METHODS 3: MULTILEVEL STATISTICAL MODELLING AND MACHINE LEARNING

COURSE OVERVIEW (SECOND HALF)

W6: Machine Learning Intro

Moving the goal away from explanations towards prediction and getting Python running

W7: Linear Regression Revisited (machine learning)

How to constrain our models to make them more predictive

W8: Logistic regression (machine learning)

Categorizing responses based on informed guesses

W9: Dimensionality reduction, Principled Component Analysis (PCA)

What to do with very rich data?

W10: Organizing and preprocessing messy data

How to clean up?

W11: Final evaluation and wrap-up of course

Ask anything

COURSE OVERVIEW (SECOND HALF)

W6: Machine Learning Intro

Moving the goal away from explanations towards prediction and getting Python running

W7: Linear Regression Revisited (machine learning)

How to constrain our models to make them more predictive

W8: Logistic regression (machine learning)

Categorizing responses based on informed guesses

W9: Dimensionality reduction, Principled Component Analysis (PCA)

What to do with very rich data?

W10: Organizing and preprocessing messy data

How to clean up?

W11: Final evaluation and wrap-up of course

Ask anything

COURSE OVERVIEW (SECOND HALF)

W6: Machine Learning Intro

Moving the goal away from explanations towards prediction and getting Python running

W7: Linear Regression Revisited (machine learning)

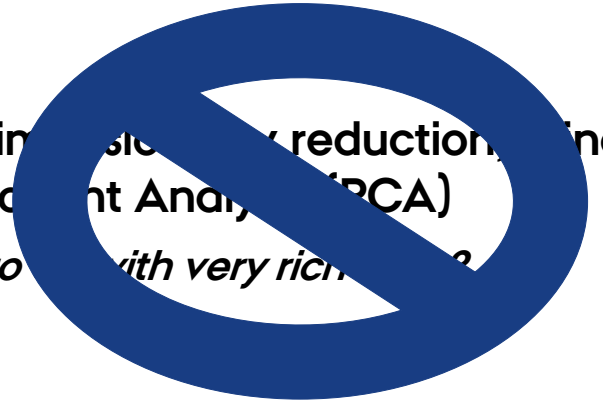
How to constrain our models to make them more predictive

W8: Logistic regression (machine learning)

Categorizing responses based on informed guesses

W9: Dim[illegible] reduction[illegible]ncipled Compo[illegible]nt Anal[illegible]PCA)

What to [illegible]ith very rich[illegible]



W10: Organizing and preprocessing messy data

How to clean up?

W11: Final evaluation and wrap-up of course

Ask anything

TODAYS PLAN

- Slides with tips on assignment
    - Only some tips, since you should be progressing with assignment
- Work on the assignment

TODAYS PLAN

- Catch-up
- .gitignore
    - (being pro-active, and resolving issues)
- Python Class()
    - (overall and with example)
- 3D arrays
    - (averaging, collapsing/flattening)
- Assignment tips
- Assignment code-review
    - Over the shoulder
    - Pair programming

TODAYS PLAN

- Catch-up
- .gitignore
    - (being pro-active, and resolving issues)
- Python Class()
    - (overall and with example)
- 3D arrays
    - (averaging, collapsing/flattening)
- Assignment tips
- Assignment code-review
    - Over the shoulder
    - Pair programming

Anything that seems redundant?
(can skip it – but in doubt as to what has been explained by Lau)

CATCH-UP

CATCH-UP

- How are you holding up?
- Any comments on the course for Lau or me?

CATCH-UP

- Long and tough assignment
- Use me as a resource… Ask(!)/Write
    - Also did the assignment, so might as well utilize it
- I'll be there at the coding-café this Friday also

CATCH-UP

- Feedback from last class:
  - Python classes and basics (.fit concept, etc.)
  - Exercise help (close to coding)
  - Help on Spyder/other IDE's
  - Python workshop?

CATCH-UP

- Feedback from last class:
  - Python classes and basics (.fit concept, etc.)
  - Exercise help (close to coding)
  - Help on Spyder/other IDE's
  - Python workshop?

.GITIGNORE

.GITIGNORE

```
(base) Astrids-MacBook-Pro:week_08 astrid$ git push origin main
Enumerating objects: 52, done.
Counting objects: 100% (51/51), done.
Delta compression using up to 8 threads
Compressing objects: 100% (44/44), done.
Writing objects: 100% (44/44), 130.25 MiB | 8.88 MiB/s, done.
Total 44 (delta 20), reused 0 (delta 0)
remote: Resolving deltas: 100% (20/20), completed with 5 local objects.
remote: error: Trace: be2077ecfc73f8475b1254af6c9a2ed07d6648b25ba1806a1b738e847f9182bf
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File week_08/megmag_data.npy is 133.21 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/AddiH/github_methods_3
 ! [remote rejected] main -> main (pre-receive hook declined)
error: failed to push some refs to 'https://github.com/AddiH/github_methods_3'
(base) Astrids-MacBook-Pro:week_08 astrid$ git rm week_08/megmag_data.npy
fatal: pathspec 'week_08/megmag_data.npy' did not match any files
```

.GITIGNORE

- Live examples
  - Resolving issues
    - git reset –-soft HEAD~1 *(Deletes last commit)*
    - git restore --staged <file> *(untracks file)*
  - Being proactive
    - Create .gitignore (use python troubleshooting pdf)
    - git add .gitignore

.GITIGNORE

- Rather quick live example… But only if interested?
  - Note to self: remember to zoom in bash

PYTHON CLASSES

PYTHON CLASSES

- Only large difference between R and Python
  - *(apart from other types of objects, dict, lists, tuples, etc.)*

PYTHON CLASSES

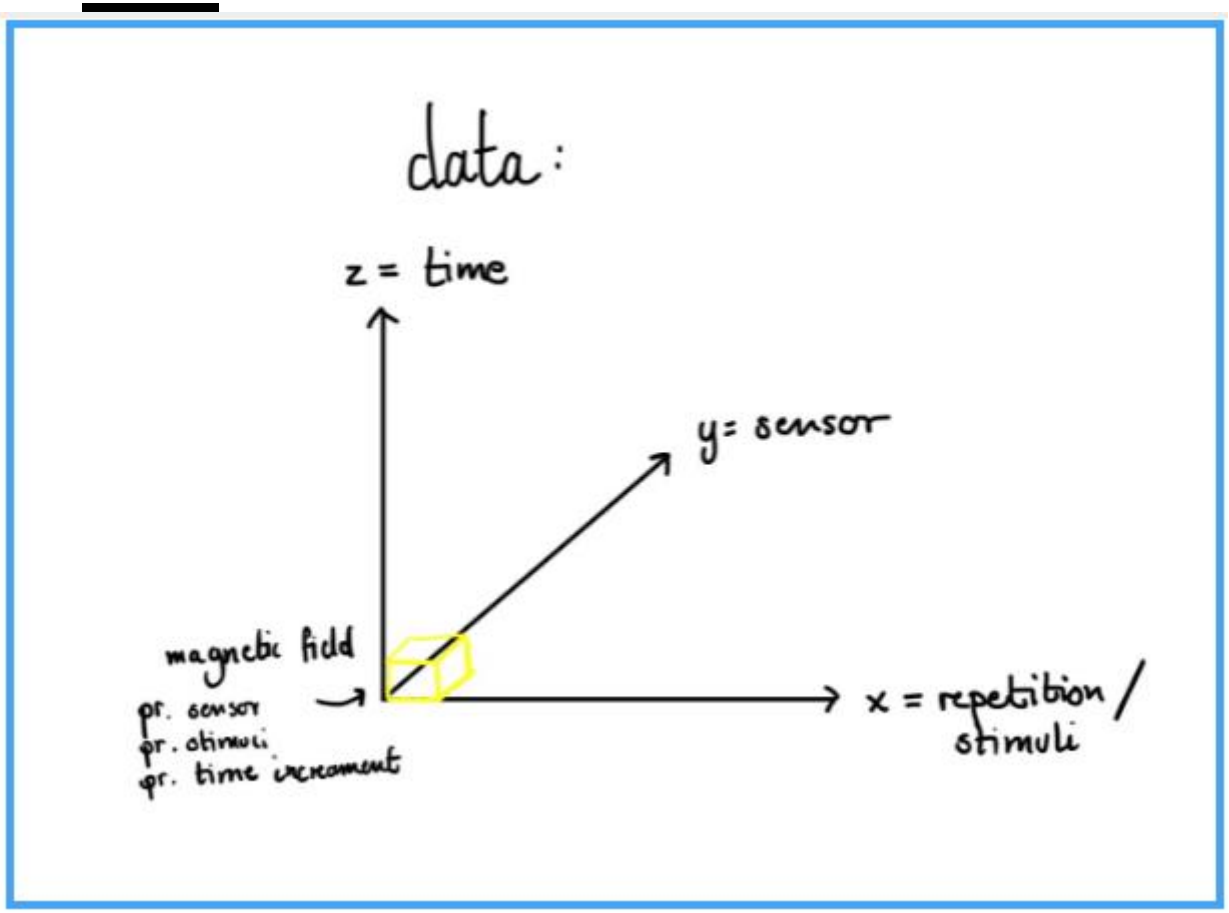
- Going through an example of a Class()
- Won't necessarily give full understanding but …
  - Blogposts
  - Youtube
  - **Trying it out yourself (feel free to use my script also)**

PYTHON CLASSES

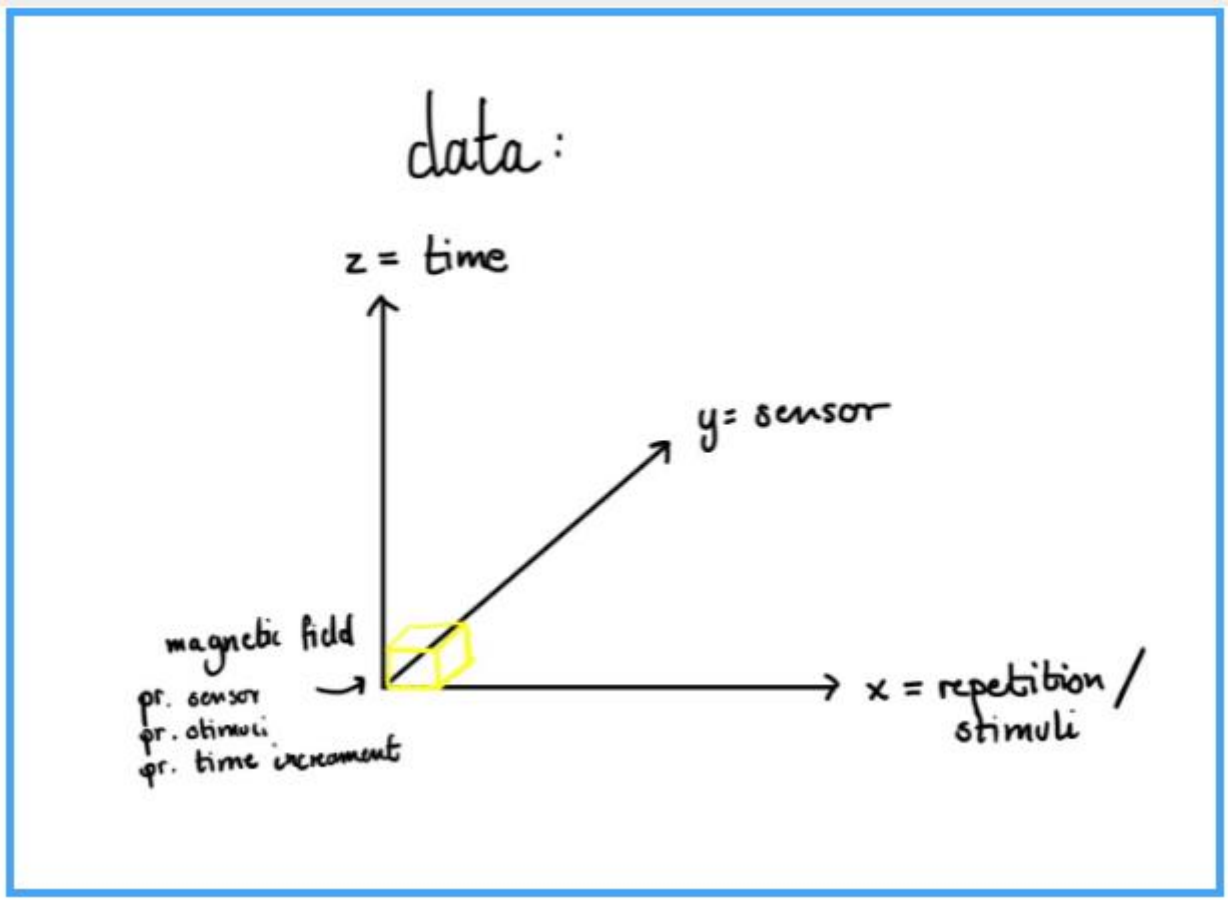
- Live example

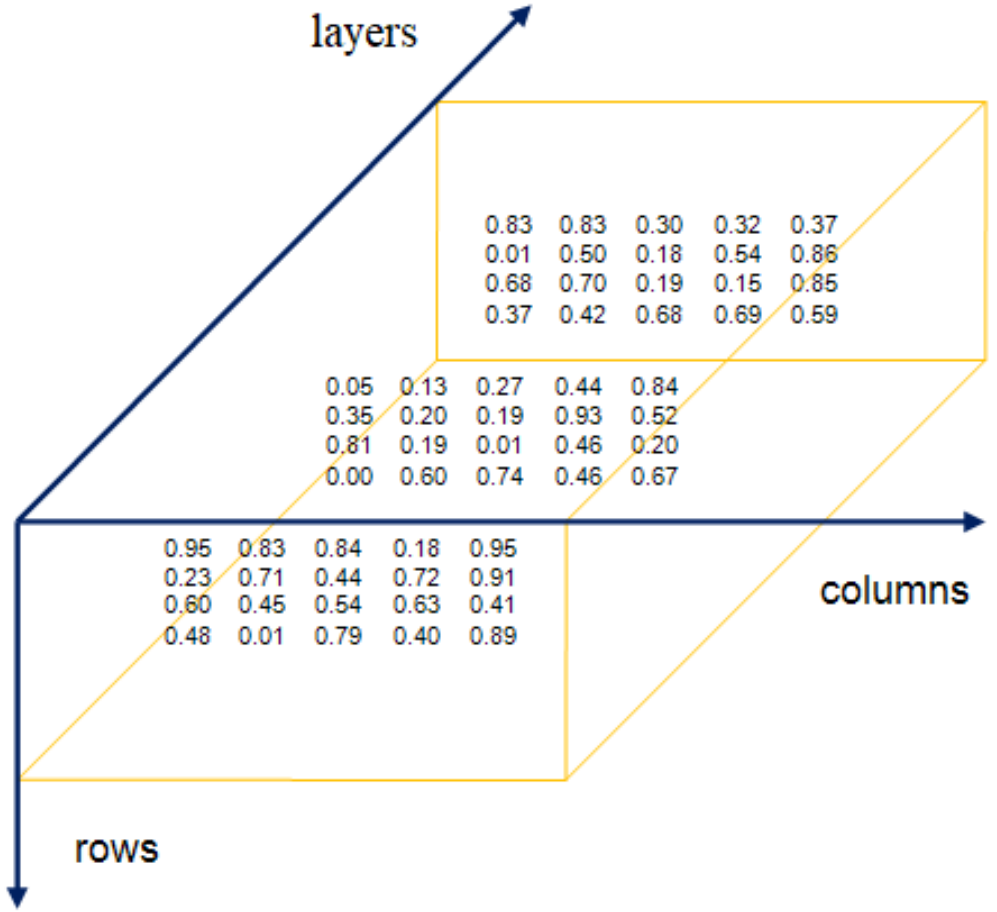
ARRAYS AND MEANS

ARRAYS


data:
z = time
y= sensor
magnetic field
pr. sensor
pr. stimuli
pr. time increament
x = repetition / stimuli

ARRAYS





A 3-D array, with size **4×5×3**, may be described as a "block array" containing three **4×5** matrices (one per page), or also four **5×3** matrices

ARRAYS

- How do they behave?

ARRAYS

```
>>> two_d = np.array([[1,2,3,4],[2,3,4,5]])
>>> two_d
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
```

ARRAYS

```
>>> two_d = np.array([[1,2,3,4],[2,3,4,5]])
>>> two_d
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
>>> two_d.shape
(2, 4)
```

Rows, Columns

=

Roman Catholics

ARRAYS

```
>>> two_d = np.array([[1,2,3,4],[2,3,4,5]])
>>> two_d
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
>>> two_d.shape
(2, 4)
>>> np.mean(two_d, axis=0)
```

ARRAYS

```
>>> two_d = np.array([[1,2,3,4],[2,3,4,5]])
>>> two_d
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
>>> two_d.shape
(2, 4)
>>> np.mean(two_d, axis=0)
array([1.5, 2.5, 3.5, 4.5])
```

~~Mean of each column?~~

Takes mean so that axis 0 is collapsed

ARRAYS

```
>>> three_d = np.array([[[1,2,3,4],[2,3,4,5]], [[10,9,8,7], [7,6,5,4]], [[7,6,5,4], [7,8,9,10]]])
>>> three_d
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
>>> three_d.shape
```

ARRAYS

```
>>> three_d = np.array([[[1,2,3,4],[2,3,4,5]], [[10,9,8,7], [7,6,5,4]], [[7,6,5,4], [7,8,9,10]]])
>>> three_d
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
>>> three_d.shape
(3, 2, 4)
```

Unexpected?

ARRAYS

```
>>> three_d = np.array([[[1,2,3,4],[2,3,4,5]], [[10,9,8,7], [7,6,5,4]], [[7,6,5,4], [7,8,9,10]]])
>>> three_d
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
>>> three_d.shape
(3, 2, 4)
>>> np.mean(three_d, axis=0)
```

Takes mean so that axis 0 is collapsed

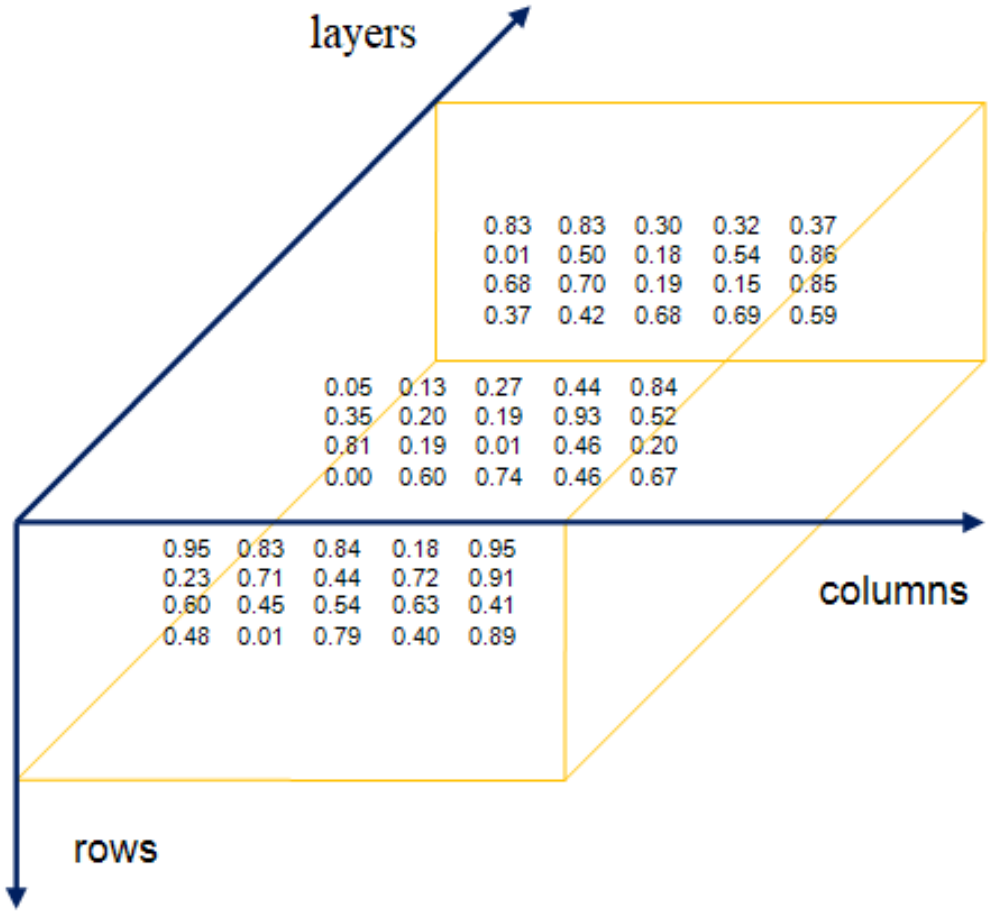
ARRAYS

```
>>> three_d = np.array([[[1,2,3,4],[2,3,4,5]], [[10,9,8,7], [7,6,5,4]], [[7,6,5,4], [7,8,9,10]]])
>>> three_d
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
>>> three_d.shape
(3, 2, 4)
>>> np.mean(three_d, axis=0)
array([[6.        , 5.66666667, 5.33333333, 5.        ],
       [5.33333333, 5.66666667, 6.        , 6.33333333]])
>>> np.mean(three_d, axis=0).shape
(2, 4)
```

ARRAYS

```
>>> np.mean(three_d, axis=0)
```

- What will happen here?
- What are the new dimensions?



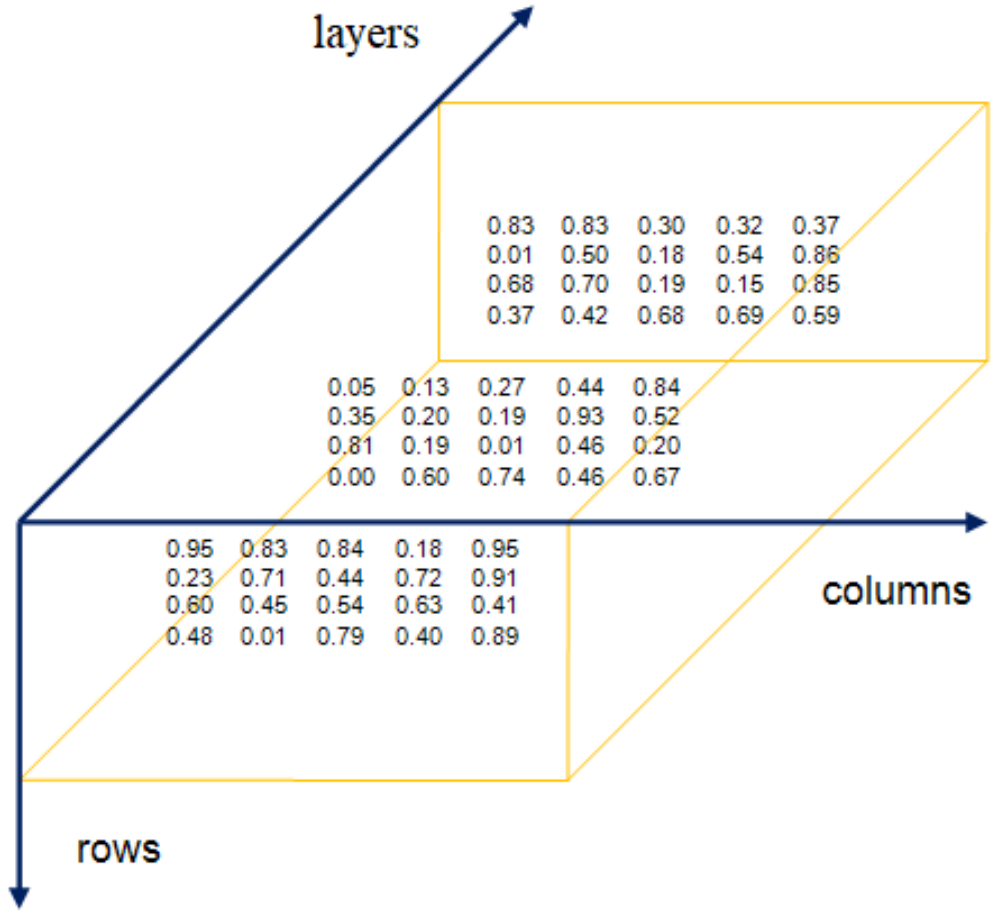
A 3-D array, with size **4×5×3**, may be described as a "block array" containing three **4×5** matrices (one per page), or also four **5×3** matrices

ARRAYS

```
>>> np.mean(three_d, axis=0)
```

- What will happen here?
- What are the new dimensions?
  - *We won't have the dimension[0]*

    -> we won't have depth/layers
  - *New shape = (4,5)*



A 3-D array, with size **4×5×3**, may be described as a "block array" containing three **4×5** matrices (one per page), or also four **5×3** matrices

ARRAYS AND FLATTENING

ARRAYS

- Collapsing/flattening arrays using np.reshape()

ARRAYS

- Collapsing/flattening arrays using

  np.reshape()

```
2.1.ii. Scikit-learn expects our observations (`data_1_2`) to be in a 2d-array, which
has samples (repetitions) on dimension 1 and features (predictor variables) on
dimension 2. Our `data_1_2` is a three-dimensional array. Our strategy will be to
collapse our two last dimensions (sensors and time) into one dimension, while keeping
the first dimension as it is (repetitions). Use `np.reshape` to create a variable
`X_1_2` that fulfils these criteria.
```

ARRAYS

2.1.ii. Scikit-learn expects our observations (`data_1_2`) to be in a 2d-array, which has samples (repetitions) on dimension 1 and features (predictor variables) on dimension 2. Our `data_1_2` is a three-dimensional array. Our strategy will be to collapse our two last dimensions (sensors and time) into one dimension, while keeping the first dimension as it is (repetitions). Use `np.reshape` to create a variable `X_1_2` that fulfils these criteria.

- We want to go from this:

```
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
```

Trials Sensors Timepoints

- Of shape: `(3, 2, 4)`

ARRAYS

2.1.ii. Scikit-learn expects our observations (`data_1_2`) to be in a 2d-array, which has samples (repetitions) on dimension 1 and features (predictor variables) on dimension 2. Our `data_1_2` is a three-dimensional array. Our strategy will be to collapse our two last dimensions (sensors and time) into one dimension, while keeping the first dimension as it is (repetitions). Use `np.reshape` to create a variable `X_1_2` that fulfils these criteria.

- We want to go from this:

```
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
```

Trials Sensors Timepoints

- Of shape: `(3, 2, 4)`

- To this:

```
array([[ 1,  2,  3,  4,  2,  3,  4,  5],
       [10,  9,  8,  7,  7,  6,  5,  4],
       [ 7,  6,  5,  4,  7,  8,  9, 10]])
```

Trials Sensors and timepoints concatenated

- Of shape: `(3, 8)`

ARRAYS

```
2.1.ii. Scikit-learn expects our observations (`data_1_2`) to be in a 2d-array, which
has samples (repetitions) on dimension 1 and features (predictor variables) on
dimension 2. Our `data_1_2` is a three-dimensional array. Our strategy will be to
collapse our two last dimensions (sensors and time) into one dimension, while keeping
the first dimension as it is (repetitions). Use `np.reshape` to create a variable
`X_1_2` that fulfils these criteria.
```

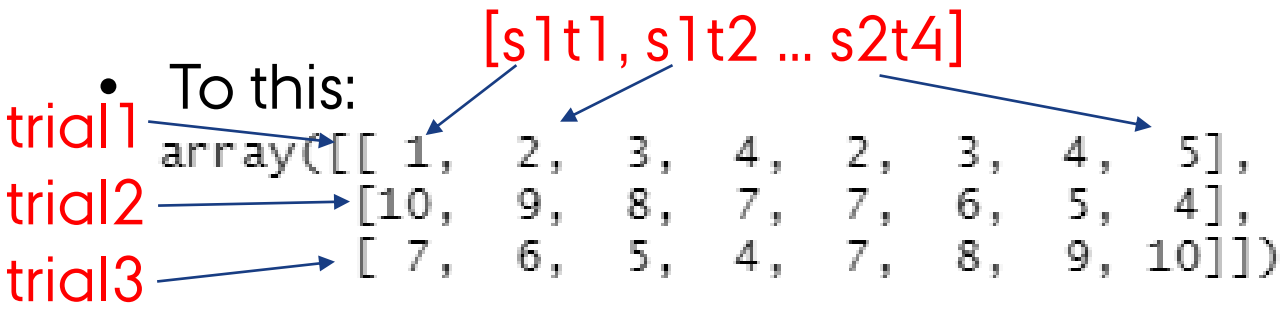
- We want to go from this:

```
array([[[ 1,  2,  3,  4],
        [ 2,  3,  4,  5]],

       [[10,  9,  8,  7],
        [ 7,  6,  5,  4]],

       [[ 7,  6,  5,  4],
        [ 7,  8,  9, 10]]])
```

Trials Sensors Timepoints

- Of shape: `(3, 2, 4)`

- To this:



Trials Sensors and timepoints concatenated

- Of shape: `(3, 8)`

ARRAYS

```
>>> three_d.reshape(3,-1)
array([[ 1,  2,  3,  4,  2,  3,  4,  5],
       [10,  9,  8,  7,  7,  6,  5,  4],
       [ 7,  6,  5,  4,  7,  8,  9, 10]])
```

https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

CODE EXAMPLES

CODE EXAMPLES

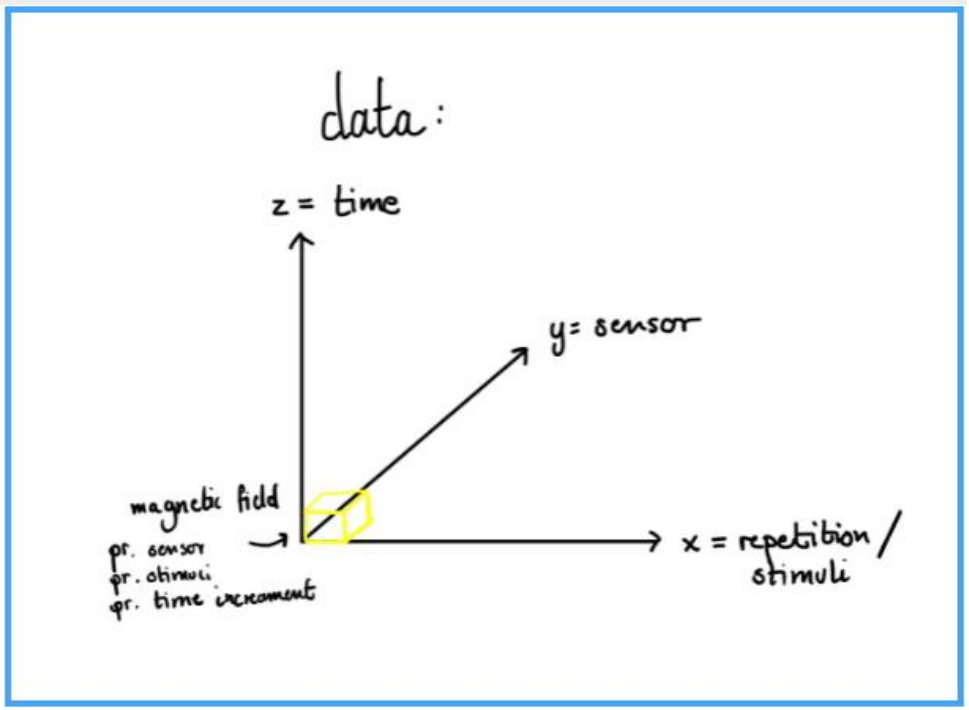
- Can be found on GitHub
  - week_09/support_files/code_for_slides.Rmd
  - week_09/support_files/NumPy.pdf
  - week_09/recapitulation_support_vector_machine.pdf

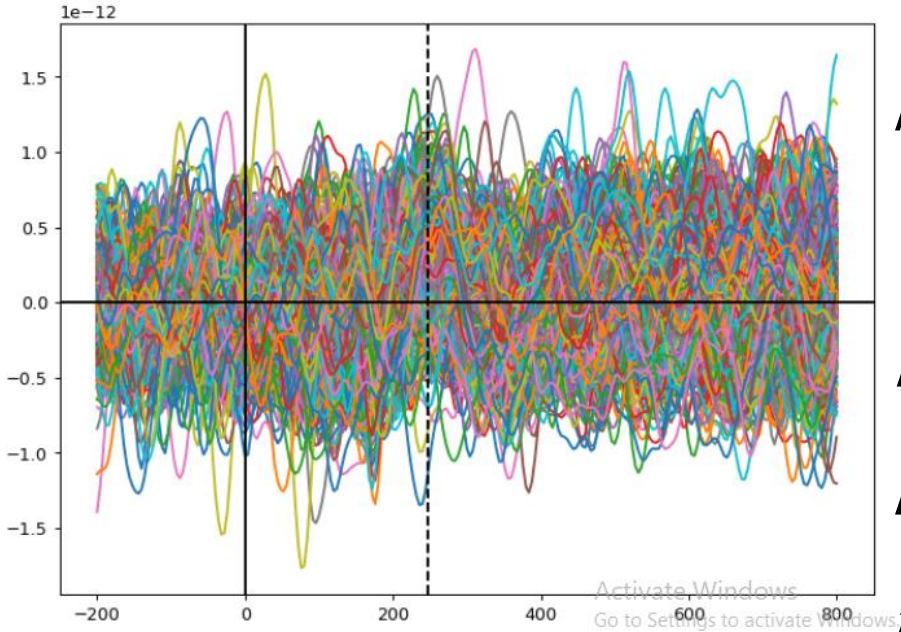
ASSIGNMENT TIPS

ASSIGNMENT TIP

- Conceptual understanding:
  - Have you had neuroscience (and know about epochs/ERP's/evoked signals)?

ASSIGNMENT TIPS



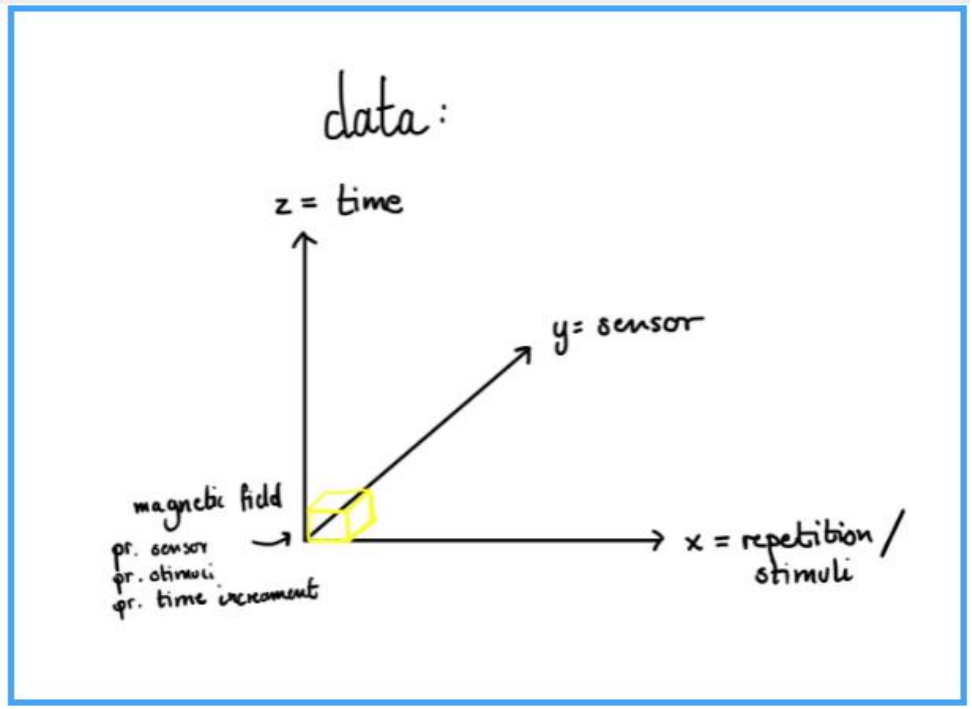


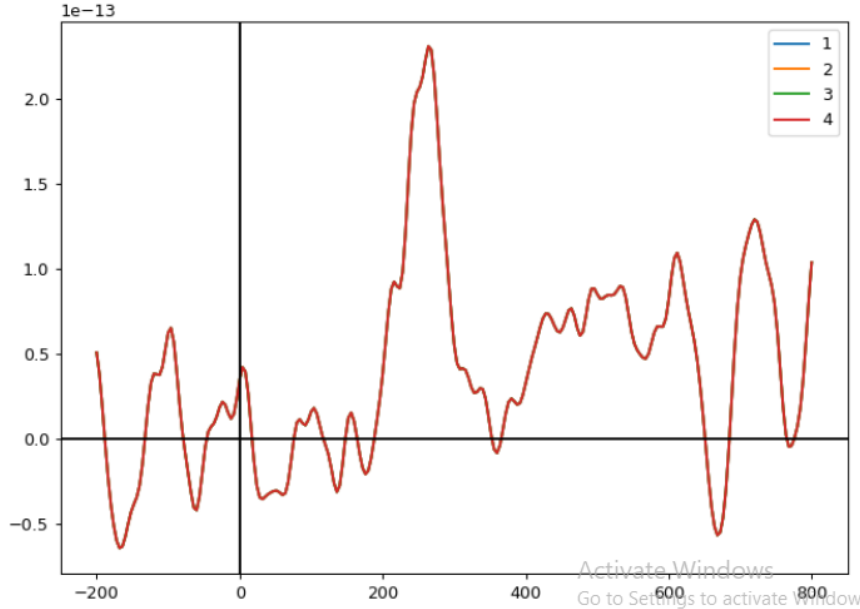
Plot:

Tesla over time, for 1 sensor.

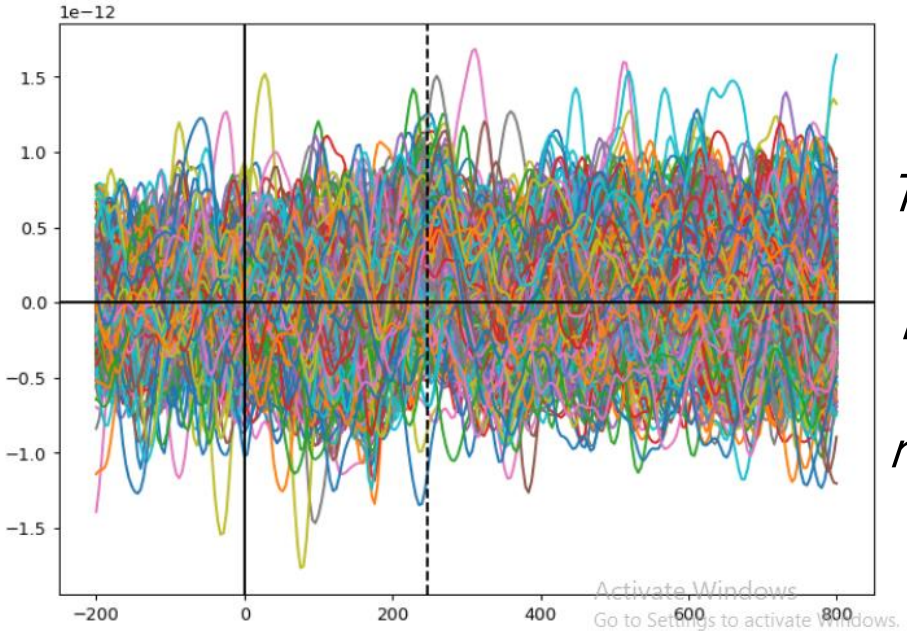
Lines represent trials

ASSIGNMENT TIPS





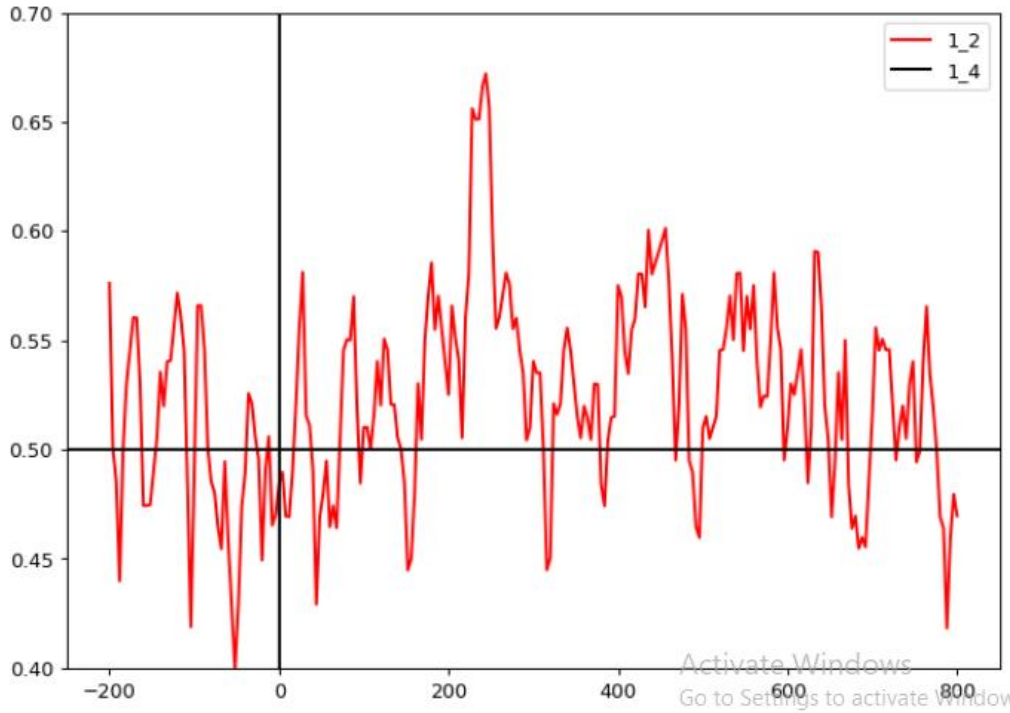
Tesla over time, for 1 sensor. Lines is averaged across trials



Tesla over time, for 1 sensor. Lines represent trials

ASSIGNMENT TIPS

- Conceptual understanding of assignment (as I understand it, at least):
  - Can we train a classifier to predict the PAS rating?



Plot:

Accuracies for 251 models (one for each time point)

ASSIGNMENT TIPS

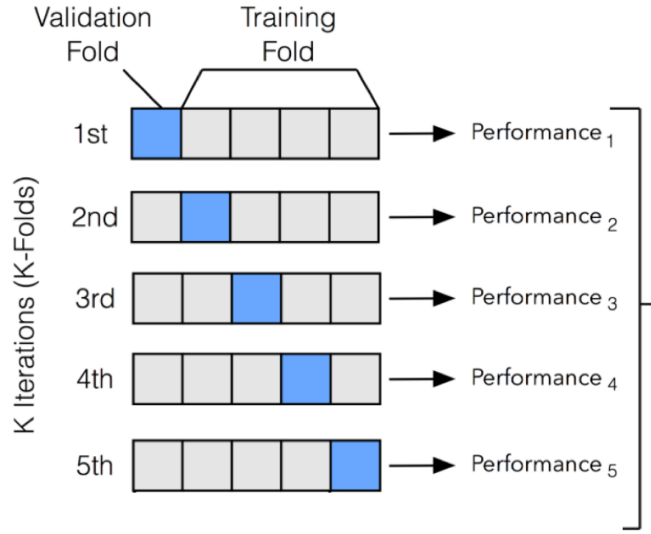
- How to do cross-validation using sklearn? (2.2.ii)
  - cross_val_score()

ASSIGNMENT TIPS

- How to do cross-validation using sklearn? (2.2.ii)
  - cross_val_score()
  - cross_val_score(LinearRegression(), X, y, StratifiedKFold(n_folds=_)
  - This function takes a classifier (class object), data, and a way to split dataset.

ASSIGNMENT TIPS

- How to do cross-validation using sklearn? (2.2.ii)
  - cross_val_score()
  - cross_val_score(LinearRegression(), X, y, StratifiedKFold(n_folds=_)
  - This function takes a classifier (class object), data, and a way to split dataset.
  - The function them performs k-fold cv:
  - Output:
    - List of validation performances (one for each fold)



ASSIGNMENT TIPS

- How to do cross-validation using sklearn? (2.2.ii)

```
lr = LogisticRegression()
cv = StratifiedKFold()
cross_val_score(lr, x, y, cv)
```

ASSIGNMENT TIPS

- Exercise 3:
  - Support vector machines (SVMs)

```
# Import SVC class
from sklearn.svm import SVC

# Define new class object
svm_linear = SVC(kernel='linear', C=1)
```

Radial is called "rbf"

ASSIGNMENT (PEER REVIEW)

ASSIGNMENT (PEER REVIEW)

```
252 ```{python}
253 # Forming groups of group size = 2. Group forming has to be outside study groups
254 groups = group_up(group_size=2, outside_studygroup=True)
255
```

ASSIGNMENT (PEER REVIEW)

```
252 ```{python}
253 # Forming groups of group size = 2. Group forming has to be outside study groups
254 groups = group_up(group_size=2, outside_studygroup=True)
255
256 # For group in groups
257 for group in groups:
258
259   # While time is smaller than 11:10
260   while time < 11:10:
261
262     # Do over the shoulder programming. When finished computing, done == True
263     done = over_the_shoulder(group)
```

ASSIGNMENT (PEER REVIEW)

```
252 ```{python}
253 # Forming groups of group size = 2. Group forming has to be outside study groups
254 groups = group_up(group_size=2, outside_studygroup=True)
255
256 # For group in groups
257 for group in groups:
258
259   # While time is smaller than 11:10
260   while time < 11:10:
261
262     # Do over the shoulder programming. When finished computing, done == True
263     done = over_the_shoulder(group)
264
265     # If done, do pair-wise programming
266     if done = True:
267       pair_wise_programming(group)
268
```

ASSIGNMENT (PEER REVIEW)

```python
# Forming groups of group size = 2. Group forming has to be outside study groups
groups = group_up(group_size=2, outside_studygroup=True)

# For group in groups
for group in groups:

  # While time is smaller than 11:10
  while time < 11:10:

    # Do over the shoulder programming. When finished computing, done == True
    done = over_the_shoulder(group)

    # If done, do pair-wise programming
    if done = True:
      pair_wise_programming(group)

  # When time is not smaller than 11:10
  pair_wise_programming(group)
```

ASSIGNMENT (PEER REVIEW)

```python
# Forming groups of group size = 2. Group forming has to be outside study groups
groups = group_up(group_size=2, outside_studygroup=True)

# For group in groups
for group in groups:

  # While time is smaller than 11:10
  while time < 11:10:

    # Do over the shoulder programming. When finished computing, done == True
    done = over_the_shoulder(group)

    # If done, do pair-wise programming
    if done = True:
      pair_wise_programming(group)

  # When time is not smaller than 11:10
  pair_wise_programming(group)
```

Make sure to ask!

https://github.com/ualsbombe/github_methods_3/blob/main/week_04/practical_exercise_4.pdf

AARHUS
UNIVERSITY