Contents

Apžvalga

2020-09-15

Temos:

- Kurso struktūra
- Įrankių apžvalga
  - Komandinė eilutė
  - Anaconda
  - Git
  - Teksto rašymo programos
  - Markdown
  - Latex
  - Dokumentų eksportavimas
  - Paieška informacijos internete

Kurso planas

1. Įvadas. Duomenų analizės įrankių ekosistema.
2. Python pagrindai.
3. Duomenų radimas ir nuskaitymas.
4. Duomenų vaizdinimas.
5. Duomenų tvarkymas.
6. Baziniai algoritmai ir signalo parametrai.
7. Eksperimento kontrolė.

Įvertinimas

Semestro metu renkamas kaupiamasis balas, vidurkis skaičiuojamas iš keturių atsiskaitymų. Atsiskaitymo metu naudojantis pratybų ir laboratorinių darbų aprašais sprendžiamos duomenų analizės užduotys

Jei bendras kaupiamasis balas yra mažesnis nei 5 studentas (-ė) privalo laikyti egzaminą raštu. Jei bendras kaupiamasis balas yra 5 ir daugiau, bet netenkina studento, galima laikyti egzaminą raštu, tada galutinis įvertinimas toks, koks yra egzamino įvertinimas.

Atsiskaitymų vertinimas

- Komentarai aprašantys sprendimą
- Pritaikyti metodai
- Pavaizdavimas rezultatų
- Rezultatas
- Kodas

Bioduomenų analizės tikslai

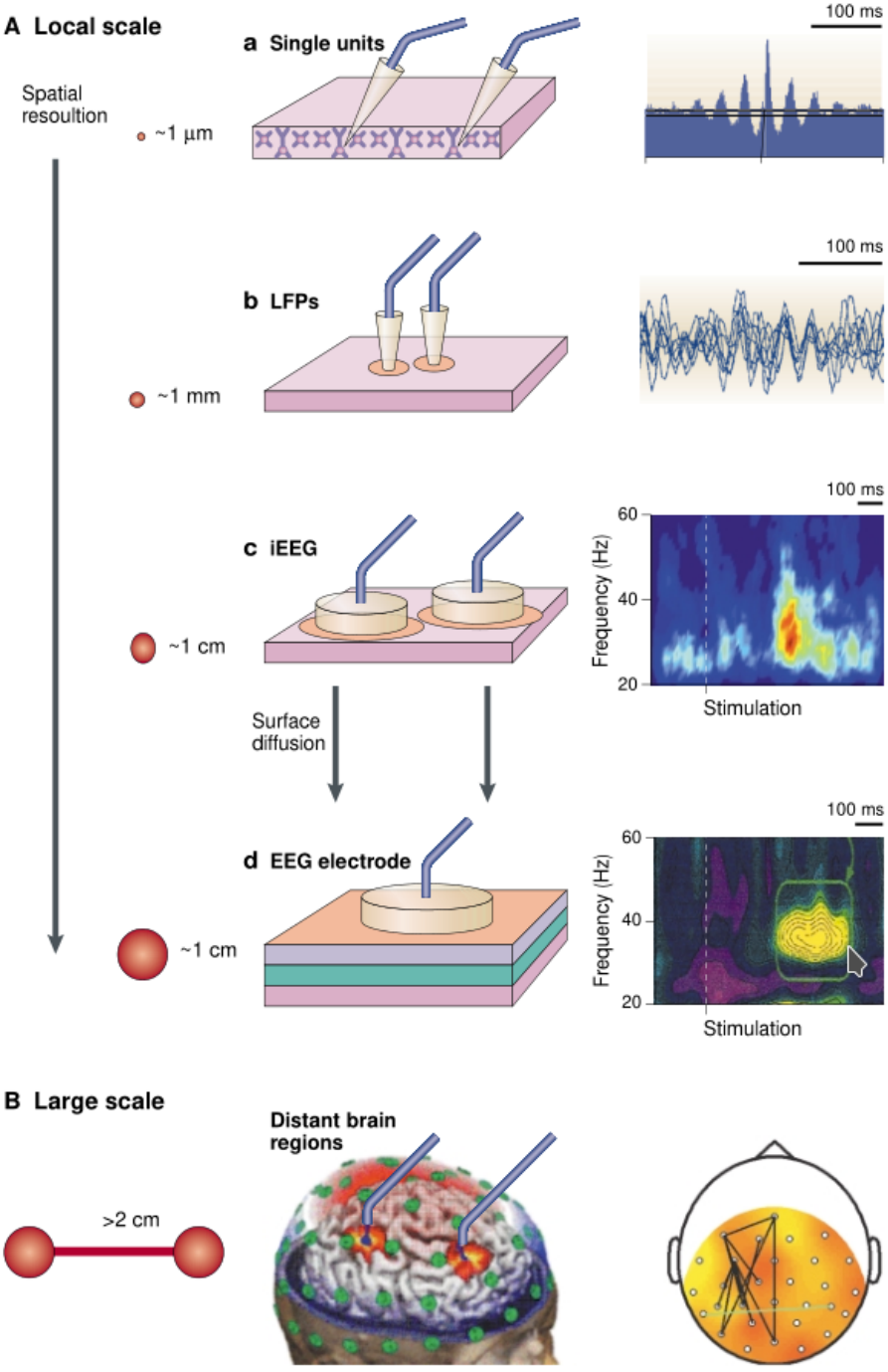
- Informacijos surinkimas -- matavimai siekiant interpretuoti sistemą
- Diagnozė -- aptikimas patologijos, sutrikimo
- Monitoringas -- gavimas periodinės informacijos apie sistemą
- Terapija ir kontrolė -- keitimas sistemos elgesio remiantis praeitais žingsniais
- Įvertinimas progreso -- kokybės kontrolė, efektas gydymo

Matavimų ir analizės sunkumai

- Signalo variabilumas
- Signalo pasiekimas
- Sąveika fiziologinių sistemų
- Matavimo prietaisų įtaka matavimo sistemai
- Fiziologiniai artefaktai
- Signalo silpnumas, įtaka pašalinių stipresnių signalų
- Tiriamųjų saugumas

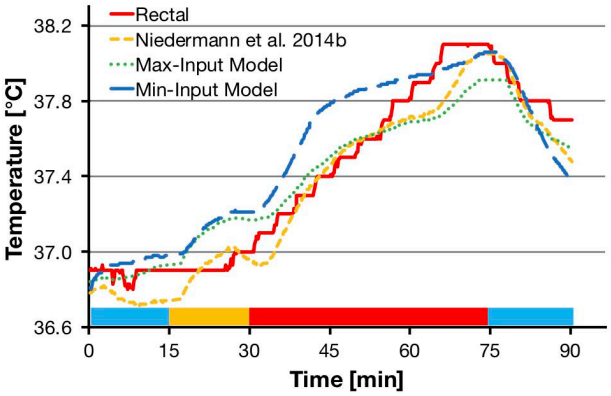
Bioduomenų pavyzdžiai

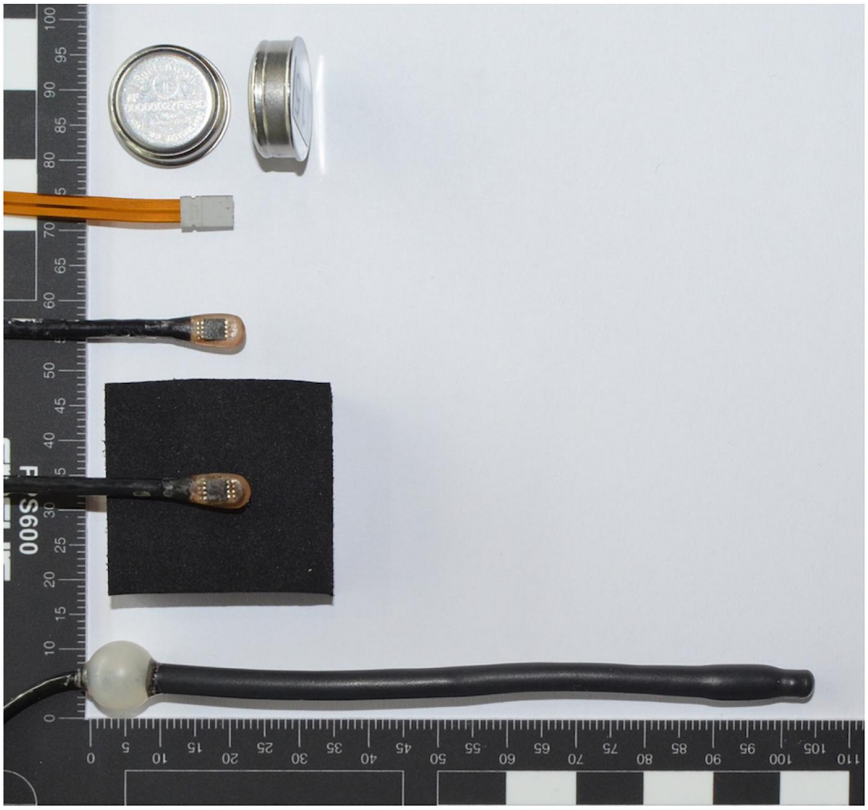
Smegenų neuronų matavimai skirtingose skalėse



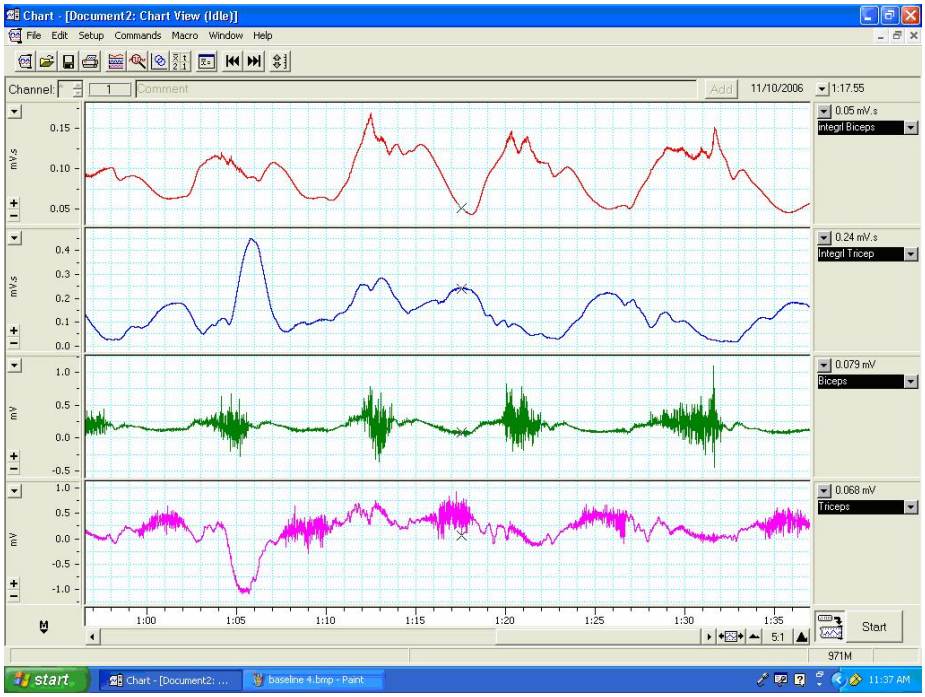
Temperatūra

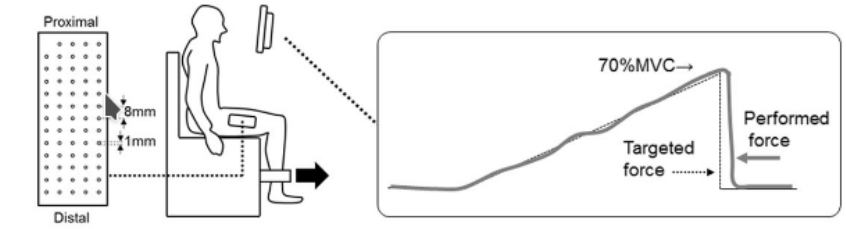
- Kūno temperatūra
- Aplinkos temperatūra
- Objekto temperatūra

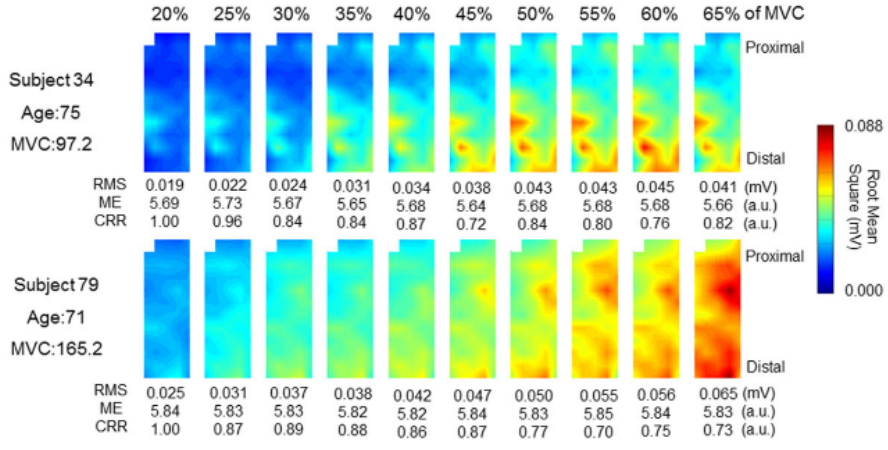




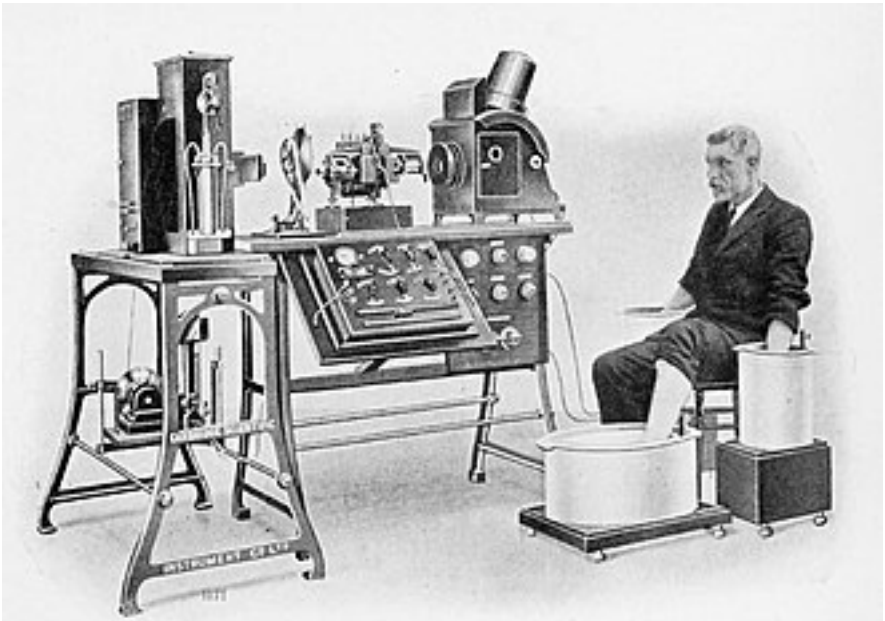
Elektromiograma (EMG) - Raumenų aktyvumo tyrimas



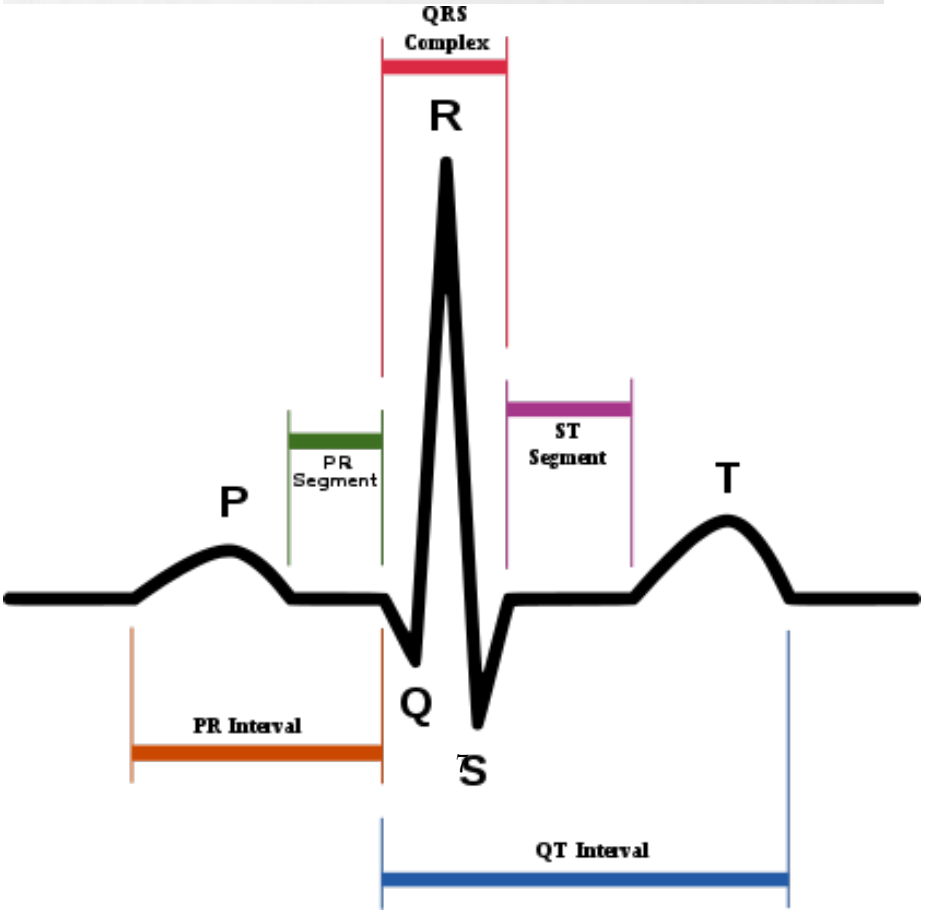




Elektrokardiograma (ECG)

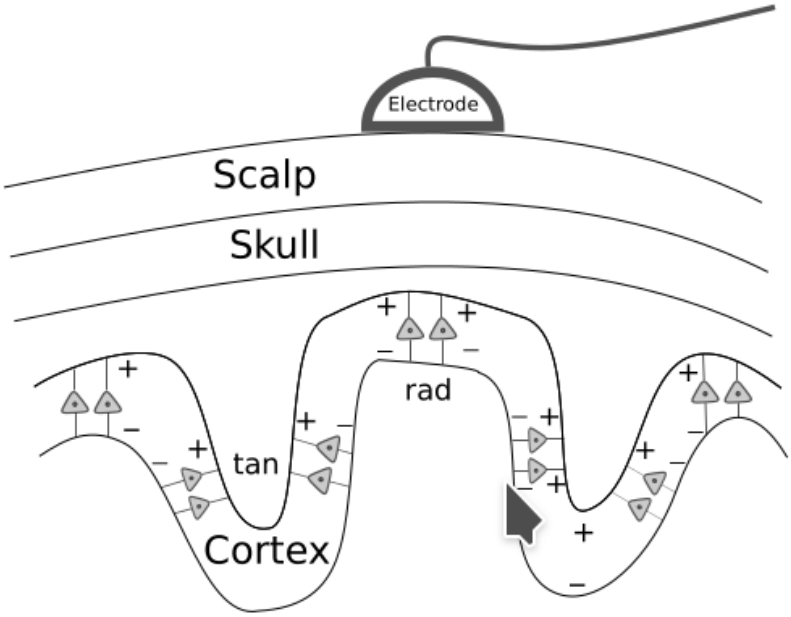


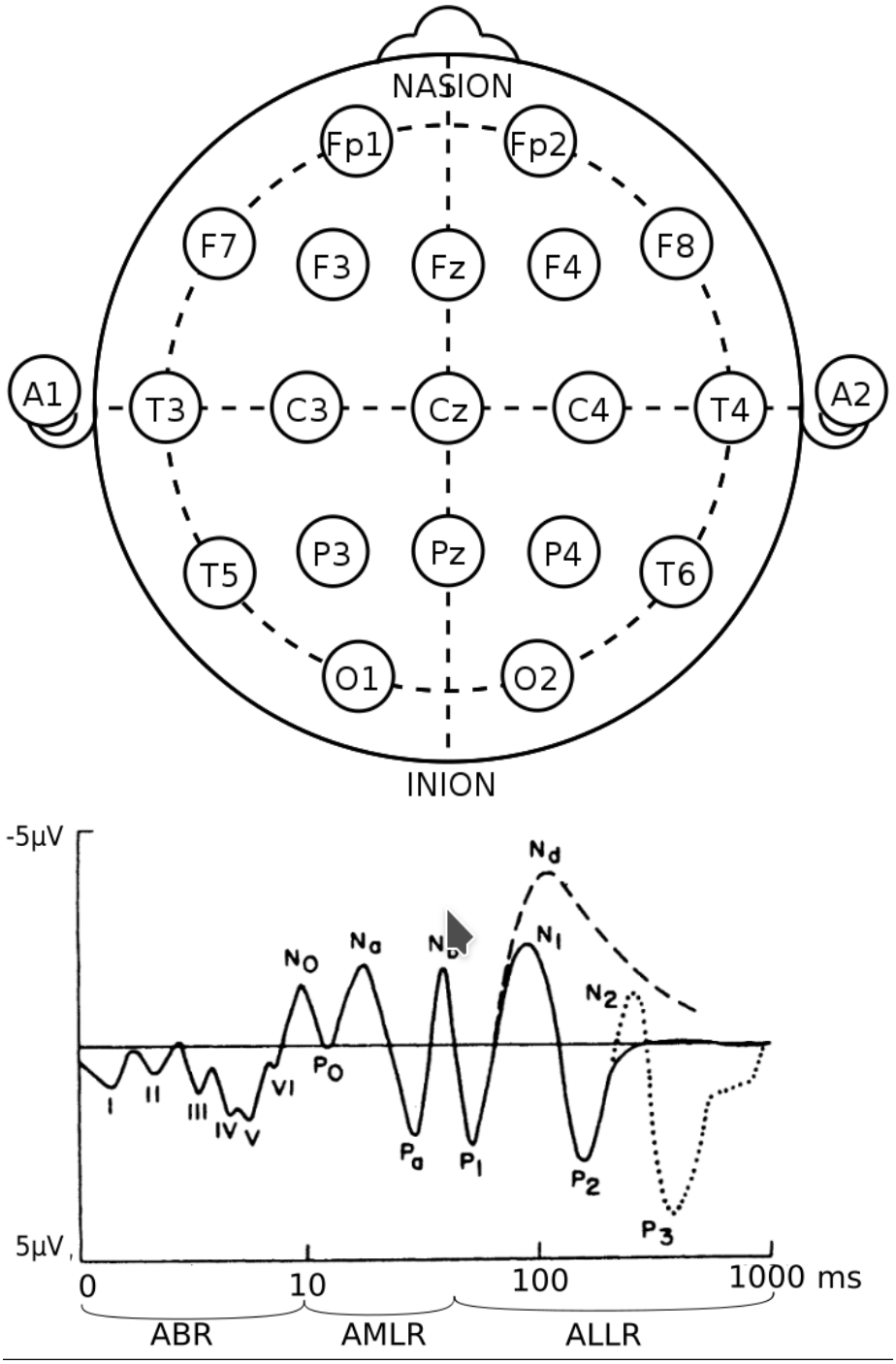
PHOTOGRAPH OF A COMPLETE ELECTROCARDIOGRAPH, SHOWING THE MANNER IN WHICH THE ELECTRODES ARE ATTACHED TO THE PATIENT, IN THIS CASE THE HANDS AND ONE FOOT BEING IMMERSED IN JARS OF SALT SOLUTION



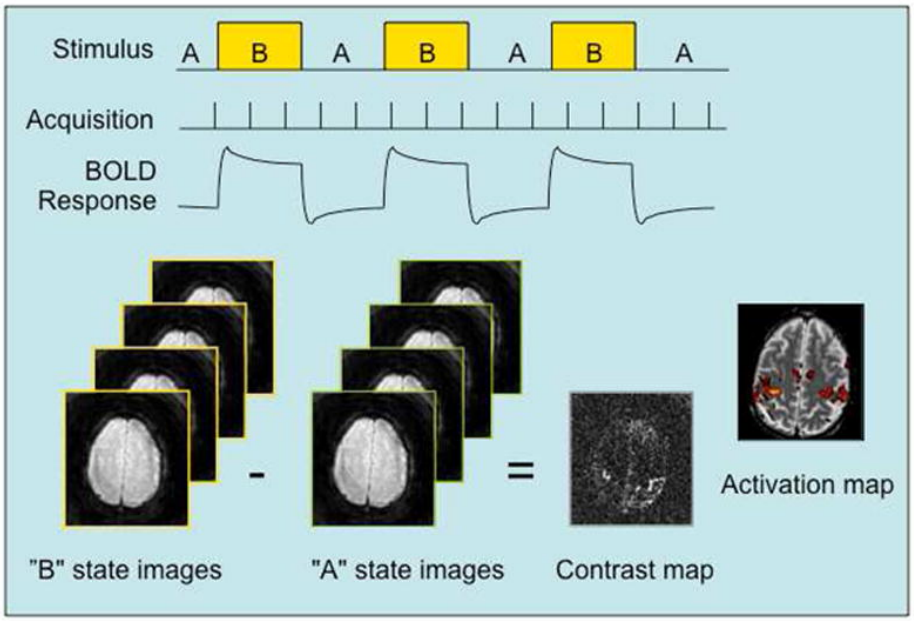
Elektroencefalograma (EEG) ir Magnetoencefalograma (MEG)

- Delta <4Hz
- Theta 4-7Hz
- Alpha 8-15Hz
- Beta 16-31 Hz
- Gamma >31

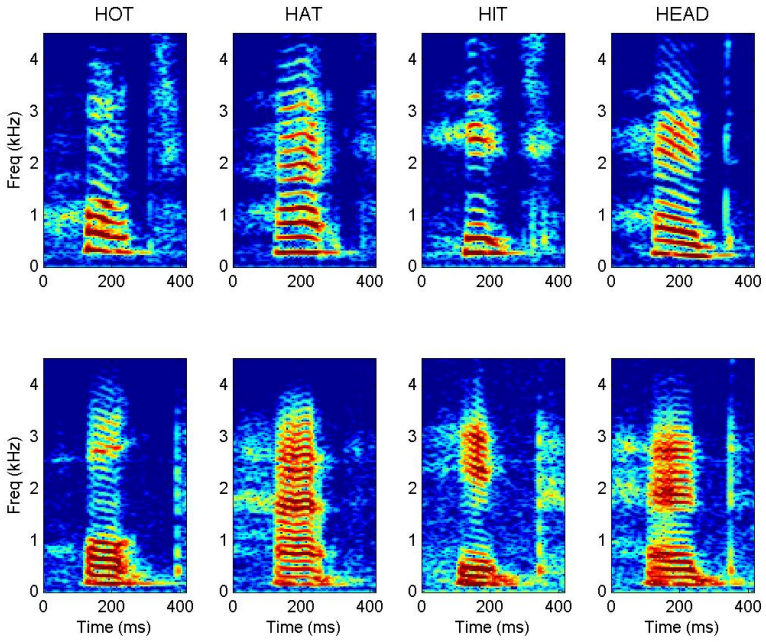



NASION
Fp1
Fp2
F7
F3
Fz
F4
F8
A1
T3
C3
Cz
C4
T4
A2
T5
P3
Pz
P4
T6
O1
O2
INION
-5μV
5μV
No
Na
Nb
Nd
N1
N2
Po
I
II
III
IV
V
VI
Pa
P1
P2
P3
0
10
100
1000 ms
ABR
AMLR
ALLR

Funkcinis magnetinis rezonansas (fMRI)



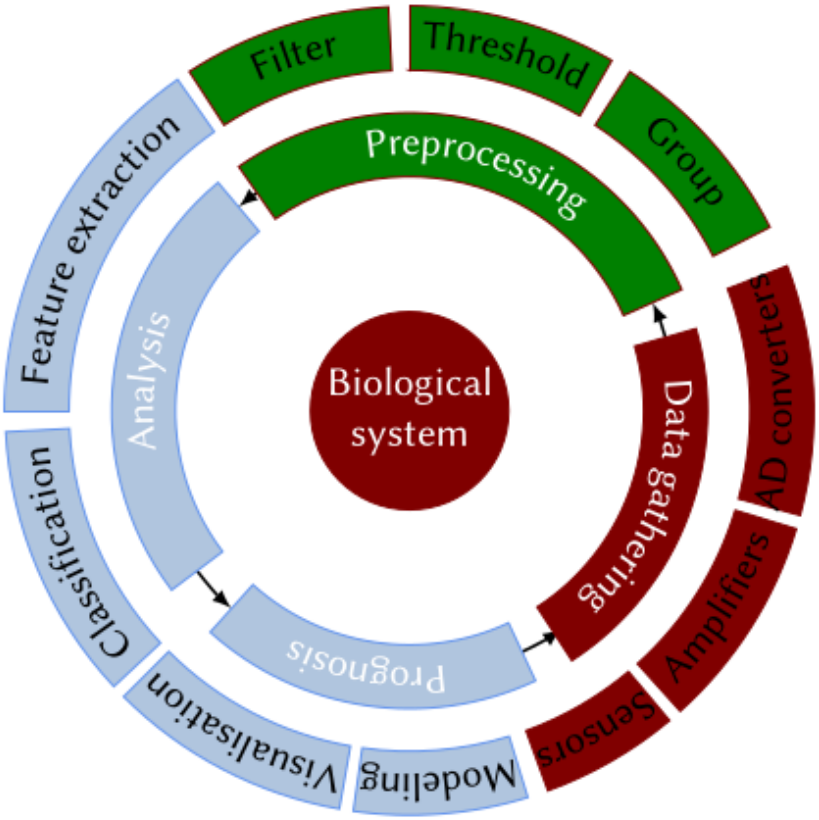
Kalba



Klausimynai (Opler, 2017)

- Pozityvių ir negatyvių simptomų skalė (PANSS) skirta nustatyti šizofrenija sergančių pacientų simptomų išreikštumą.
- Skalę sudaro trys dalys, kuriuose įvertinami įvairūs ligos simptomai (pozityvūs, negatyvūs, neuromotoriniai, depresiniai), taip pat šeimos narių, slaugytojų pranešima
- Iš 30 testo elementų, septyni priskiriami pozityviai skalei (minimalus įvertinimas 7, maksimalus -- 49), kuri aprašo normalių funkcijų pakitimus (pvz., haliucinacijos, kliedesiai, didybė, iliuzijos)
- Negatyvi skalė reprezentuoja normalių funkcijų, kaip gebėjimo atskirti realybę ar reikšti emocijas, netekimą (7 elementai, minimalus įvertinimas 7, maksimalus įvertinimas 49)
- Bendra psichopatologijos skalė vertina 16 elementų (minimalus įvertinimas 16, maksimalus įvertinimas 112) tokių kaip depresija, dezorientacija ir t.t.
- Galiausiai suminė PANSS vertė gaunama susumuojant visas teigiamos, neigiamos ir bendros skalės vertes (minimali vertė 30 ir maksimali vertė 210)

Analizės schema



Aplinkos paruošimas

- Operacinė sistema (www.distrowatch.com)
- Komandinė eilutė (https://docs.microsoft.com/en-us/windows/wsl/install-win10)
- Sistemos valdymo programos
- Analizės vykdymo ir pavaizdavimo programos

Komandinė eilutė

Pagrindiniai kompiuterių veiksmai:

- programų paleidimas
- duomenų saugojimas
- komunikavimas su kitais kompiuteriais
- komunikavimas su žmonėmis

Dažnai komunikacijai naudojamos grafinį vaizdą (*GUI*) turinčios programos valdomos pelės pagalba. Norint pagreitinti darbą ir automatizuoti pasikartojančius veiksmus programas galima valdyti klaviatūra per terminalą (*CLI*).

Pagrindinės komandos, veiksmai:

Terminalas startuoja vartotojo namų direktorijoje `/home/neurobiofiz/` tai galime pamatyti įvedę komandą `pwd` Namų direktorija Linux sistemose trumpinama `~`. Terminalas startuodamas visada perskaito konfigūracijos failą `~/.bashrc`. Konfigūracijos failas skirtas praplėsti ir pritaikyti terminalą prie savo poreikių.

```
pwd
```

```
'/home/aleks/Documents/biod2020/lect'
```

Komanda `ls` parodo direktorijos turinį:

```
ls
```

```
lect1_Intro.html  lect1_Intro.ipynb*  lect1_Intro.pdf  lect1_Intro.py*
```

Failai prasidedantys tašku `.` kaip kad terminalo konfigūracija yra paslėpti. Norint kad komanda ls juos parodytų reikia pridėti papildomų argumentų (*flags*).

```
ls -a
```

Norint pamatyt failus esančius kitoje direktorijoje prie komandos `ls` nurodome norimą direktoriją. Pavyzdžiui norėdami pamatyti turinį direktorijos ~/Documents/biod/ suvedame `bash ls ~/Documents/biod/` Pasviri brūkšniai / skiria direktorijas linux sistemoje

TAB klavišas pabaigia pradėtus rašyt žodžius. Pradėjus rašyt `~/Doc` paspaudus TAB automatiškai užbaigs žodį arba parodys galimus variantus.

Dokumentacija pasiekiama `man` komanda ir naudoja šiuos universalius simbolius:

- Navigacija hjkl arba rodyklės
- / paieška
- Ctrl+d/u žemyn aukštyn puse ekrano
- q išeiti

```
man ls
```

Reliatyvios direktorijos

Priklausomai kurioje vietoje esame į kitas direktorijas galima patekti įvairiais būdais. Jei mes esame direktorijoje pratybos/, o visas direktorijų medis atrodo taip:

```
/home/neurobiofiz/Documents/biod/
  duomenys
```

```
      csv
      img
      mat
      txt
      xlsx
  paskaitos
  pratybos
      kodas
          cpp
          jpynb
          m
          md
          py
          r
          sh
      paveikslai

17 directories
```

Jei mes norime patekti į `md` direktoriją tai galime padaryti tokiais būdais:

1. Nurodydami pilną sistemos kelią link direktorijos `cd /home/neurobiofiz/Documents/biod/pratybos/k`
2. Nurodydami pilną sistemos kelią link direktorijos pakeičiant namų direktoriją trumpiniu `cd ~/Documents/biod/pratybos/kodas/md`
3. Nurodydami reliatyvų kelią link direktorijos `cd ./kodas/md` Taškas nurodo dabartinę direktoriją ir jį galima dažnai praleisti ir rašyt `cd kodas/md`. Dvitaškis `..` nurodo direktoriją prieš dabar esamą. Taigi jei esame direktorijoje `md` `ls ../` komanda parodytų turinį direktorijos `kodas`. Tai labai naudinga programose nurodant kur yra duomenys -- pasikeitus kompiuteriui ar nukopijavus projektą į kitą laikmeną pilnas kelias pasikeičia, bet reliatyvus lieka toks pats. Taigi jei mes turime savo programą direktorijoje `py` ir iš jos norime pamatyti duomenų pavadinimus direktorijoje `txt` rašome komandą:

```
ls ../../../duomenys/txt/
```

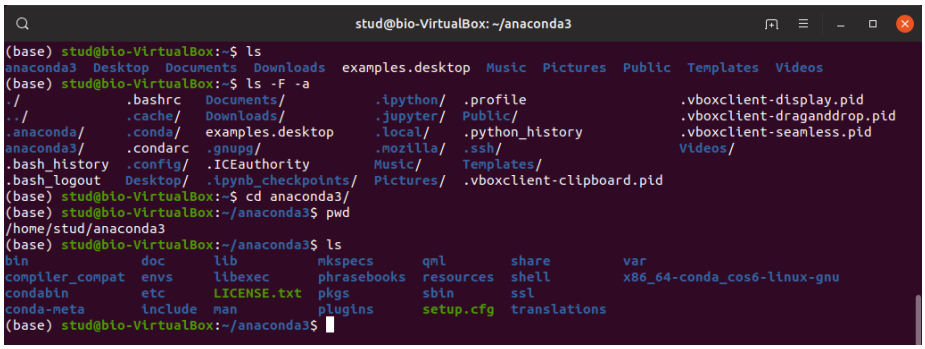
```
cd .
```

```
cd ..
```

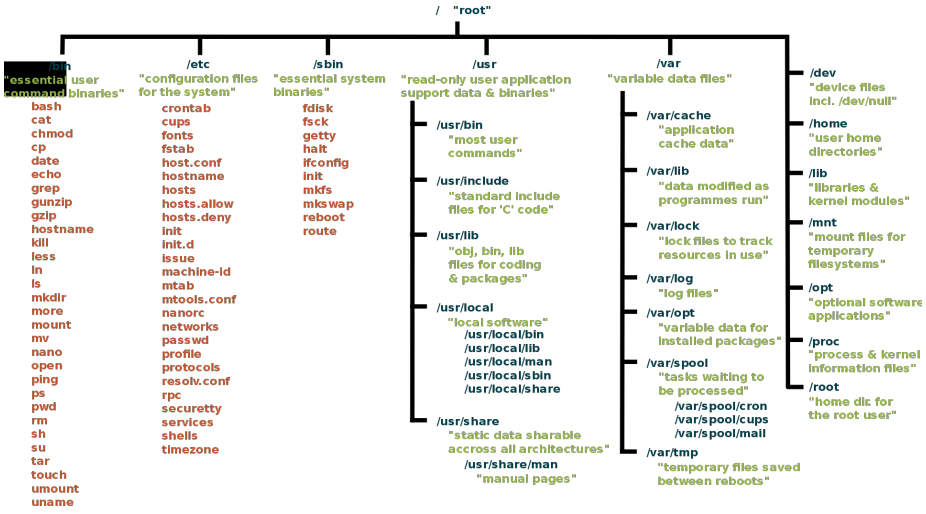
```
cd ~
```

```
cd -
```

```
cd ~/Documents/biod2020/
```



Linux Failų sistema



Kūrimas direktorijų ir failų

- mkdir direktorijosPavadinimas

```
mkdir testas
ls
```

- Nenaudot tarpų, lietuviškų raidžių. Esant tarpams reikia apsupti žodžius kabutėmis.
- Naudotini simboliai: raidės, skaičiai, -, _

Galima sukurti iškarto visą direktorijų medį:

```
mkdir -p Biodata/{paskaitos,pratybos/{paveikslai,kodas/{py,jpynb,md,sh,m,cpp,config}},duomen
```

```
Biodata/
   duomenyscsv
   duomenysimg
```

```
    duomenysmat
    duomenystxt
    duomenysxlsx
    paskaitos
    pratybos
        kodas
            cpp
            jpynb
            m
            md
            py
            config
            sh
        paveikslai

16 directories, 0 files
```

- touch failoVardas - sukuria tuščią failą. Sukurti failus galima naudojant nano, vim arba grafinę sąsają turinčias programas (gedit, kate, notepad)

```
cd test
touch test.txt
ls
```

- Failai perkeliami, pervadinami su move komanda: mv ~/Documents/tekstas.txt ~/Downloads/tekstas2.txt (kopijuoja komanda --- cp, ištrina --- rm)

```
mv test.txt test2.txt
ls
```

Filtravimas Wild cards

`*` ir `?` simboliai praverčia norint dirbti su daugiau nei vienu failu ar direktorija. Visus duomenų failus parodo komanda: `bash ls ~/Documents/biod/duomenys/csv/` Jei mes norėtumėme pamatyti tik failus kurių pavadinime yra raidžiu junginys LY naudotumėm simboli `*` kuris atstoja 0 ar daugiau simbolių `bash ls ~/Documents/biod/duomenys/csv/*LY* ?` atstoja 1 simbolį.

```
rm *txt
ls


cd ..
rm -rf test
```

- Įrašom tekstą į failą teksto redaktoriumi arba komanda echo

```
echo 'komandine eilute' > textas.txt
```

Peržiūrėti failo turinį galima šiais būdais

```
cat textas.txt
```

```
less textas.txt
```

```
head textas.txt
```

```
tail textas.txt
```

Operacijų grandinės (pipes)

- Sujungti programas galima grandinių pagalba. Tai leidžia skirtingas programas sujungti unikaliems tikslams.

1. Viengubas ženklas > po komandas `komanda > failas` sukuria failą (perrašo jei egzistavo) ir į jį įrašo išeigą komandos.
2. Dvigubas » po komandos `komanda >> failas` failo gale prideda išeigą komandos
3. Vertikalus brūkšnys | tarp dviejų komandų nurodo terminalui kad mes norim pirmos komandos išeigą naudoti sekančioje programoje pavyzdžiui eilučių skaičių visuose .txt failuose suskaičiuoti ir įrašyti galima tokia komanda

```
wc -l *.txt >> textas.txt
cat textas.txt
```

arba sugeneruoti atsitiktinę seka ATGC raidžių:

```
cat /dev/urandom | tr -dc 'ATGC' | fold -w 60 | head -n 5
```

Paieška

- paieška teksto ar žodžių failuose:

```
grep -n 'eilute'
```

- paprastesnė sintaksė ir daug greitesni: silversearch (ag), ripgrep (rg)

```
rg eilute
```

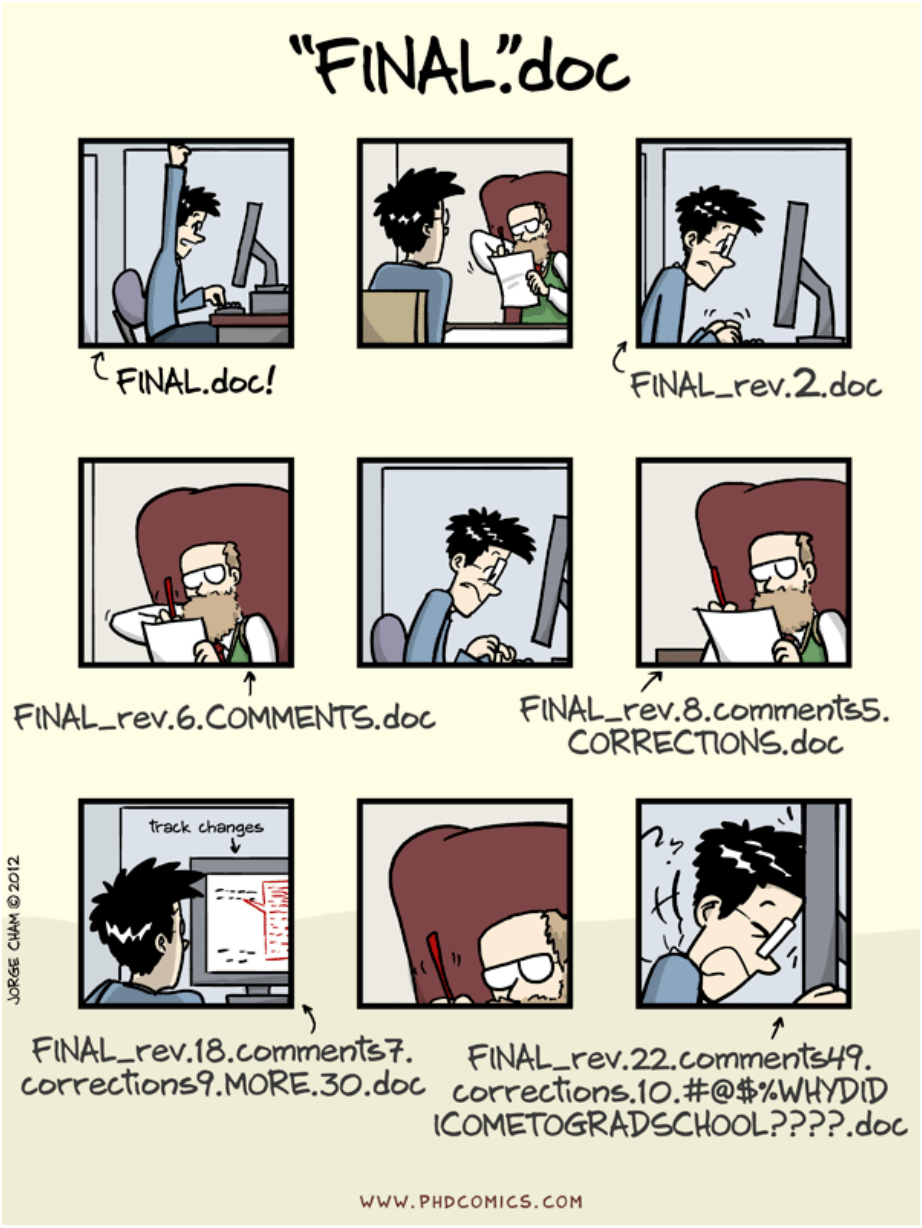
- paieška failų:

```
find . -name lect1
```

Failų prieigos teisės

- Teisės kiekvienam failui ar direktorijai galima pamatyt komanda: ls -l failoVardas
- Teisės keičiamos komanda: chmod u=rwx failoVardas
- Įgauti administratoriaus teises galima naudojant komandą: sudo

Git įvadas



- github, bitbucket, gitlab - nemokamos talpyklos
- figshare.com, zenodo.org, osf.io tuo pačiu standartu besikuriančios atkartojamo mokslo talpyklos
- talpyklos su python darbais
- tensorflow

- istorijos
- osf.io ir panašūs puslapiai talpinantys mokslinius straipsnius

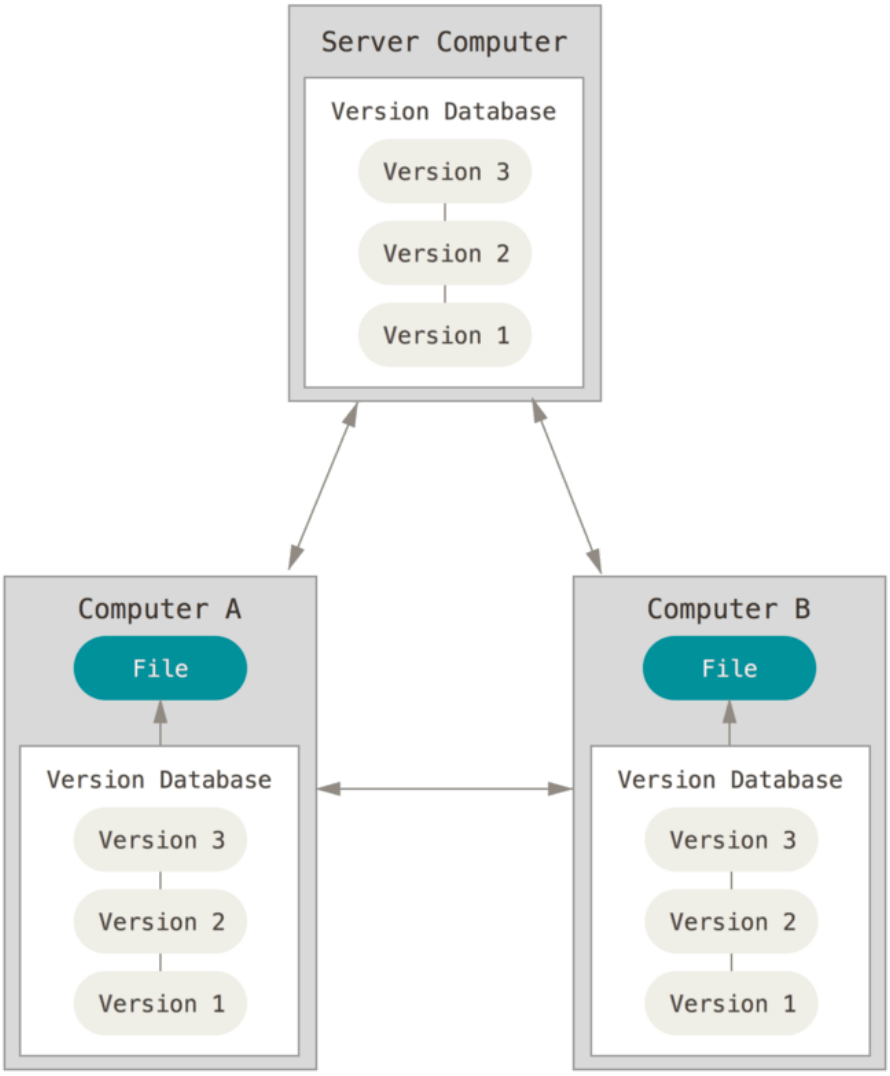
Kam reikia?

Leidžia kontroliuoti dokumentus

- Grįžti prie buvusios versijos
- Palyginti skirtingas versijas
- Vienu metu turėti daug versijų
- Matyti kas ką ir kada pakeitė
- Sincronizuoti tarp skirtingų irenginių
- Saugoti atsarginę pilną versiją
- Dirbti komandoje
- Visa projekto eiga lengvai atkuriama (reproducible)

Pilna dokumentacija: https://git-scm.com/book/en/v2

Git istorija



- Vietinė versijų kontrolė (RCS)
- Centralizuota vietinė versijų kontrolė (subversion, CVS)
- Paskirstyta versijų kontrolė (Git, Mercurial)

Diegimas

https://git-scm.com/download/ (arba is anacondos: conda install git)

Pirmą kartą naudojantis git reikia atlikti **konfigūraciją**. Nurodome vardą ir

el-paštą kuris matysis atliekant pakeitimus.

```
git config
```

```
git config --global user.name "Vardas Pavarde"
```

```
git config --global user.email pastas@domain.com
```

Numatytasis teksto editorius naudojamas git - vim Norint pasikeisti reikia įrašyt atitinkamą komandą (išeilės: nano, notepad++, gedit, isual studio code):

```
git config --global core.editor "nano -w"
```

```
git config --global core.editor "'c:/program files (x86)/Notepad++/notepad++.exe' -multiInst
```

```
git config --global core.editor "gedit --wait --new-window"
```

```
git config --global core.editor "code --wait"
```

Šiuolaikiniai IDE tokie kaip PyCharm, Jupyter lab, visual studio code turi GUI git integraciją.

Talpyklos kūrimas

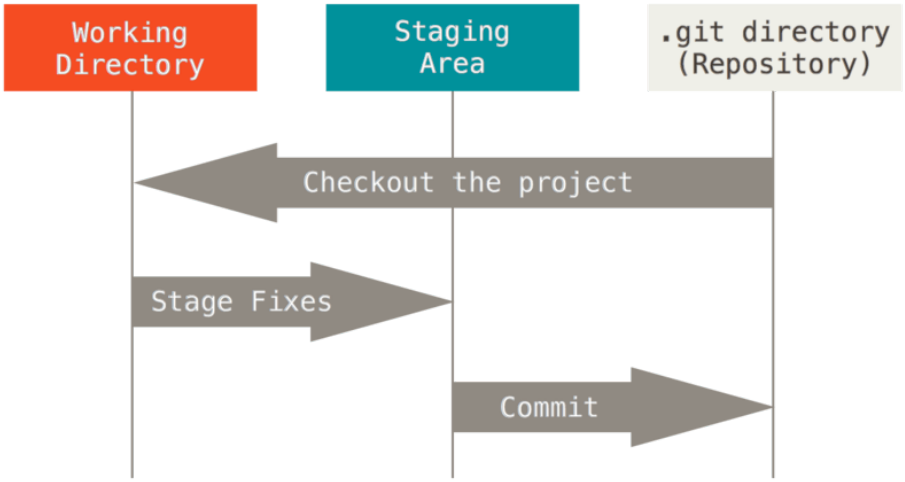
```
git init
```

Atsisiuntimas egzistuojančių (pvz kurso talpyklos)

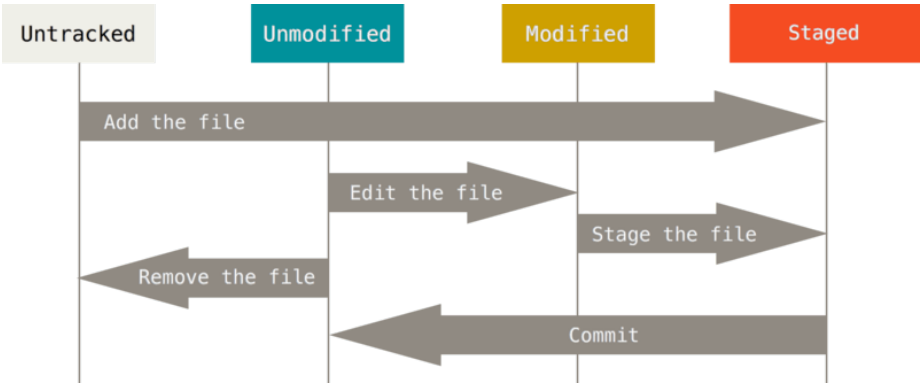
Galimi variantai protokolo https://, git:// (ssh)

```
git clone https://github.com/avoicikas/biod2020.git
```

Git darbo eiga



- Modify, stage, commit
- Valdoma komandinės eilutės, specifinės programos arba tiesiai iš teksto įvedimo programų



Darbas su saugykla

Informacija

```
git status
```

Failo pridėjimas. -A prideda visus failus saugyklos ribose.

```
git add tt.txt
```

```
git status
```

```
git add -A
```

```
git add .
```

```
git status
```

Failas pašalinamas iš sekamų

```
git rm --cached tt.txt
```

Failų ignoravimas

Dažnai yra tokių failų kurių mes nenorime kitimo sekti ar saugoti. Pvz: techniniai failai, dideli duomenų failai. Tam yra sukuriamas .gitignore failas kuriame nurodomos taisyklės kurių failų saugykla neturėtų matyti. Šio failo paiso ir daugelis kitų programų. Pvz: rg neieškos teksto šiuose failuose.

- https://www.gitignore.io/
- Glob taisyklės: ? * [abc] [0-9]
- # komentarai ignoruojami

Pavyzdžiai taisyklių:

- ignoruos failus kurių plėtiniai o arba a `*.[oa]`
- ignoruos failus besibaigiančius tilde `*~`
- nors ir yra taisyklė ignoruoti a gale turinčius failus lib.a nebus ignoruojamas `!lib.a`
- ignoruoti direktoriją build `build/`
- ignoruos doc direktorijoje visus txt failus `doc/*.txt`
- ignoruos pdf failus doc direktorijoje ir jos subdirektorijose `doc/**/*.pdf`

Pakeitimų užsaugojimas

Pridėjus failus prie sekamų sąrašo ir atlikus darbus norint užsaugoti esamą situacija atliekama commit komanda

```
git commit -m 'zinute kad atsiminti kokie darbai atlikti sioje stadijoje'
```

Padarius klaidą galima ištaisyti žinutę komanda:

```
git commit --amend
```

Pažiūrėti istoriją saugojimų galime komanda log:

```
git log
git log --stat
git log -p -2
git log --pretty=oneline
git log --pretty=format:"%h - %an, %ar : %s"
```

Pamatyti skirtumus tarp versijų

```
git diff hash1 hash2
```

Atstatyti versijas

```
git reset HEAD filename
```

arba

```
git checkout -- filename
```

Nuotolinė saugykla

Pamatyti nuotolinės saugyklos adresą

```
git remote -v
```

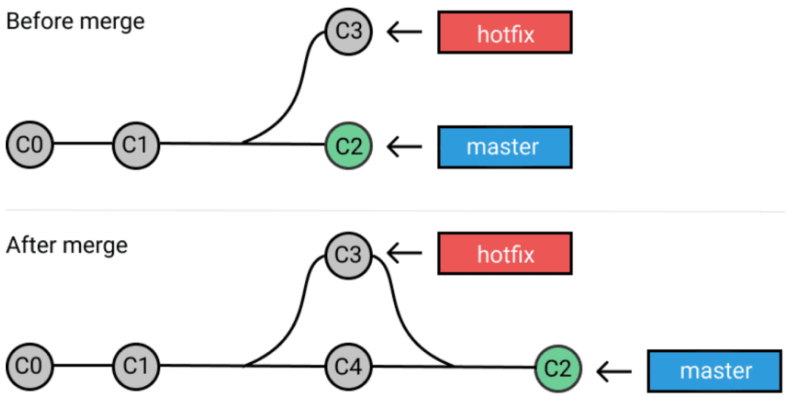
Pridėti nuotolinę saugyklą

```
git remote add shortname URL
```

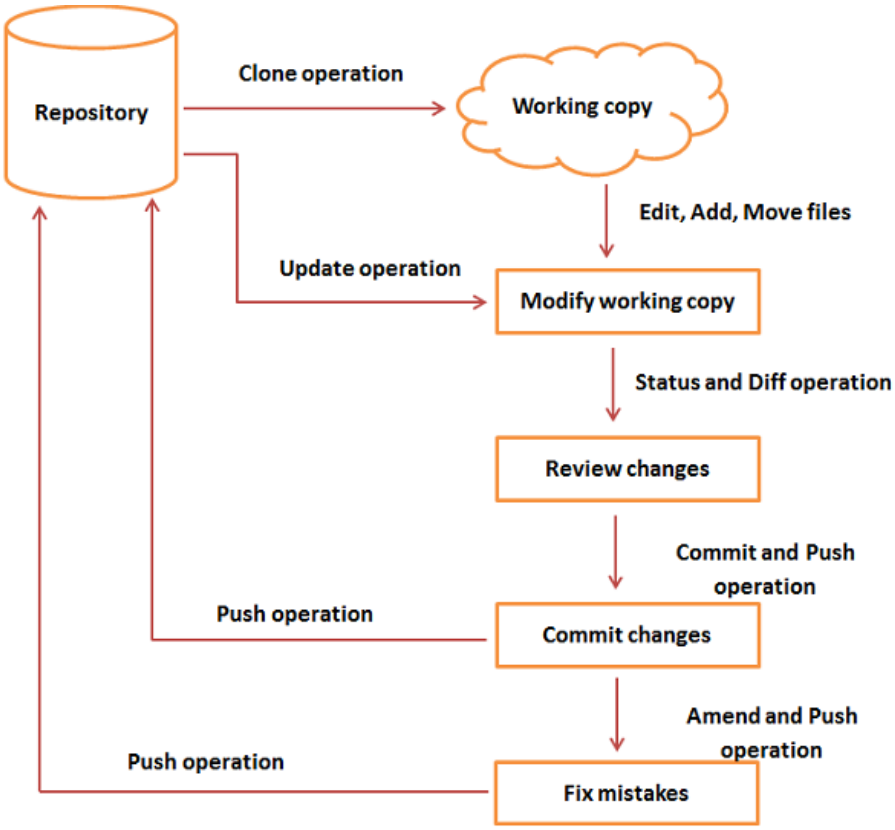
Atnaujinti duomenis

```
git pull
```

Šakojimas



- git branch branchName sukūrimas atšakos
- git checkout master
- git merge branchName sujungimas atšakų
- git branch visų atšakų pavaizdavimas



Daugiau informacijos ir nuotolinių saugyklų aprašus galima rasti:

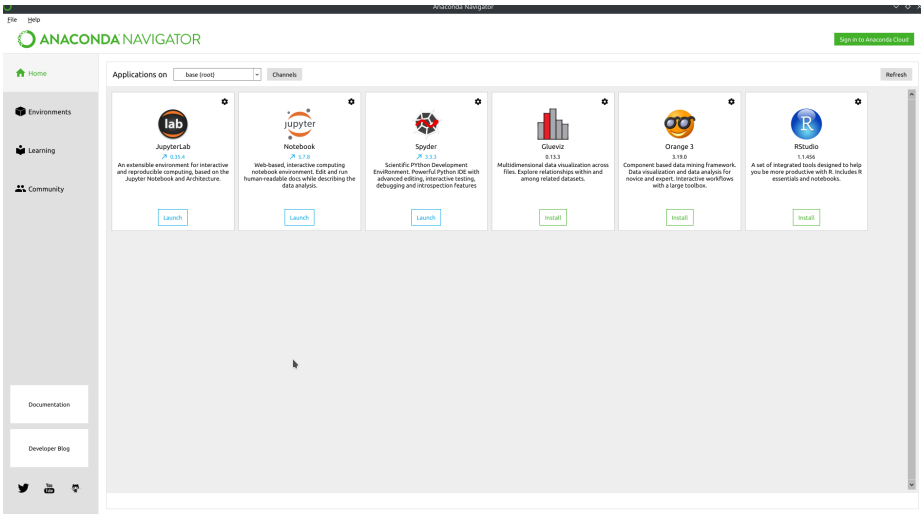
- github.com
- bitbucket.org
- gitlab.com
- https://guides.github.com/activities/hello-world/
- https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud

Anaconda Tai populiariausia python mokslui ir duomenų analizei skirta platforma tvarkanti pakuotes ir aplinkas. Skirtingai nuo pip conda pakuotės yra sukompiliuotos (pip atsisiuntęs turi sukompiliuoti pakuotes, o tam reik atitinkamai paruošti kompiuterį skirtingoms pakuotėms). Conda yra platesnio profilio tvarkyklė galinti tvarkyti ir python versijas ir kitomis kalbomis parašytas programas. Visgi pip turi daug kartų daugiau pakuočių skirtų įvairiausiems tikslams tuo tarpu conda fokusuojasi į duomenų analizę. Prireikus pakuočių nesančių conda sąraše galima jis įdiegti conda sukurtoje aplinkoje standartine pip diegimo tvarka.

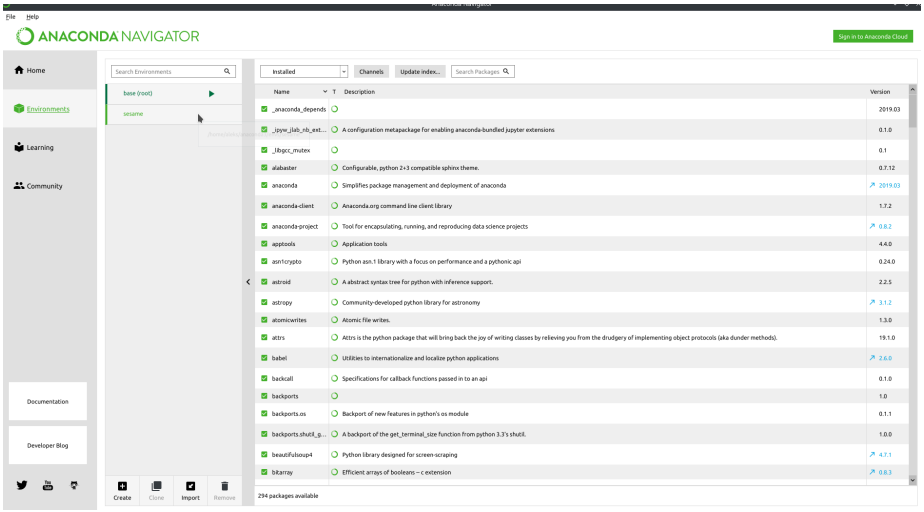
Diegimas windows aplinkoje Chocolatey programa skirta automatizuoti windows sistemoje įdiegimą ir tvarkymą programų.

- Windows:
  - choco install anaconda
  - arba atsisiuntus Anaconda
- Linux
  - naudoti distribucijos tvarkykles arba atsisiųsti

Tiek windows tiek Linux variante galima conda valdyt GUI pagalba (anaconda-navigator). Home skiltyje galima pasirinkti ir startuoti programas kodo redagavimui pasirinktoje aplinkoje (*environments*).



Environments skiltyje galima pasirinkti/sukurti/ištrinti aplinkas ir instaliuoti bei atnaujinti aplinkose esančius paketus.



Detaliau oficialiame Anaconda apraše.

Norint detaliau valdyti situaciją atsidarome terminalą. Windows sistemoje ieškome `Anaconda Prompt`.

Atidarytame terminale matome aktyvią conda aplinką `base`. Naują aplinką pavadinimu NEURO galima sukurti `conda create --name NEURO` komandos pagalba. Naują aplinką būtina kurti kai naudojamos nestandartinės (senesnės) versijos pakuotės ar kai norim izoliuoti sistemą (pavyzdžiui esant konfliktuojančioms pakuotėms), arba norim naudoti specifinę python versiją.

Visas sukurtas aplinkas galima pamatyti komanda

```
conda info --envs
```

Gautoje išeigoje * žymi aktyvią aplinką

```
conda environments:

      base          * /home/neurobiofiz/Anaconda3
      NEURO           /home/neurobiofiz/Anaconda3/envs/NEURO
```

Aktyvuoti norimą aplinka:

```
conda activate NEURO
```

Pakuotes sudiegtas aplinkoje matome komanda

```
conda list
```

Paieška pakuotės numpy

```
conda search numpy
```

Diegimas pakoutės numpy

```
conda install numpy
```

Bendra informacija

```
conda info
```

- Sukūrimas naujos izoliuotos aplinkos ir diegimas reikalingų pakuočių

```
conda create --name environment_name python=2.7 matplotlib
```

- Visų aplinkų pavaizdavimas

```
conda info --envs
```

- Aplinkos įjungimas ir išjungimas

```
conda activate|deactivate environment_name
```

- Aplinkos ištrynimas

```
conda remove --name environment_name --all
```

- Paieška pakuočių

```
conda search package_name
```

- Diegimas pakuočių dabartinėj aplinkoje

```
conda install python=3.4 numpy
```

Kadangi conda dirba ne vien su python pakuotėmis galime atsisiųsti ir daug kitų programų naudojamų analizėj

```
conda install git
```

- Įdiegtų pakuočių aplinkoje pavaizdavimas

```
conda list
```

Dažnai pateikiama informacija apie pilną conda aplanką environment.yml failuose. Turint tokį failą lengva sudiegti reikalingas pakuotes naujame izoliuotame aplanke.

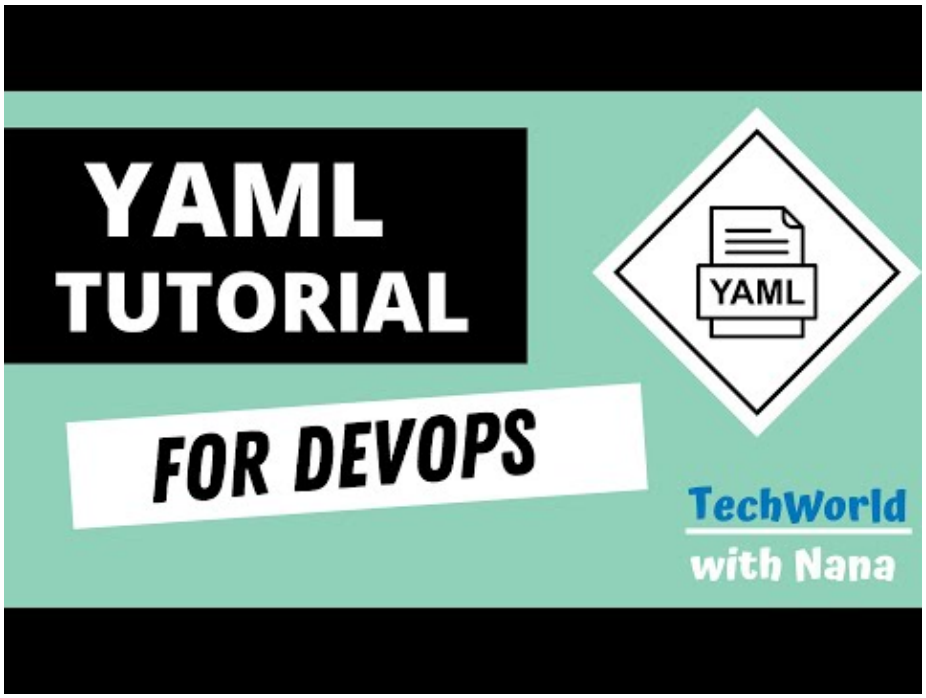
```
conda env create --name mne --file environment.yml.
```

Detaliau oficialiame Anaconda apraše

Komandų trumpas sąrašas

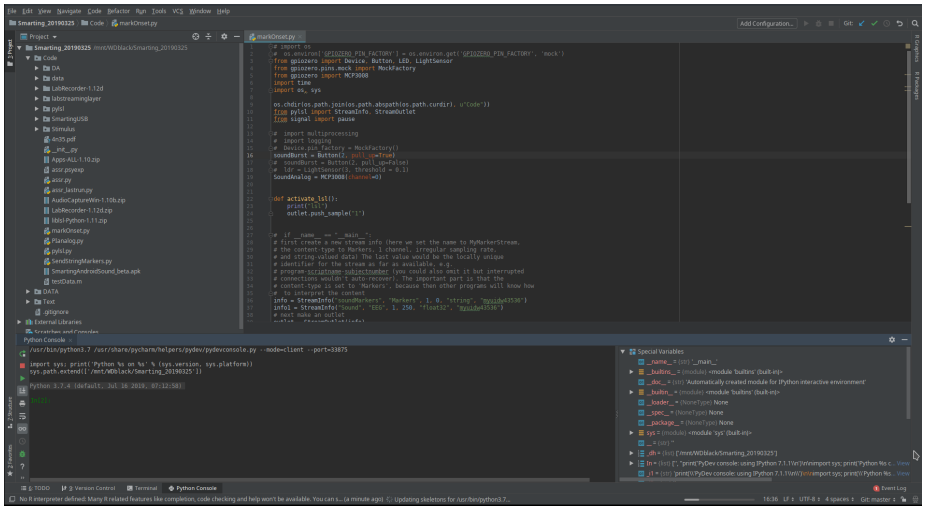
YAML intro

```
from IPython.display import YouTubeVideo
video = YouTubeVideo(id='1uFVr15xDGg', width=400, height=200, fs=1, autoplay=0)
video
```

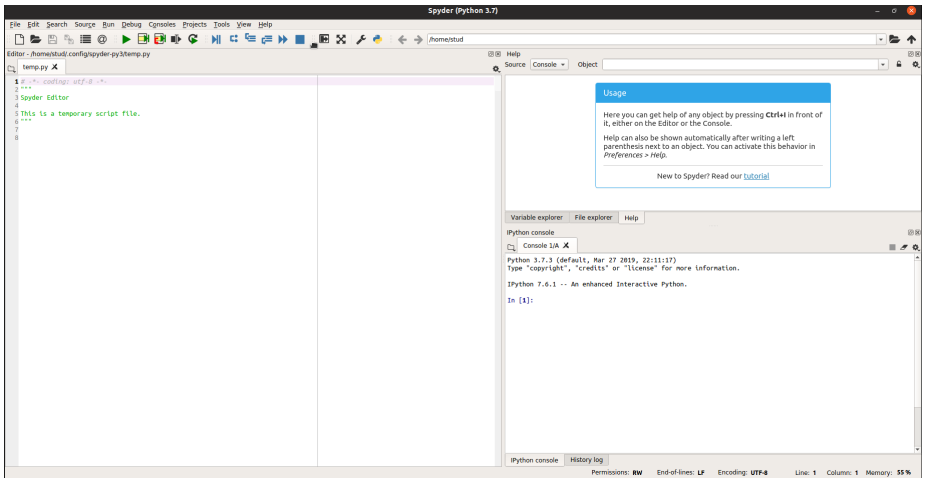


Teksto surinkimo programos

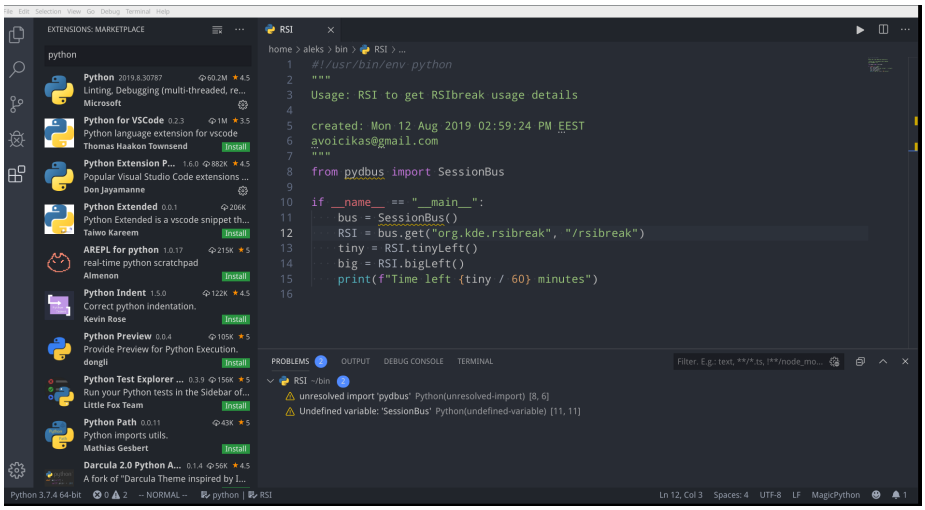
- PyCharm



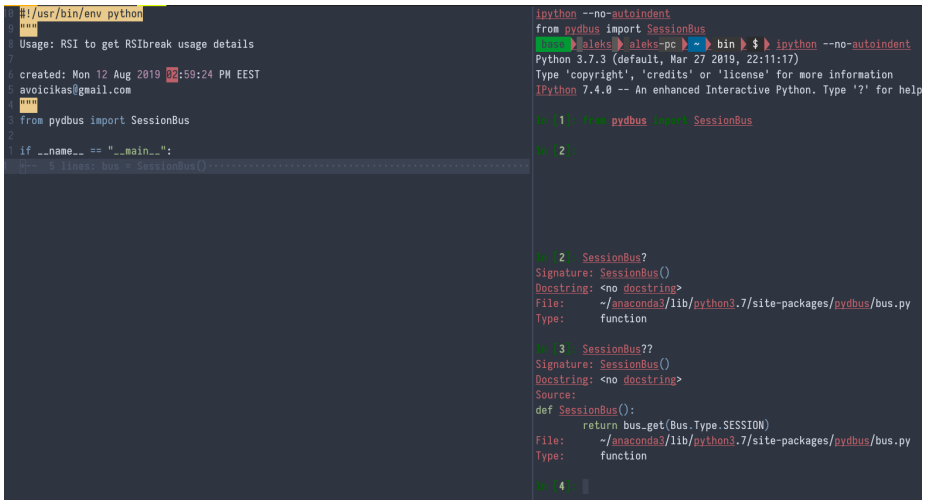
- Spyder



- Idle
- Visual Studio Code



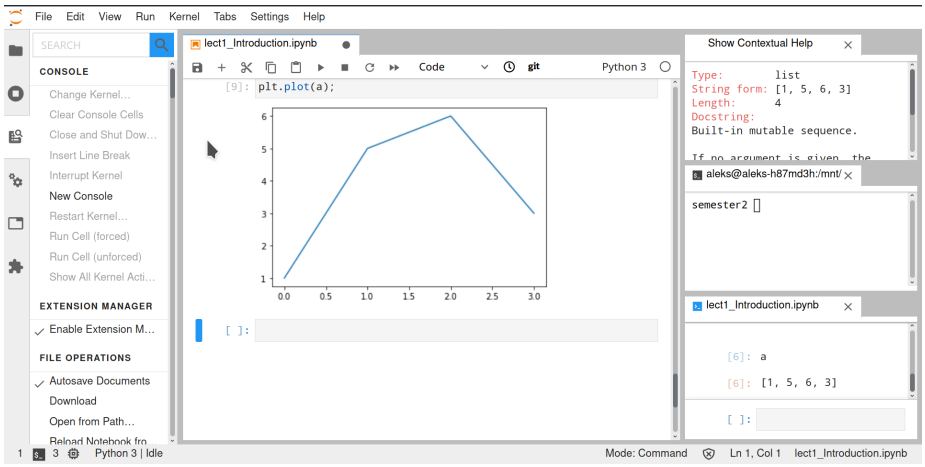
- VIM



- JupyterLab

Jupyter lab

JupyterLab padeda dirbti su Jupyter ipynb failais, teksto redaktoriais, terminalais, ir kitais komponentais integruotoje lengvai praplečiamoje aplinkoje.



Įjungimas ir išjungimas

- Terminale norimoje direktorijoje `jupyter lab` komanda startuoja. Paspaudus `Ctrl-c` ir patvirtinus `y` jupyterlab sustabdomas.
- Terminale parašius anaconda-navigator atsidaro GUI su įvairiomis opcijomis kur galima startuoti jupyterlab paspaudus ant ikonos.

Bazinės komandos

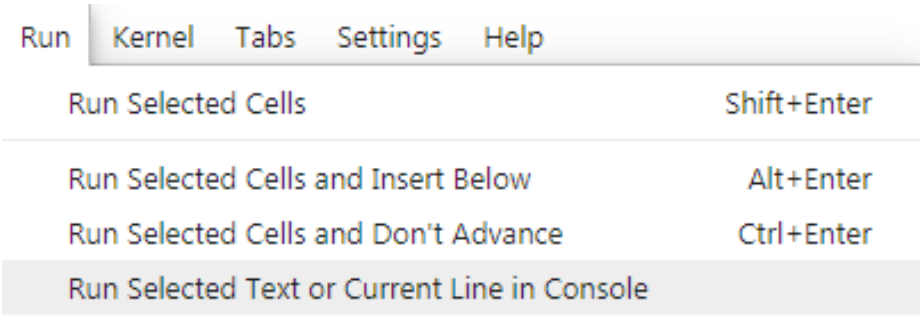
Help -> JupyterLab reference

Pradėjus darbą atkarpoje **cell** pasirinkti kokio tipo bus atkarpa: markdown ar code

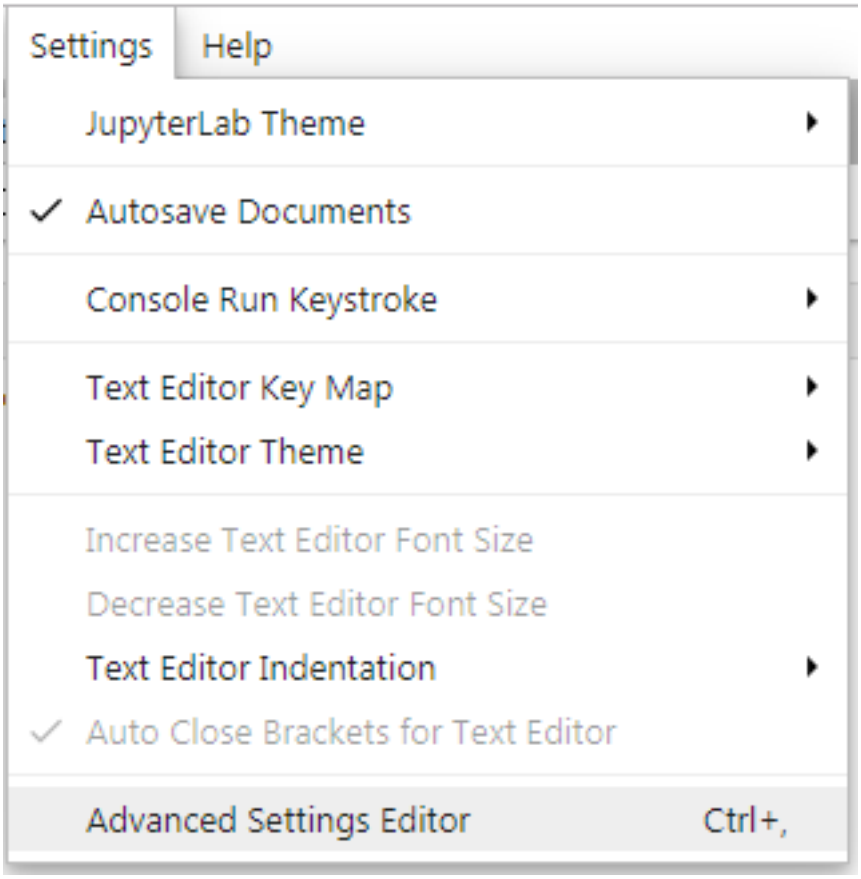
shift+enter - įvykdo komandas **cell** atkarpoje

Siuntimas kodo į konsolę

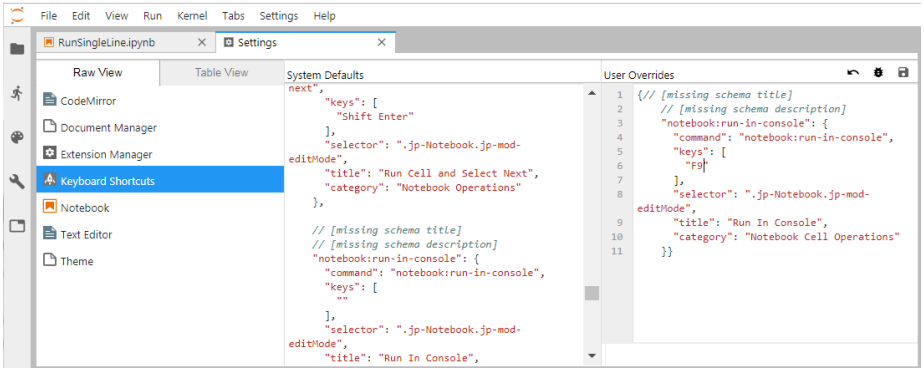
Run -> Run selected text or line in console



Norint padaryti klaviatūros spartųjį klavišą: Settings -> Advanced Settings Editor



ctrl+F paieškoti cun-in-console eilutės ir nukopijuoti ją į šalia esantį langą pavadinimu User Overrides (Pasirinkti norima klaviša pvz F9) ir paspausti dešinėje viršuje išsaugojimo mygtuką.



Plėtinių aktyvavimas

Advanced settings editorius, extension manager sekcija -> true. Išsaugoti ir paspausti enable. Kairiame šone atsiranda plėtinių paieškos opcija.

Naudingų plėtinių pavyzdžiai

- https://github.com/ryantam626/jupyterlab_code_formatter
- https://github.com/jupyterlab/jupyterlab-toc
- https://github.com/QuantStack/jupyterlab-drawio

Markdown

Jupyter lab sujungia python ir kitas programavimo kalbas su teksto maketavimo kalbom markdown ir latex. Tai leidžia viename dokumente turėti analizės eigą kartu su paveikslėliais, rezultatais ir aprašu.

dokumentacija

Jupyter užrašinėja (notebook, ipynb) galima suprogramuoti interaktyvius grafikus:

```
import ipywidgets as widgets

p = 0.5
n = 10
flips = np.random.choice(["H", "T"], p=[1-p, p], size=n)
@widgets.interact(p=(0.0, 1.0), n_flip=(1, 1000), continuous_update=False)
def simulate_coin_flip(p, n_flip):
    flip = np.random.choice([0, 1], p=[1-p, p], size=n_flip)
    c_sums = np.cumsum(flip)
    c_means = c_sums / np.arange(1, n_flip + 1)
    plt.plot(np.arange(1, n_flip+1), c_means)
    plt.plot(range(n_flip), [p] * n_flip, 'k', linewidth=1)
    plt.xlabel('Number of flips')
    plt.ylabel('Fraction of heads')
    plt.xlim([1, n_flip])
    plt.ylim([0, 1])
```

{"model_id":"4f7ca23f952a48b7b9b7385ac9beb27c","version_major":2,"version_minor":0}

Įkelti įvairaus formato mediją, rašyti html kodą

```
video = YouTubeVideo(id='1WBgTrCBKEM', width=400, height=200, fs=1, autoplay=0)
video
```

Latex sintakse rašyti formules:

Command	Description	Output
\frac	Build a fraction like so: | $\fra	$\frac{1}{7}$
\frac{\frac{}}{}	You can nest fractions | $\	$\frac{\frac{1}{2}}{2}$
\alpha	alpha | $\al	α
\beta	beta | $\bet	β
\gamma	gamma | $\ga	γ
\delta	delta | $\de	δ
\epsilon	epsilon | $\	ϵ
\zeta	zeta | $\zet	ζ
\eta	eta	η
\theta	theta | $\th	θ
\iota	iota | $\iot	ι
\kappa	kappa | $\ka	κ
\lambda	lambda | $\l	λ
\mu	mu |	μ
\nu	nu |	ν
\xi	xi |	ξ
\rho	rho	ρ
\sigma	sigma | $\si	σ
\tau	tau	τ
\upsilon	upsilon | $\	υ

Command	Description	Output
\phi	phi	ϕ
\chi	chi	χ
\psi	psi	ψ
\omega	omega | $\om	ω
\forall	For | $\f	$\forall$
\exists	Exists | $\e	$\exists$
\lor	Or	$\lor$
\land	And | $\lan	$\land$
\veebar	Xor | $\v	$\veebar$
\neg	Not	$\neg$
\cdot	Dot | $\cdo	$\cdot$
\div	Division	$\div$
\pm	Plus |	$\pm$
\neq	Not	$\neq$
\approx	Approximately | $\a	$\approx$
\leq	Less	$\leq$
\geq	Greater	$\geq$
\ll	Much |	$\ll$
\gg	Much |	$\gg$
\supset	supset | $\s	$\supset$
\supseteq	Superset | $	$\supseteq$
\subset	Proper | $\s	$\subset$
\subseteq	Subset | $	$\subseteq$
\in	Member |	$\in$
\emptyset	Empty set| $	$\emptyset$
^	superscript | $	x^2
^{}	exponents with >1 digit |	x^{23}
_	subscript | $	x_i
_{}	subscript with >1 digit |	x_{ir}
\sum	Sum over digits	$\sum_i$

Dokumentų eksportavimas

- Iš grafinės sąsajos:

File -> export

- nbconvert komanda galime atlikti įvairias operacijas su užrašinėmis.

```
jupyter nbconvert --to notebook --execute labreport.ipynb
```

```
jupyter nbconvert --ExecutePreprocessor.timeout=1200 --ExecutePreprocessor.kernel_name='pyt
```

Paversti užrašinę į html neįtraukiant kodo

```
jupyter nbconvert --to=html --TemplateExporter.exclude_input=True labreport.nbconvert.ipynb
```

```
jupyter nbconvert labreport.ipynb --TagRemovePreprocessor.enabled=True --TagRemovePreprocess
```

- Pandoc konvertuoja ipynb į įvairiausius formatus.

Norint eksportuoti į pdf formatą reikės įdiegti texlive (linux) arba mikitex (windows) programas.

```
pandoc -f ipynb lect1_Intro.ipynb -o lect.pdf --toc --pdf-engine=xelatex
```

- Slidify konvertuoja užrašinę į skaidres
- jupytext konvertuoja užrašinę į ir iš markdown, R Markdown, MyST ir įvairiausių programavimo kalbų skriptus.

jupytext diegimas:

```
conda install -c conda-forge jupytext
```

Gali reikėti dar įdiegti plėtinį kad atsirastu tiesioginis konvertavimas grafinėje jupyter lab aplinkoje.

- Jupyter Lab komandos (Commands: jupytext)
- Terminale

```
jupytext --to py:light labreport.ipynb
```

```
jupytext --to notebook labreport.py
```

```
jupytext --set-formats ipynb,py labreport.ipynb
```

```
jupytext --sync labreport.py
```

Skirtingi formatai užrašinės konvertavimo į programavimo skriptus:

- percent
- Hydrogen
- Markdown
- R Markdown
- MyST
- light

lect1_Intro.py yra susieta jupytext su lect1_Intro.ipynb light formatu. Markdown elementai tampa komentarais.

Paieška informacijos internete

- komandinės eilutės santrumpos google ar panašioms svetainėms su daugeliu nustatymų

```
define biophysics
```

- Nesekančios istorijos
- Atmetančios didžiausias svetaines
- Apsimokančios iš kitų paieškos sistemų

Santrauka

Trumpa seka ko tikrai reikės:

- Atsidaryt conda ir įdiegti pakuotes (conda install X)
- Atsisiųsti naujausia medžiagą. (git pull)
- Rašyti tekstą jupyterlab aplinkoje markdown ir python sintakses pagalba
- Eksportuoti parašytą darbą html arba pdf formatu.

Demonstracija sekos windows aplinkoje