- From PCR Matrix to hamming- distanz matrix (Not hamming-Distanz, sondern euklidische Distanz), due to it is easier to calculate from PCA the Euclid distance.

- LDSC= we must replace our value instead the good value in GWAS-Statistics

- PCR

- Start with snakemake

- Agenda : Errors and etc.

Ein Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

**Methodes to reduce dimension**

There are a few methode to reduce dimension. You can see a list of them and for what they will be use:

1. **Principal Component Anylysis (PCA):**

* Identifizieren “Hauptkomponenten” der Daten
* Minimaler Informationsverlust
* Gut für lineare genetische Daten
* Nicht-lineare korrelation können verloren gehen.

1. **t-Distributed Stochastic Neighbor Embedding (t-SNE)**

* wird zur Visualisierung in 2-3 dimensions angewemdet
* häufig in der Genomik and Proteomik
* lokal Strukturen gut zu erhalten (näher beieinander liegende Datenpunkte im eingebetteten Raum auch beieinanderbleiben.

(gut geeignet)

1. **Uniform manifold Approximation and Projection (UMAP)**

* Ähnlich wie bei t-SNE, aber schneller und oft besser
* Bewahrt die lokale als auch globale Struktur der Daten
* Für die Exploration genetischer Daten und andere komplex strukturierte Datensätze
* Weniger Infos als t-SNE verliert

(Besser geeignet)

1. **Multidimensional Scaling (MDS)**

* Geeignet für visualisierung hochdimensinale Datensätze

1. **Autoencoder:**

* Neuronale Netze
* Eingabedaten auf einen kleinen Latenten Raum zu komprimieren und danach dekpmprimieren
* Nützlich für komplexe genetische Datensätze, insbesondere wenn nichtlineare Beziehungen berücksichtig werden müssen

1. **Kernel PCA:**

* nützlich für Datensätze mit nichtlinearen Strukturen. Durch die Verwendung von Kernel-Methoden kann KPCA komplexe Muster erkennen, die in einem höherdimensionalen Raum linear erscheinen, in dem Raum der ursprünglichen Variablen jedoch nicht linear sind.

**PCA**

**Vorgehensweise der PCA:**

1. Standardisierung der daten

Wird dann gemacht, wenn die Daten verschiedene Maßstäbe haben (jedes Merkmal einen Mittelwert von 0 & eine standardabweichung von 1)

1. Berechnung der Kovarianzmatrix

wie stark die Änderungen in einem Merkmal mit Änderungen in einem anderen Merkmal zusammenhängen. n x n Matrix

1. Eigenwertzerlegung der Kovarianz

Die Eigenwerte und Eigenvektoren der Kovarianzmatrix werden berechnet. Eigenwerte geben die Menge an Varianz an, die von jedem Eigenvektor (Hauptkomponente) erfasst wird.

1. Auswahl der Hauptkomponenten

Die Eigenvektoren werden nach der Größe ihrer Eigenwerte sortiert. Die größten Eigenwerte repräsentieren die Richtungen mit der größten Varianz in den Daten, d.h. die Hauptkomponenten.

1. Projektion der daten auf die Hauptkomponenten

Die Daten werden dann auf die ausgewählten Hauptkomponenten projiziert, um den reduzierten Datensatz zu erhalten.

**Grundlagen:**

**Varianz:** Die Varianz misst, wie stark die Datenpunkte um den Mittelwert verteilt sind.

Kovarianz: Misst, wie zwei Variablen miteinander variieren. Wenn zwei Variablen tendenziell gleichzeitig steigen oder fallen, ist die Kovarianz positiv; wenn eine steigt, während die andere fällt, ist sie negativ.

**Kovarianzmatrix:** Eine Matrix, die die Kovarianz zwischen jedem Paar von Variablen in den Daten darstellt. Es ist eine symmetrische Matrix und gibt die Kovarianz zwischen jedem Merkmalspaar an.

(What we need)

**Eigenwerte und Eigenvektoren:** Ein Eigenvektor einer Matrix ist eine Richtung, in der die Anwendung der Matrix nur den Maßstab des Vektors ändert, nicht aber die Richtung. Der dazugehörige Eigenwert gibt an, um wie viel der Eigenvektor gestreckt oder gestaucht wird. Bei der PCA entsprechen die Eigenvektoren den Hauptkomponenten und die Eigenwerte geben an, wie viel Varianz jede Hauptkomponente erfasst.

**Dimensionalitätsreduktion:** Durch Auswahl der Eigenvektoren mit den größten Eigenwerten (die Hauptkomponenten) und Projektion der Daten auf diese Vektoren wird die Dimensionalität der Daten reduziert.

**Mathematik:**

**Kovarianzmatrix :** Gegeben ein Datensatz *X* mit *m* Merkmalen, ist die Kovarianzmatrix definiert als , wobei die transponierte Matrix von ist.

**Eigenwertzerlegung:** Die Kovarianzmatrix wird in Eigenwerte und Eigenvektoren zerlegt, so dass **.**

**Sortierung:** Die Eigenwerte werden absteigend sortiert und die zugehörigen Eigenvektoren entsprechend geordnet.

**Projektion:** Für die Datenreduktion werden die Eigenvektoren ausgewählt, die den größten Eigenwerten entsprechen, und die Daten werden darauf projiziert, um die neuen Hauptkomponenten zu erhalten: .

**Code:**

|  |
| --- |
|  |

How to use PCA on genetic Data:

<https://genomicsbootcamp.github.io/book/principal-component-analysis-pca.html>

<https://speciationgenomics.github.io/pca/>

<https://rstudio-pubs-static.s3.amazonaws.com/975097_5afd0f5ab3fe4d76a93faccff2fda3c2.html>

<https://adnguyen.github.io/2017_Ecological_Genomics/Tutorial/2017-03-22_Tutorials_PopulGenomics4.html>

<https://bcm-uga.github.io/pcadapt/articles/pcadapt.html>

**t-SNE**

**Datenreinigung**

* Daten in korrekten Format
* fehlende Daten ersetzen oder entfernen

**Normalisierung**

* SNP-Daten müssen normalerweise normalisiert werden:
  + Es gitb 5 Arten von Normalisierung:
    - Standardisierung: Für quantitative Analyse: jedes Merkmal einen Mittelwert von 0 und eine Standardabweichung von 1. Wie? Von jedem Wert den Mittelwert des SNPs abziehen und das Ergebnis durch die Standardabweichung des SNPs teilen.
    - Min-Max-Normalisierung: Datenwerte zwischen einem bestimmten Bereich liegen. (zwischen 0 & 1). Wie? Für jeden SNP wird der minimale Wert des SNPs vom aktuellen Wert abgezogen und dann durch die Differenz zwischen dem maximalen und minimalen Wert des SNPs geteilt.
    - Allelfrequenz-Kodierung: Nützlich für populationsgenetischen Studien.
    - Genotype-Imputation: eigentliche keine Normalisierungsmethode, aber hilft die Vollständigkeit der Daten zu verbessern.
    - Codierung unter Berücksichtigung des Dominanz- oder Additivitätsmodells: im additiven Modell jeder SNP basierend auf der Anzahl der Minor Allele kodiert werden (0,1,2)

(Man kann es anhand software und Tools wie Plink, GCTA und R-Pakete: GenABEL, SNPRelate machen)

**Anwendung von t-SNE**

* Auswahl der Hyperparameter: Auswahl von Anzahl der Komponenten (Die Ziel-Dimensionalität) und der Perplexität
* Durchführung von t\_SNE in Python(scikit-learn):
  + <https://www.datacamp.com/tutorial/introduction-t-sne>
  + <https://builtin.com/data-science/tsne-python>
  + <https://www.datatechnotes.com/2020/11/tsne-visualization-example-in-python.html>

**Berechnung der genetischen Verwandtschaftsmatrix (GRM)**

* Berechnung mit reduzierten Daten
* Analyse der GRM

**UMAP**

**Datenreinigung**

* Daten in korrekten Format
* fehlende Daten ersetzen oder entfernen

**Normalisierung**

**Filterung**

* SNPs basierend auf ihrer Minor Allele Frequency (MAF), Call Rate oder anderen biologischen Überlegungen zu filtern.

**Anwendung von UMAP**

* Parameterauswahl: Anzahl der Dimensionen für die reduzierte Darstellung, die Distnazmetrik (z.B. euklidische Distanz, kosinusähnlichkeit) und die Größe der lokalen Nachbarschaft
* Dimensionsreduktion:
  + <https://www.r-bloggers.com/2019/06/running-umap-for-data-visualisation-in-r/>
  + <https://jchiquet.github.io/ds4m/tutorial_nonlinear_methods.html>
  + <https://www.kaggle.com/code/bextuychiev/beautiful-umap-tutorial-on-100-dimensional-data>

**Berechnung der genetischen Verwandtschaftsmatrix (GRM)**

* Berechnung mit reduzierten Daten
* Analyse der GRM

**Distance Matrix**

1. Euklidische Distanz: Die Standarddistanz im euklidischen Raum, die der "direkten" Entfernung zwischen zwei Punkten entspricht.

2. Manhattan-Distanz (auch City-Block-Distanz genannt): Summe der absoluten Differenzen ihrer Koordinaten. Geeignet für rasterbasierte Daten, wie sie in Stadtumgebungen vorkommen.

3. Chebyshev-Distanz: Die größte Differenz zwischen zwei Vektoren in einer Dimension. Praktisch, wenn die Distanz als der schlimmste Fall über alle Dimensionen hinweg gemessen werden soll.

4. Minkowski-Distanz: Eine Verallgemeinerung der Euklidischen und Manhattan-Distanz. Es beinhaltet einen Parameter p, der variiert werden kann, um verschiedene Distanzen zu erhalten (für p=2) erhalten Sie die euklidische Distanz, für p=1 die Manhattan-Distanz).

5. Kosinus-Distanz: Misst den Kosinus des Winkels zwischen zwei Vektoren im Raum. Gut geeignet für Textdaten oder wenn die Länge der Vektoren nicht von Bedeutung ist.

6. accard-Index: Misst die Ähnlichkeit und Diversität zwischen Sample-Sets. Geeignet für binäre Daten oder Situationen, in denen die Größe der Schnittmenge im Verhältnis zur Größe der vereinigten Menge wichtig ist.

7. Hamming-Distanz: Zählt die Anzahl der Stellen, an denen sich zwei gleich lange Strings unterscheiden. Nützlich für kategoriale Daten.

8. Mahalanobis-Distanz: Berücksichtigt die Kovarianz zwischen den Variablen, um eine Distanz zu berechnen, die die Struktur der Daten einbezieht. Gut geeignet, wenn Daten unterschiedliche Varianzen aufweisen und korreliert sind.

9. Canberra-Distanz: Eine gewichtete Version der Manhattan-Distanz, die Unterschiede in kleinen Dimensionen stärker gewichtet.

10. Pearson-Korrelationsdistanz: Basierend auf dem Pearson-Korrelationskoeffizienten, misst sie die lineare Abhängigkeit zwischen zwei Datensätzen.

11. Spearman'sche Rangkorrelationsdistanz: Basierend auf der Spearman'schen Rangkorrelation, bewertet sie die monotone Beziehung zwischen zwei Datensätzen.

12. genetische Distanz (wie die Nei’s Distanz): Spezifisch für genetische Studien entwickelt, berücksichtigt sie die Wahrscheinlichkeit der Allelverteilung und ist in populationsgenetischen Analysen nützlich.

13. Die IBS-Distanzmatrix (Identical by State): misst die genetische Ähnlichkeit zwischen Individuen basierend auf ihren Allelen an bestimmten genetischen Markern. Sie zeigt, wie viele Allele zwischen den Individuen identisch sind, ohne die Herkunft dieser Allele zu berücksichtigen. Die IBS-Werte reichen von 0 (kein Allel identisch) über 1 (ein Allel identisch) bis 2 (beide Allele identisch). Diese Matrix wird in der genetischen Forschung verwendet, um die genetische Struktur einer Population zu analysieren oder die Verwandtschaft zwischen Individuen zu bestimmen.

**Scikit-Learn**

Eine Open-source-Softwarebibliothek für ML in Python.

**Anwendungen:**

1. Klassifizierung
2. Regression
3. Clustering
4. Dimensionalitätsreduktion
5. Modellauswahl
6. Pipelines

**Code**

|  |
| --- |
| import numpy as np  from sklearn.decomposition import KernelPCA  from sklearn.manifold import TSNE  import umap  # Angenommen, X ist Ihr Datensatz mit den SNP-Daten  # X = ... # Ihr Datensatz hier  # Kernel PCA  kernel\_pca = KernelPCA(n\_components=2, kernel='rbf')  X\_kpca = kernel\_pca.fit\_transform(X)  # t-SNE  tsne = TSNE(n\_components=2, random\_state=42)  X\_tsne = tsne.fit\_transform(X)  # UMAP  umap\_reducer = umap.UMAP(n\_components=2, random\_state=42)  X\_umap = umap\_reducer.fit\_transform(X)  # Jetzt haben Sie X\_kpca, X\_tsne und X\_umap, die die Daten in zwei Dimensionen reduziert darstellen |

|  |
| --- |
| import tensorflow as tf  from tensorflow.keras import layers, models  # Daten laden (hier verwenden wir MNIST als einfaches Beispiel)  (x\_train, \_), (x\_test, \_) = tf.keras.datasets.mnist.load\_data()  # Daten normalisieren und die Dimension anpassen  x\_train = x\_train.astype('float32') / 255.  x\_test = x\_test.astype('float32') / 255.  x\_train = x\_train[..., tf.newaxis]  x\_test = x\_test[..., tf.newaxis]  # Autoencoder-Modell  input\_img = layers.Input(shape=(28, 28, 1))  # Encoder  x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(input\_img)  x = layers.MaxPooling2D((2, 2), padding='same')(x)  x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)  x = layers.MaxPooling2D((2, 2), padding='same')(x)  x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)  encoded = layers.MaxPooling2D((2, 2), padding='same')(x)  # Decoder  x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)  x = layers.UpSampling2D((2, 2))(x)  x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)  x = layers.UpSampling2D((2, 2))(x)  x = layers.Conv2D(16, (3, 3), activation='relu')(x)  x = layers.UpSampling2D((2, 2))(x)  decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)  # Modell instanziieren  autoencoder = models.Model(input\_img, decoded)  autoencoder.compile(optimizer='adam', loss='binary\_crossentropy')  # Autoencoder trainieren  autoencoder.fit(x\_train, x\_train,  epochs=50,  batch\_size=256,  shuffle=True,  validation\_data=(x\_test, x\_test)) |

**Compute der Distanzmatrix**

There are many different distance matrix. But we can calculate the euclide distance matrix.

To do that we can use this formel:

Wobei n die Anzahl der ausgewählten Hauptkomponenten

: die Werte des i-ten und j-ten Datenpunkts auf der K-ten Hauptkomponente.

Es gibt schon dafür einige Funktionen in R und Python, um Distanzmatrix nach PCA zu berechnen.

**Code:**

|  |
| --- |
| from sklearn.metrics.pairwise import euclidean\_distances # Biblothek die wir brauchen  distanzmatrix = euclidean\_distances(„reduzierte Daten nach PCA“) |

**Das Berechnen einer Identität-durch-Abstammung (IBD)-Matrix**

**1. Datenvorbereitung**

- Genetische Daten sammeln: Dies beinhaltet in der Regel SNP-Daten (Single Nucleotide Polymorphisms), die durch Genotyping-Technologien gewonnen werden.

- Qualitätskontrolle: Um sicherzustellen, dass die Daten zuverlässig sind, werden in diesem Schritt typischerweise Individuen und SNPs mit niedriger Qualität oder hoher Fehlerrate entfernt.

**2. IBD-Segmente identifizieren**

- IBD-Segmentierung: Hierbei werden Algorithmen eingesetzt, um Segmente des Genoms zu identifizieren, die zwischen Paaren von Individuen IBD sind. Diese Analyse kann durch Software wie PLINK, BEAGLE oder GCTA durchgeführt werden. Die Software vergleicht Genotyp-Daten zwischen allen Paaren von Individuen, um Abschnitte des Genoms zu finden, die mit großer Wahrscheinlichkeit von einem gemeinsamen Vorfahren geerbt wurden.

**3. Berechnung der IBD-Matrix**

- Matrixkonstruktion: Auf Basis der identifizierten IBD-Segmente wird für jedes Paar von Individuen der Prozentsatz des Genoms berechnet, der IBD ist. Die Einträge in der Matrix repräsentieren den IBD-Anteil oder -Prozentsatz, mit der Diagonale der Matrix, die theoretisch den gesamten genetischen Anteil eines Individuums mit sich selbst repräsentiert (häufig als 100% oder 1 angegeben, wenn sie auf eine Skala von 0 bis 1 normiert ist).

- Software-Nutzung: Für die Berechnung der IBD-Matrix kann Software wie GCTA, PLINK oder andere spezialisierte Tools verwendet werden, die eine Analyse von IBD-Segmenten ermöglichen und daraus eine Matrix erstellen.

**Beispiele für Software-Tools**

- PLINK: Ein weit verbreitetes Tool für die genetische Datenanalyse, das Funktionen für die Identifizierung von IBD-Segmenten und die Berechnung von Verwandtschaftskoeffizienten bietet.

- GCTA: Ein Tool für komplexe Trait-Analysen, das unter anderem zur Schätzung von Verwandtschaftsmatrizen aus genetischen Daten genutzt wird.

- BEAGLE: Ein Software-Tool, das unter anderem für die IBD-Segmentierung und die Phasenbestimmung genutzt wird.

**Code**

|  |
| --- |
| import numpy as np  # Beispiel SNP-Daten (dummy Daten)  snp\_data = np.array([[0, 1, 2, 0],  [2, 0, 1, 0],  [1, 1, 2, 2],  [0, 0, 0, 1]])  # Funktion zur Berechnung der IBD-Matrix  def calculate\_ibd\_matrix(snp\_data):  n\_samples, n\_snps = snp\_data.shape  ibd\_matrix = np.zeros((n\_samples, n\_samples))  for i in range(n\_samples):  for j in range(i, n\_samples):  # Berechne Anzahl der gemeinsamen Allele  shared\_alleles = np.sum(snp\_data[i] == snp\_data[j])    # Berechne IBD-Wert  ibd\_value = shared\_alleles / n\_snps    # Speichern des IBD-Werts in der Matrix (symmetrisch)  ibd\_matrix[i][j] = ibd\_value  ibd\_matrix[j][i] = ibd\_value  return ibd\_matrix  # IBD-Matrix berechnen  ibd\_matrix = calculate\_ibd\_matrix(snp\_data)  print("IBD-Matrix:")  print(ibd\_matrix) |

**Charite und Jupyter:**

ssh charite-front

ssh -N -L 9999:s-sc-node012:2024 charite-front

vim .ssh/config

ssh -m hmac-sha2-512 [sakh13@s-sc-frontend1.charite.de](mailto:sakh13@s-sc-frontend1.charite.de)

ssh sha2-512 [sakh13@s-sc-frontend1.charite.de](mailto:sakh13@s-sc-frontend1.charite.de)

**Nochmal einloggen:**

#Mit dem Server verbinden: ssh charite-front

#get a compute node: srun --nodes=1 --ntasks-per-node=1 --time=08:00:00 --pty --mem=10G -p compute bash -I

# Enviroment aktivieren: conda activate hE

#script für Jupyter:

|  |
| --- |
| #!/bin/bash  #SBATCH --nodes=1  #SBATCH --ntasks=1  #SBATCH --cpus-per-task=1  #SBATCH --mem 100000  #SBATCH -t 48:00:00  #SBATCH --partition=compute  #SBATCH --job-name=jupyter-notebook  #SBATCH --output=/home/khnguyen/errors/jupyternb\_%j.out  #SBATCH --error=/home/khnguyen/errors/jupyternb\_%j.err  ## Define the port that Jupyter listens  port=9482  ## Go to home directory  cd  # Run Jupyter  jupyter notebook --no-browser --port=${port} --ip=0.0.0.0 |

#submission job: sbatch sjupyter.sh

# Check which node your job is running: squeue -u sakh13

# In a new terminal session: ssh -N -L 9999:s-sc-node005:2024 charite-front

# Go to http: //localhost:9999

**Die Links:**

Links: Jupyter Hub: <https://s-sc-hub.charite.de/>

Miniconda: <https://docs.anaconda.com/free/miniconda/index.html>

How to add a conda env: <https://medium.com/@nrk25693/how-to-add-your-conda-environment-to-your-jupyter-notebook-in-just-4-steps-abeab8b8d084>

<https://anaconda.org/bioconda/plink2>

<https://anaconda.org/bioconda/gcta>

<https://anaconda.org/bioconda/ldsc>

**Die Codes von Slack:**

ssh config file:

|  |
| --- |
| host charite-front  Hostname s-sc-frontend1.charite.de  User sakh13  host charite-front2  Hostname s-sc-frontend2.charite.de  User sakh13 |

|  |
| --- |
| Host node???  HostName s-sc-%h.charite.de  User sakh13  ProxyCommand ssh -m hmac-sha2-512 -x -a -q charite-front nc %h 22  DynamicForward 22  Host gpu???  HostName s-sc-%h.charite.de  User sakh13  ProxyCommand ssh -m hmac-sha2-512 -x -a -q charite-front nc %h 22  DynamicForward 22 |

Config um dimensionen zu betrachten:

|  |
| --- |
| # general  plink2: /sc-resources/ukb/data/shared/open/software/plink2  crossmap: /sc-projects/sc-proj-dh-ukb-intergenics/envs/liftover/bin/CrossMap.py  snpeff: /sc-resources/ukb/data/shared/open/software/snpEff/snpEff.jar  download\_path: /sc-projects/sc-proj-dh-ukb- intergenics/raw\_data/finngen/thl/downloads/geno  thl\_withdrawls: /sc-projects/sc-proj-dh-ukb-intergenics/raw\_data/finngen/thl/downloads/geno/withdrawls.txt  processed\_path: /sc-projects/sc-proj-dh-ukb-intergenics/scratch/finngen/thl/processed  hg38to19chain: /sc-projects/sc-proj-dh-ukb-intergenics/raw\_data/liftover/hg38ToHg19.over.chain  hg19fa: /sc-projects/sc-proj-dh-ukb-intergenics/raw\_data/liftover/hg19.fa  # ukb\_pc\_map: /sc-resources/ukb/data/shared/open/files/snp\_pca\_map.txt  ukb\_pc\_map: /sc-resources/ukb/data/shared/open/files/snp\_pca\_map\_more\_ids.txt  dbsnp38: /sc-resources/ukb/data/shared/open/files/human\_9606\_b151\_GRCh38p7-vcf-00-All.vcf.gz  # run-specific  var\_file: none  ds\_location: /sc-projects/sc-proj-dh-ukb-intergenics/scratch/finngen/thl/processed/none  ds\_name: none  score\_name: none  score\_base\_path: /sc-projects/sc-proj-dh-ukb-intergenics/raw\_data/pgscatalog |

Make an env directory:

|  |
| --- |
| mkdir sc-projects/sc-proj-dh-ag-eils-ml/envs |

put this in your .condarc:

|  |
| --- |
| envs\_dirs:  - /sc-projects/sc-proj-dh-ag-eils-ml/envs |

Automatic environment discovery: conda install -c conda-forge nb\_conda\_kernels

Development folder for your directory: /sc-projects/sc-proj-dh-ag-eils-ml

Symlink development directory to home:

|  |
| --- |
| cd  ln -s /sc-projects/sc-proj-dh-ukb-intergenics/analysis/development/sakh13/code |

put an alias in .bashrc to go to the code directory quickly:

|  |
| --- |
| source ~/.bashrc  alias code="cd /sc-projects/sc-proj-dh-ukb-intergenics/analysis/development/sakh13/code" |

For running the jupyter script:

|  |
| --- |
| conda activate dtools  sbatch sjupyter.sh |

**Data:**

the 100k SNP dataset over all chromosomes: /sc-projects/sc-proj-dh-ag-eils-ml/genotype\_data/100k\_snp\_all\_chr.h5ad

the 900k Chromosome 1 SNP dataset: /sc-projects/sc-proj-dh-ag-eils-ml/genotype\_data/900k\_snp\_chr1.h5ad

the plink files for all chromosomes: /sc-resources/ukb/data/bulk/genetic/imputed/bgen\_files\_44448/ukb22828\_<CHROMOSOME>\_b0\_v3.bgen

**Scientific project management**

Is the project

hypothesis-driven?

what is the hypothesis? what can you measure to (dis-)prove it?

is it plausible qualitatively?

is it plausible quantitatively?

exploratory

what signals could (not) be in the data?

what is the best case scenario?

what are you definitely missing?

work with good toy examples:

few samples

few features

something to see

cheap configuration (i.e. few layers, few epochs, ...)

for ML: can you overfit a single sample?

for writing workflows:

think of all of the steps (rules) first (before implmenting details): what goes in, comes out, what happens, what additional parameters are there? are you missing something along the way?

make sure you can see that every step works individually

make sure each step scales --> show that it can do the real (non-toy) task in reasonable time

reproducibility saves time!

Big O time --> scaling with sample number

only optimise things that actually take a lot of time

optimise in the environment where you will run things (i.e. not in a notebook)

fail early with clear targets

start with benchmarks --> known 'expected' result

get one example right first, then scale up

take time to actually think every now and then (+ think together)

prioritize rate limiting steps (incl. paperwork...)

if you are stuck for longer than 30min talk to someone else (start with chatGPT though)

don't automate everything, sometimes its faster to just do something by hand (or chatGPT)

use jupyter notebooks to develop things but move on to scripts / workflows etc at some point - unless you need the interactivity

if a result looks too good to be true it probably is!

- train-test leakage

- trivial problem

- confounding solves narrow case

- bugs...

re-evaluate your baselines? are they (still) the right ones

nothing is perfect: take shortcuts that don't limit you too much, and clearly describe them and the limitations they introduce (e.g. 95 % of snps can be mapped to a gene, throw the rest away if you need genes assigned)

Plan with a buffer of 30 % at least, something always comes up

start the final document at the beginning --> collect ideas, links, etc. --> most stuff becomes either a result or part of the discussion

**Standard tools**

software organistion: Github/gitlab/gitbucket

workflow managers: Snakemake/Nextflow

issue tracking: Github/trello/jira/youtrack/a slack group... -> canban board

package managers: conda/poetry/mamba/virtualenv/pyenv

literature: zenodo/endnote/bibtex/zotero

writing: sharelatex/word/gdocs

ml experiment tracking: w&b/tensorboard/neptune

ml libraries: pytorch/jit/tensorflow/jax

IDE-like jupyter/vscode/pycharm/vim/nano/

plotting: R/matplotlib/seaborn/plotly

data wrangling: pandas/numba

session manager: tmux/screen

**Embedding Data**

Ein Bild, das Screenshot, Text, Diagramm enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Karte enthält.

Automatisch generierte BeschreibungEin Bild, das Screenshot, Text, Diagramm, Kreis enthält.

Automatisch generierte BeschreibungEin Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

**matrizen**

Ein Bild, das Screenshot, Text, Display, Rechteck enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Rechteck, Quadrat enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Display, Rechteck enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Display, Screenshot, Rechteck enthält.

Automatisch generierte Beschreibung