PC MINGW64 ~/Desktop/S175
2 $ pwd
3 /c/Users/USER/Desktop/S175
4
5 $ git clone https://github.com/justasmundeikis/
   ↳ duomenu_analizes_ivadas.git
6 Cloning into 'duomenu_analizes_ivadas'...
7
8 USER@PC MINGW64 ~/Desktop/S175
9 $ ls
10 duomenu_analizes_ivadas/  test/  test2/
11
12 USER@PC MINGW64 ~/Desktop/S175
13 $ cd duomenu_analizes_ivadas/
14
15 USER@PC MINGW64 ~/Desktop/S175/duomenu_analizes_ivadas (master)
16 $ git pull
17 Already up to date.
```

Markdown sintaksė

- GitHub sukuriant repo, ją galima inicijuoti su readme.md
- .md reiškia, jog tai yra markdown formatas
- Markdown is a lightweight markup language with plain text formatting syntax. Its design allows it to be converted to many output formats, but the original tool by the same name only supports HTML. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor. (<https://en.wikipedia.org/wiki/Markdown>)
- Labai trumpa pagalba dėl formatavimo:
<https://commonmark.org/help/>
- Vėliau mes susipažinsime su RMarkdown

Google Scholar

- Viskas prasideda nuo daug skaitymo, tarkime aptinkate "The Economist" straipsnį apie "Brexit"
- Nusprendžiate kad esė tema galėtų būti "Kaip Brexit paveiks Lietuvos ekonomiką"
- Dabar reikia eiti ieškoti mokslinių straipsnių, geriausiai tam tinka...
- <https://scholar.google.lt/>
- Funkcijos
 - Laikotarpis
 - Kurti įspėjimą
 - citavimas
 - Cituoja
 - Visos versijos
 - AND OR
 - Išplėstinė paieška
- Dabar galime kartoti temos pavadinimo / klausimo epiciklą

Literatūra

- Apie duomenų analizę R.D. Peng & E.Matsui: "The Art of Data Science" 1-3 skyriai
- CLI patarčiau šią mokamąją medžiagą: [Learn Enough Command Line to Be Dangerous](#)
- Git patarčiau šią mokamąją medžiagą: [Learn Enough Git to Be Dangerous](#)
- Youtube: ieškant "Git Basics"

Namų darbai

- Namuose instaliuoti visas reikalingas programas (Git, Sublime)
- Pakartoti skaidrėse praeitus žingsnius
- Išspręsti Seminar 1 užduotis
- VU VMA įkelti savo Seminar 1 dokumento prašomas nuorodas
- Reguliariai patikrinti, ar nėra VMA įkeltų straipsnių skaitumui, jeigu yra, perskaityti, nes apie savaitinius skaitinius gali būti klausiama teste!!!