PRAKTIKUM 3: VEEBIKATSED

Selles praktikumis loome nägude tajumise katse, milles vaatame lähemalt piltstiimulite esitamist. Täpsemalt loome variatsiooni Glasgow nägude kõrvutamise katsest (Glasgow Face Matching Task, Burton et al., 2010). Tutvume sellega, kuidas seadistada piltide laadimist ja esituseks ettevalmistamist ajakriitilises katses. Tutvume PsychoPy-s stiimulite suuruse väljendamiseks kasutatavate kõrgusühikutega, kasutame värvuse väljendamiseks RGB kolmikut ja paneme koodikomponenti kasutades ühe elemendi kestuse juhuslikult varieeruma. Praktikumi teises pooles laeme katse ka Pavlovia keskkonda. Vaatleme, milliste komponentide kasutamist Pavlovia toetab, kuidas koodikomponent Pavloviale loetavasse keelde tõlkida ja õpime seda infot PsychoPy veebikeskkonnast leidma.

GFMT (*Glasgow Face Matching Task*) katses esitame katseisikule kaks nägu ja küsime, kas piltidel on sama või erinev inimene. See on üllatavalt keeruline ülesanne ja seetõttu ei pane me ülesande täitmisele ka ajapiirangut ehk laseme katseisikutel ülesannet omas tempos täita. Seega erinevalt Stroopi katsest on selle katse sõltuvaks muutujaks katseisikute vastamistäpsus. Ülesande lahendamiseks vajalikud vahendid leiad praktikumi materjalide hulgast[1]. Nende hulgas 40 pilti, millest pooltel on kujutatud samu inimesi (ehk pildid, mis on kaustas *same*) ja pooltel erinevaid (ehk pildid, mis on kaustas *different*). Kui ühe neist piltidest lahti klõpsame, siis näeme, et piltidel on valge taust, millel on esitatud kaks nägu: üks vasakul ja teine paremal. Lisaks on materjalide hulgas ka tingimuste fail, et tingimuste faili loomisele selles praktikumis liiga palju aega mitte pühendada.

Ülesanne 0. Loo kaust, kuhu salvestad uue katse tooriku, nägude piltidega kaustad (*same, different*) ja tingimuste fail (conditions.xlsx).

Avame kõigepealt PsychoPy *builderi* vaates ja salvestame uue eksperimendi ning anname sellele näiteks nimeks GFMT. Järgmiseks peaksime looma tingimuste faili, milles on kolm veergu: teekond (*path*) piltstiimuli juurde (imageFile), õige vastuse klahv (corrAns) ja info selle kohta, kas piltidel on sama või erinev inimene (sameOrDiff).

Aja kokkuhoiu mõttes kasutame juba eelnevalt ettevalmistatud tingimuste faili. Hoolikamal vaatlusel märkad kindlasti, et piltide nimetustega koos pole esimesse veergu kantud mitte kogu faili tee, vaid välja on toodud ainult kaust, milles konkreetne pilt paikneb. Kui piltide kaustad ja loodav juhtfail (GFMT.psyexp) on salvestatud samasse kausta, siis lubab PsychoPy anda stiimulite teed juhtfaili suhtes ehk teiste sõnadega ütleme juhtfailile, millisest juhtfailiga samas kaustas paiknevast kaustast tuleks stiimulit otsida. Teoorias võib piltide kaustad ka juhtfaili kaustast välja paigutada, kuid sellisel juhul peaksime tingimuste tabelis välja tooma kogu faili teekonna (nt C:\Users\ Documents\pildi_nimi.jpg). Kuna nii pika faili tee väljakirjutamine oleks tüütu ja peaksime seda muutma iga kord, kui sooviksime programmi mõnes teises arvutis käivitada, siis enamasti eelistatakse kasutada esimest

[1] PsychoPy õpiku materjalid (sh selles praktikumis kasutatavad stiimulfailid) on korraga allalaetavad siin: https://study.sagepub.com/psychology

lähenemist. Õige vastuse veerg on tingimuste failis üles ehitatud selliselt, et samale vastab vasak ja erinevale parem nooleklahv.

	A	B	C
1	imageFile	corrAns	sameOrDiff
2	same/020_C2_DV.jpg	left	same
3	same/025_C2_DV.jpg	left	same
4	same/068_C2_DV.jpg	left	same
5	same/069_C2_DV.jpg	left	same
6	same/074_C2_DV.jpg	left	same
7	same/075_DV_C2.jpg	left	same
8	same/081_C2_DV.jpg	left	same
9	same/085_C2_DV.jpg	left	same
10	same/092_C2_DV.jpg	left	same
11	same/102_C2_DV.jpg	left	same
12	same/108_DV_C2.jpg	left	same
13	same/115_C2_DV.jpg	left	same
14	same/120_DV_C2.jpg	left	same
15	same/129_C2_DV.jpg	left	same
16	same/136_C2_DV.jpg	left	same
17	same/139_DV_C2.jpg	left	same
18	same/168_C2_DV.jpg	left	same
19	same/170_DV_C2.jpg	left	same
20	same/235_C2_DV.jpg	left	same
21	same/303_DV_C2.jpg	left	same
22	different/013_235_R.jpg	right	different
23	different/019_074_L.jpg	right	different
24	different/025_235_L.jpg	right	different

Katse tingimuste faili esimesed 24 rida

Ülesanne 1. Lisa *trial* rutiinielemendile komponentide menüüst piltstiimul (*Image*) ja muuda selle järgnevaid parameetreid:

a) sea algushetkeks rutiinielemendi algus ehk 0 ja jäta kestuse lahter tühjaks (sellega kindlustame, et komponent jääb ekraanile kuni rutiinielemendi lõpuni)
b) kirjuta lahtrisse *Image* nimetus, mis vastaks tingimuste failis veerule, kuhu on salvestatud piltstiimulite faili teed (nt same/pildi_nimi.jpg) ja muuda lahtri väärtuse värskendamise parameetrit selliselt, et seda igal seerial uuendataks (NB! Ära unusta veeru nimetuse ette dollari märki lisada, nt $imageFile)

Ülesanne 2. Lisa *trial* rutiinielemendile vastuse komponent ja muuda selle järgmisi parameetreid:

a) määra lubatud klahvivajutusteks vasak ja parem

b) muuda vastuse komponenti nii, et see salvestaks ka katseisiku õige vastuse (kirjuta *Correct answer:* lahtrisse dollari märgi järele tingimuste tabelis õige vastuse veerg, nt corrAns)

Ülesanne 3. Lisa *trial* rutiinielemendi ümber tsükkel (*insert loop*) katseskeemi ehk *flow* menüüs ja muuda selle järgmisi parameetreid:

a) määra korduste arvuks 1
b) kirjuta *Conditions* lahtrisse tingimuste faili nimetus (nt conditions.xlsx)

Ülesanne 4. Ilmselt nägid, et näod on esitatud valgel taustal, kuid katse tausta värvus on vaikimisi hall. Selles ülesandes muudamegi katse tausta värvuse valgeks.

Selleks mine katse seadetesse (mutriga ikoon) ja klõpsa ekraani sätete sakil. Sealt leiad parameetri, mis kannab nimetust *Color*. Vaikimisi on sinna kandiliste sulgude sisse ja dollarimärgi järele kirjutatud kolm nulli ($[0,0,0]). Erinevalt varasematele ülesannetele, kus kirjutasime värvikasti värvinimetuse (nt *white*) harjutame siin RGB kolmikute kaudu tähistamist. RGB kolmiku defineerimisel saadakse elemendi värvus kolme värvi (punane, roheline, sinine) kombinatsioonina, mis varieeruvad vahemikus -1 ja 1. Null tähistab olukorda, kus punast, rohelist ja sinist on täpselt sama palju ja annab kokku halli. Kui soovime tausta valgeks muuta, siis peaksime kõik kolm värvust põhja keerama ehk asendame nullid ühtedega ($[1,1,1]). Sarnaselt pilditöötlusprogrammidele saab siingi avada menüü, mis lubab ka silma järgi värvi valida. Selleks tee parem hiireklõps *Color* kastikeses (ainult vanem PsychoPy versioon) või tee vasak hiireklõps kastikese järel oleval ikoonil.

Õnnitlen, selliselt on katse juba täiesti jooksutatav! Päris katse puhul tahaksime siia lisada kindlasti ka instruktsioonid ja treeningseeriad. Võid lisada ekraanile ka sildid, et katseisikul klahvide tähendused meeles püsiksid. Selleks tuleks *trial* rutiinielemendile liita veel kaks tekstikomponenti, millest üks paikneks vasakul üleval (sama) ja teine paremal üleval (erinev). Kui oleme veendunud, et katse töötab, siis lisame eksperimendile veel ühe rutiinielemendi: isi (*inter stimulus interval*), mille eesmärk on ühelt poolt teha väike paus esituste vahele ja teisalt annab see meile võimaluse uue pildi mällu laadimiseks, mis teeb programmi töö ajaliselt täpsemaks. Kuigi antud katses pole ajaline täpsus väga kriitiline, siis pakub ta hea võimaluse harjutada, kuidas mõnd ajakriitilisemat piltstiimuleid kasutavat katset üles ehitada.

Ülesanne 5. Lisa katseskeemi menüüsse uus rutiinielement ja anna sellele nimeks isi. Lisa selle rutiinielemendi koosseisu stiimulkomponentide hulgast visuaalne kujund (*polygon*) ja muuda selle järgmisi parameetreid:

a) anna selle nimeks fix (fiksatsioonirist)
b) määra selle kujuks rist (*cross*)
c) määra selle kestuseks 1 sekund
d) määra selle värvuseks must
e) määra selle küljepikkusteks 0.05 pikkusühikut (*height*)

Veebikatsete puhul kasutatakse stiimulite suuruste ja positsiooni väljendamisel tihti kõrgusühikuid (*height*). Vaatame height ühikuid võrdluses teise normaliseeritud ühikuga *norm*, mille puhul vasakult kõige alumine punkt vastaks koordinaatidele [-1, -1] ja paremalt kõige ülemisele [1, 1]. Teiste sõnadega akna kõige parempoolsele pikslile 1 ja kõige vasakpoolsemale -1 ning kõige alumisele -1 ja kõige ülemisele 1. Sõltumata sellest, kas akna resolutsiooniks on 1920x1080, 1024x768 või 800x600. Kuigi sellised ühikud teevad visuaalsete koordinaatides orienteerumise hõlpsaks (nt ekraani keskpunkti koordinaadiks on punkt [0,0]), siis ei võta need ühikud arvesse, et monitoride laius on enamasti suurem kui kõrgus. Probleeme valmistab see põhiliselt siis, kui tahaksime ekraanil kujutada midagi, mille kõik küljed on ühe pikkused. Näiteks kui kuvaksime ekraanile fiksatsiooniristi ja määraksime tema laiuseks ja kõrguseks 0.05 normaliseeritud ühikut, siis oleks tulemuseks väljavenitatud rist, sest horisontaaltelge mööda vastab 0.05 ühikule enam piksleid kui vertikaaltelge mööda. *Height* ühikud on normaliseeritud kõrguse suhtes selliselt, et sellegi puhul on ekraani keskkohaks punkt [0,0], kuid nende ühikute puhul ei tea me täpselt, milline koordinaat vastab kõige vasakpoolsele või parempoolsele pikslile. Seda seetõttu, et ühikud on väljendatud akna kõrguse suhtes (täisekraanile kuvades on see sama, mis monitori suurus) selliselt, et kõige kõrgemal paiknev piksel vastab arvule 0.5 ja kõige madalam arvule -0.5. Kuna monitorid on enamasti pikkupidi välja venitatud, siis monitori külgedele jääb rohkem kui 0.5 ühikut, kuid see kui palju täpselt sõltub konkreetse monitori küljepikkuste suhtest. Näiteks standardse 4:3 suhte puhul oleks vasakult kõige alumiseks koordinaadiks [-0.6667, -0.5]. Selliste ühikute eelis näiteks pikslite ees on see, et kuvatud elementide suurused on suhtes ekraani suurusega, mis väldib olukorda, kus stiimulid moodustavad ühel ekraanil väga suure ja teisel ekraanil väga väikese osa kogu ekraanist.

Ülesanne 6. Lisa eelmises ülesandes loodud rutiinielemendi (isi) koosseisu tellitav (*custom*) komponent *static* ja muuda selle järgmisi parameetreid:

a) määra selle algushetkeks 0 ja kestuseks 1 sekund

Static komponendi ajal saab programm lahendada ülesandeid, mille täitmine (näiteks piltstiimulite mällu lugemine) esituse ajal võiks stiimulite esitustäpsust kahandada. Selles katses katsetame olukorda, kus programm valmistab esituste vahele jääval ajal ette (loeb mällu) järgmises esituses kuvatava piltstiimuli.

Ülesanne 6.1. Muuda piltstiimuli komponenti selliselt, et piltide värskendamine toimuks mitte igal kordusel vaid meie loodud seeriate vahelisel ajal (ISI).

Pane tähele, et selline valik ilmus menüüsse alles pärast komponendi *Static* lisamist.

Mõnikord soovime, et seeriate vaheline aeg poleks konstantne. See võib olla oluline selleks, et katseisik ei saaks väga täpselt ennustada, millal järgmine seeria esitatakse. Järgmises ülesandes lisame koodikomponendi, mis paneb meie seeriate vahelise aja ühe sekundi piires varieeruma.

Ülesanne 7. Lisa *isi* rutiinielemendile koodikomponent. Defineeri rutiinielemendi algusele vastavas aknas uus muutuja, mis varieeruks juhuslikult nullist üheni ja liida see **Ülesandes 5.** loodud fiksatsiooniristi kestusele.

Selleks lisa isi rutiinielemendile kõigepealt koodikomponent (*Code*) ja kirjuta rutiinielemendi algussaki (*Begin routine*) sees olevasse vasakpoolsesse (Pythoni koodile vastavasse) kastikesse rndisi = random(). Funktsioon random() annab meie poolt nimetaud rndisi

muutujale igal seerial uue ja juhusliku väärtuse nullist üheni. Kuna soovime seda väärtust fiksatsioonielemendi kestuse defineerimisel kasutada, siis peaksime tõstma koodikomponendi fiksatsiooniristi ette. Seda saab teha kui koodikomponendil parem hiireklõps teha ja avanenud menüüst *move to top* valida. Selleks, et muutujasse salvestatud väärtus ka seeriate vahelises ajas kajastuks, tuleks see liita ka fiksatsioonielemendi kestusele. Selleks ava fiksatsioonielemendi seaded ja kirjuta elemendi kestuse järele + rndisi, mis liidab igal seerial meie koodielemendis välja arvutatud aja fiksatsioonielemendi kestusele.

KATSE VEEBIKESKKONDA LAADIMINE

Alates PsychoPy kolmandast versioonist saab PsychoPy Builder'is loodud katseid ka veebis jooksutada. Siiski võib mõningaid komponente (sh koodikomponente) kasutavate katsete üleslaadimine olla raskendatud. Pidevalt uueneva nimekirja veebikatsetes kasutatavatest komponentidest leiad PsychoPy kodulehelt.

Põhjus, miks kõiki komponente pole veel võimalik veebikatses kasutada peitub selles, et programmi üleslaadimisel tõlgib PsychoPy Pythoni programmi ümber JavaScript*i programmeerimiskeelde, kuid arendajad pole veel kõiki PsychoPy funktsionaalsusesse kuuluvaid elemente JavaScripti tõlkida jõudnud. Lihtsamaid detaile annab ühest keelest teise automaatselt tõlkida, kuid suuremate detailide korral ei toimi see kahjuks ideaalselt. JavaScripti baasfunktsionaalsusesse kuuluvate funktsioonide tõlkimiseks saab kasutada ka PsychoPy koodikomponendis olevat tõlkijat, mis tõlgib meie poolt sisestatud Pythoni koodi JavaScripti programmeerimiskeelde (Auto->Js). Meie instituudi uurijatel on kogemusi ka selliste katsete üleslaadimisega, mis kasutavad toetamata elemente, kuid see nõuab lisatööd ja head JavaScripti või HTML-i tundmist.*

Veebikatsete jooksutamiseks kasutame PsychoPy veebikeskkonda (pavlovia.org). Üks hea viis end Pavlovia võimalustega kurssi viia on Pavlovia katsete lehel ringi vaadata ja mõni katse lahti klõpsata.

Enne katse veebi laadimist veendume, et PsychoPy toetab meie poolt kasutatud komponentide veebis esitamist. PsychoPy kodulehel ringi vaadates selgub, et üks meie kasutatud komponentidest (*Static*) ei ühildu Pavlovia keskkonnaga ja seega tuleb meil sellest komponendist hetkel loobuda, kuid õnneks ei ole antud ülesande puhul ajastus väga kriitiline ja seetõttu ei ole see tegelikult väga valus kaotus.

Ülesanne 8. Selleks, et saaksime loodud programmi veebi laadida kustutame **Ülesandes 5** lisatud komponendi *Static*. Lisaks tuleks muuta ka sellega seotud pildistiimuli värskendamise seadeid, sest hetkel võttis komponent (pärast *Static* komponendi kustutamist) tagasi oma vaikeväärtuse (*constant*).

Ka praegu pole veel katse päris valmis Pavloviasse laadimiseks. Järgmiseks tuleks meil sisestada JavaScripti aknasse meie poolt eelnevalt sisestatud Pythoni süntaksile (rndisi = random()) vastav JavaScripti süntaks (rndisi = Math.random;). Levinud tõlked PsychoPy koodile leiad sellelt leheküljelt (*PsychoPy Python to* JavaScript *crib sheet*).

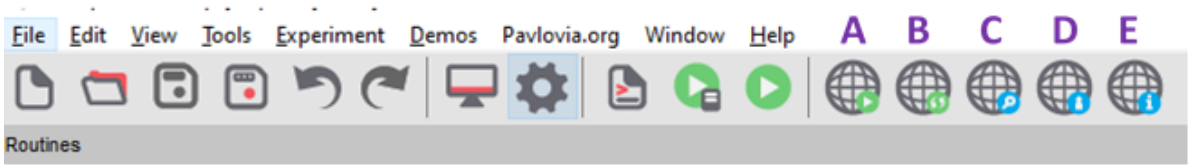
Ülesanne 9. Lisa koodielemendile meie poolt sisestatud Pythoni koodile JavaScripti tõlge.

Selleks klõpsa esmalt koodielemendil. Seejärel vali ülevalt menüüst koodi tüübiks *Both* ja kirjuta *Begin Experiment* saki all parempoolsesse ehk JavaScripti aknasse random = Math.random;

Ülesanne 10. Lae katse Pavloviasse.

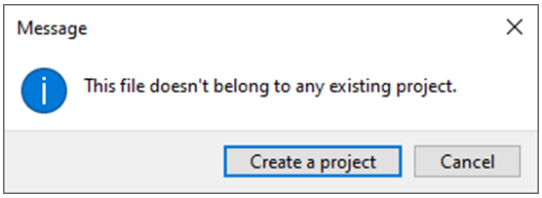
Katse veebi laadimiseks on kaks võimalust:

1) üks võimalus on valida menüüst File > Export HTML või
2) vajutada roheliste noolekestega gloobuse ikoonil (**B**), mis sünkroniseerib meie katse Pavlovia keskkonnaga.

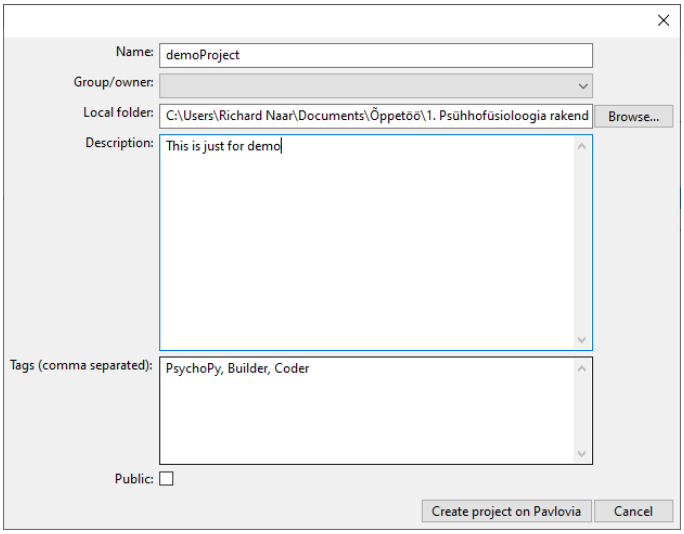


Enne kui saame katse veebi laadida peaksime ennast Pavlovia keskkonda sisse logima. Selleks klõpsame ülevalt ikoonide menüüs sisselogimise ikoonil (**D**).

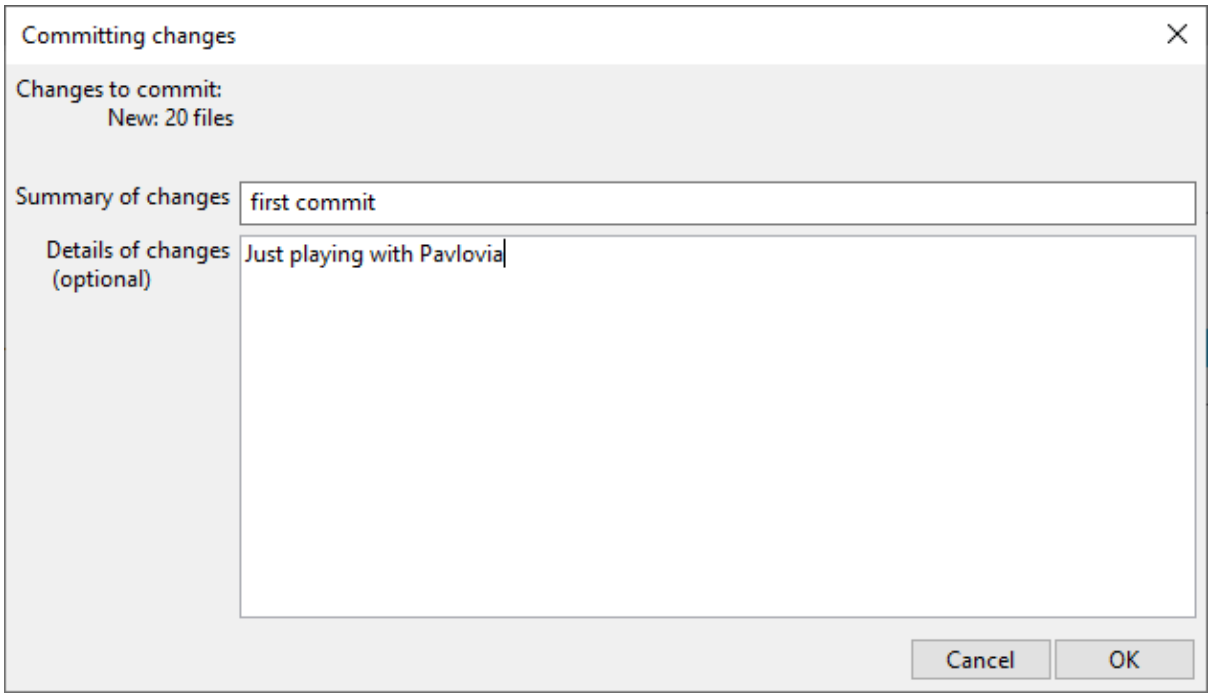
Katse esmakordsel Pavlovia keskkonnaga sünkroniseerimisel ilmub teade, mis ütleb, et loodud katse ei kuulu veel ühegi projekti alla ja seega tuleks meil see projekt kõigepealt luua. Selleks vajutame avanenud hüpikaknas *Create a project*.



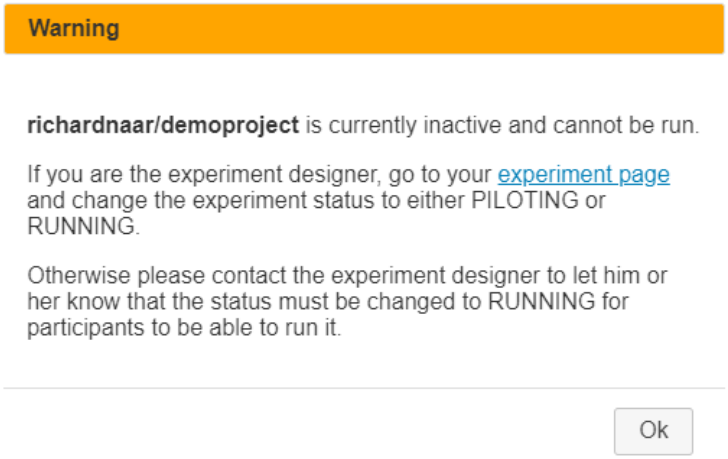
Seejärel avaneb uus aken, kuhu saame sisestada projekti nime ja kirjelduse. Kui oleme soovitud info sisestanud, siis vajutame *Create project on Pavlovia.*



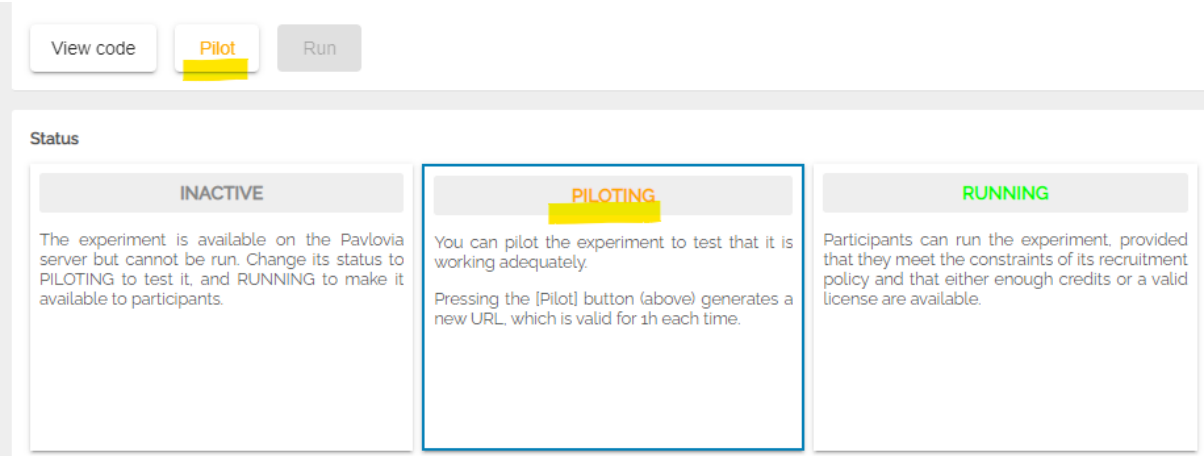
Järgmisena ilmub hüpikaken, mis annab igale meie üleslaadimisele või muudatusele sildi. Kuna tegemist on esimese muudatusega, siis võime *Summary of changes* järele kirjutada näiteks *first commit* või esimene muudatus. Järgmisesse aknasse võib tahtmise korral täpsustada, milliste muudatustega on tegemist.



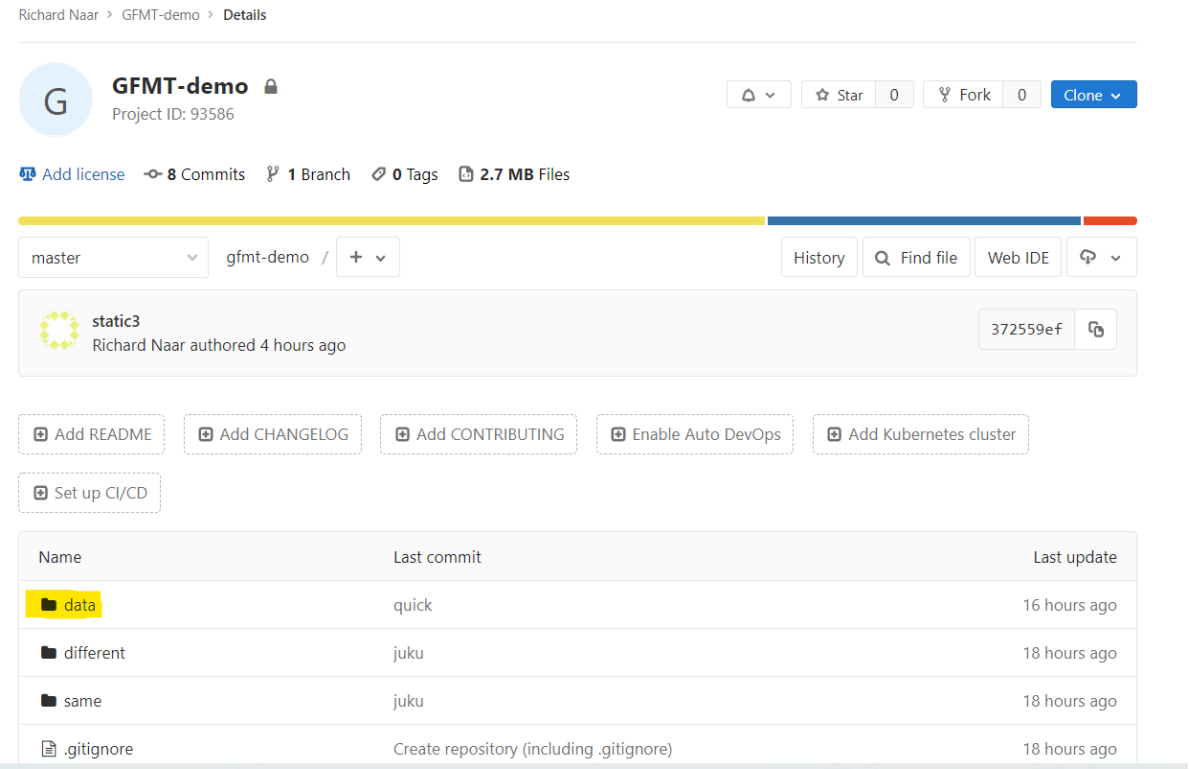
Kui vajutame nüüd veebis käivitamise ikoonil (**A**), siis saame järgmise hoiatuse, mis ütleb, et eksperiment on hetkel mitteaktiivne. Eksperimendi staatuse muutmiseks mine eksperimentide lehel (*experiment page*).



Selleks, et saaksid katset veebis proovida vali katse staatuseks *piloting*. Nupule *pilot* vajutamisel avanebki katse veebiversioon.



Katse veebi laetud koodi saad näha ka enda kasutajaga seotud gitlabi lehel. Lehele jõudmiseks vajuta Pavlovia veebikeskkonnas nupule View code (vaata eelmine joonis).



Samal lehel olevasse kausta *data* jõuavad ka kõik meie veebis kogutud andmed.

Lisaülesanded

Kui sooviksime näha, kas nägude äratundmine sõltub ka sellest, mis pidi nägu esitada, siis võiksime tingimuste tabelisse lisada veel ühe veeru (ori), milles defineerime, kas pilti tuleks esitada õiget pidi ehk selliselt, et selle orientatsioon on 0 kraadi või tagurpidi ehk selliselt, et selle orientatsioon oleks 180 kraadi. Kui soovime, et mõlemas tingimuses näidataks kõiki meie poolt kasutatavaid stiimuleid, peaksime kõiki hetkel tingimuste tabelis olevaid ridu kaks korda esitama. Selleks võime viimase rea alla uuesti read 1 kuni 41 kopeerida. Selliselt on meil tingimuste tabelis kokku 81 rida, millest pooltel on veerus ori kirjas 0 ja pooltel 180.

Lisaülesanne 1. Lisa tingimustabelisse uus veerg ori ja kirjuta kõiki ridu topelt nii, et pooltel juhtudel saab veerg ori väärtuseks 0 ja ülejäänud pooltel 180.

Lisaülesanne 2. Muuda pildi esitamise seadeid selliselt, et pildi orientatsioon muutuks igal seerial vastavalt veerus ori defineeritud väärtusele. Selleks, et programm meie uue tingimuste tabeli kasutusele võtaks peaksime tingimuste tabeli uuesti defineerima. Selleks vajutame *trial* rutiini ümber oleval aasal ja avanenud menüüs värskendame tingimuste faili.

Selles katses ilmuvad pildid järsult ekraanil. Teinekord võib olla vajalik stiimulid sujuvalt ekraanile esitada. Selleks saame kasutada stiimulite kontrastsuse parameetrit. Selles näites kasutame Michelsoni kontrasti (loe selle kohta lähemalt siit), mida saab PsychoPy-s tellida *opacity* parameetrit ehk läbipaistvust muutes. *Opacity'* varieerub nullist üheni, kus null vastab täiesti läbipaistvale ja üks kõrgeima kontrastsusega stiimulile. Kui seame *Opacity'* väärtuse igal värskendusel muutuma vastavalt rutiinielemendi algusajast möödunud ajale (t), siis ilmuvad stiimulid ekraanile sujuvalt ühe sekundi jooksul. Kui tahaksime, et stiimulid saavutaksid kõrgeima kontrastsuse poole kiiremini ehk poole sekundi jooksul, siis annaksime *Opacity'* väärtuseks t*2.

Lisaülesanne 3. Muuda pildistiimulite läbipaistvust (*Opacity*) selliselt, et pildid ilmuksid ekraanile sujuvalt ühe sekundi jooksul.