Python for Data Science

BIPN 162

Name A Better Trio. I'll Wait

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

That's it.

Objectives for today

- **Install and import packages for Python**
- Create NumPy arrays
- Execute methods & access attributes of arrays
- Create & manipulate Pandas dataframes
- Introduce the microarray data for today *(& a1!)*

Python supports **modular programming** in multiple ways.

Functions and **classes** are examples of tools for low-level modular programming.

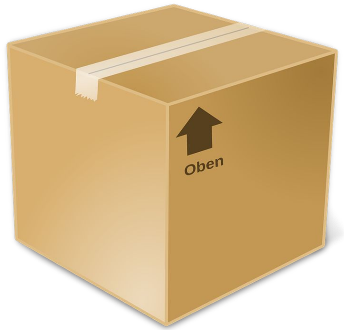
Python **modules** are a higher-level modular programming construct, where we can collect related variables, functions and classes in a module.

Modules are often bundled up into **packages**.

Packages in Python

Python's standard library works for some purposes, but there are many very useful packages for additional purposes:

- **numpy** (http://numpy.scipy.org): numerical Python
- **scipy** (http://www.scipy.org): scientific Python; built on numpy
- **matplotlib** (http://www.matplotlib.org) graphics library



Installing packages & importing modules

To install packages, use

```
$ pip install PACKAGE
```

We typically won't need to do this in the DataHub, because many packages have been installed into our container. However, you *may* need to do this for local notebook operation.

You can then import modules from the package with

```
>>> from PACKAGE import MODULE
```

to see all of the modules available, use

```
>>> print(dir(MODULE))
```

Module	Built-In	Description
csv	Yes	Aids in the reading, writing, and analysis of CSV files.
zipfile	Yes	Aids in the creation and extraction of compressed ZIP archive files.
matplotlib	No	Graphics library for plotting
plotly	No	A graphics library used for creating interactive plots for the web.
seaborn	No	A graphics library built on top of matplotlib with high-quality plots
pandas	No	A data processing library that specializes in data frames, which are analogous to spreadsheets.
scikit-learn	No	Contains basic tools for machine learning (i.e., helping to learn from data and make predictions).
numpy	No	Offers highly efficient data processing.
pygame	No	A game programming library that helps to build interactive, graphical games in Python.
django	No	Web development library that aids in designing websites and web applications.

Objectives for today

- Install and import packages for Python
- **Create NumPy arrays**
- **Execute methods & access attributes of arrays**
- Create & manipulate Pandas dataframes
- Introduce the microarray data for today *(& a1!)*

NumPy is the fundamental package for scientific computing with Python

- A numpy **array** is a grid of values which are all the same type (they're **homogenous**)
- Useful attributes:
  - `ndim` = # of dimensions
  - `shape` = a tuple of integers giving the size of the array along each dimension
  - `dtype` = type of data

Numpy Arrays

my_array = 1D array

3	2	4	1

```
my_array[0] = 3

my_array.ndim = 1

my_array.shape = (4,)

my_array.size = 4
```

2D array

3	2	4	1
1	2	5	3

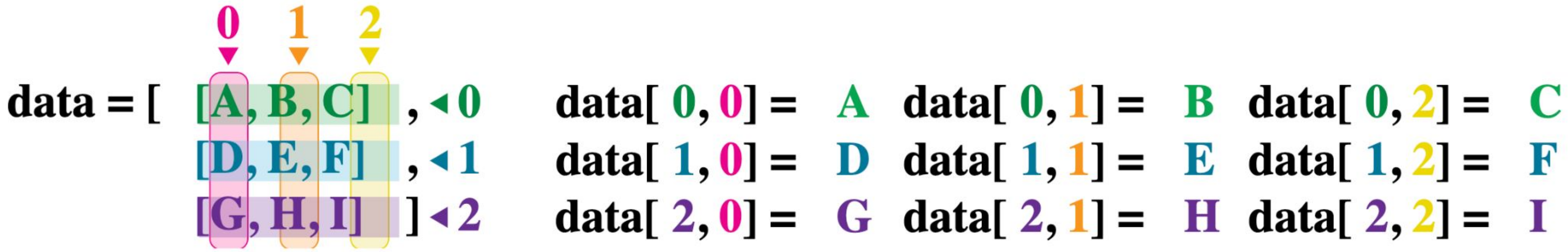
how to index 2D NumPy arrays →

```
my_array[1,3] = 3

my_array.ndim = 2

my_array.shape = (2,4)

my_array.size = 8
```



Indexing numpy arrays

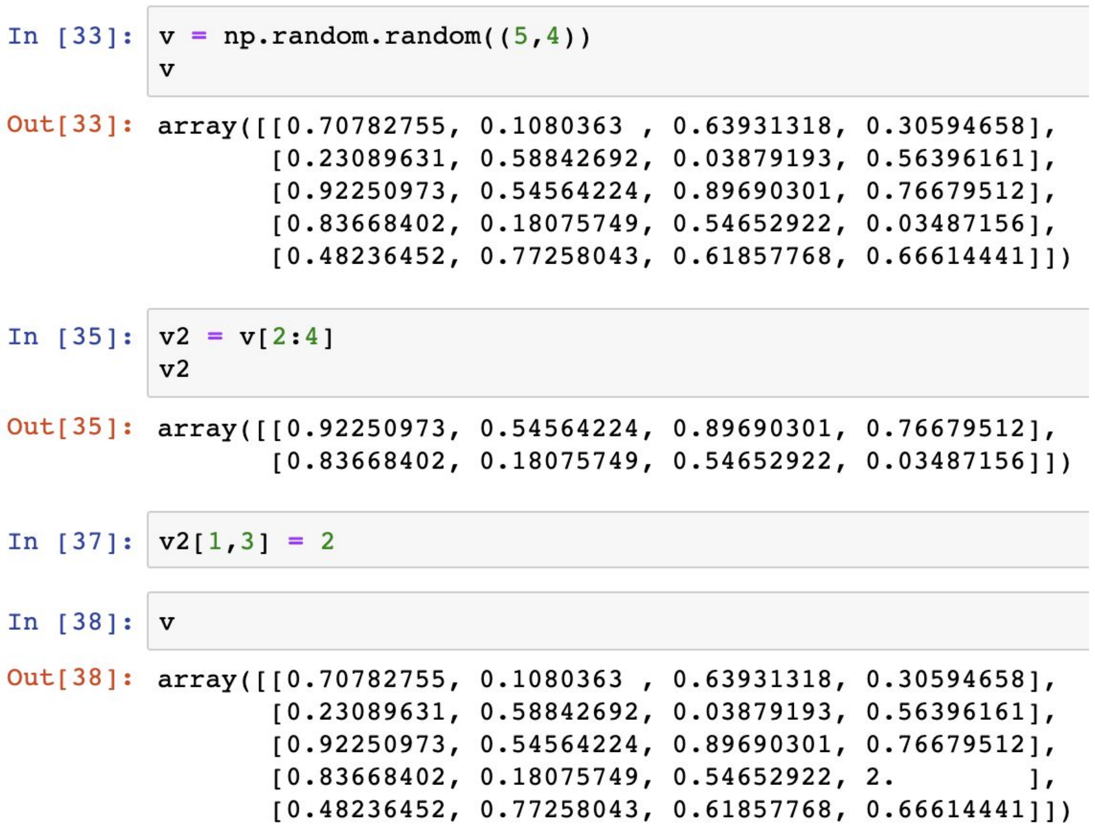
Slicing & indexing NumPy arrays works *almost* the same as with Python lists

However, **be aware that if you slice an array, it changes the original array.**

If you need to copy, you need to explicitly do:

`v3 = v[2:4].copy()`

In this case, we would not change original array (v).



```
In [33]: v = np.random.random((5,4))
         v

Out[33]: array([[0.70782755, 0.1080363 , 0.63931318, 0.30594658],
                [0.23089631, 0.58842692, 0.03879193, 0.56396161],
                [0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 0.03487156],
                [0.48236452, 0.77258043, 0.61857768, 0.66614441]])

In [35]: v2 = v[2:4]
         v2

Out[35]: array([[0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 0.03487156]])

In [37]: v2[1,3] = 2

In [38]: v

Out[38]: array([[0.70782755, 0.1080363 , 0.63931318, 0.30594658],
                [0.23089631, 0.58842692, 0.03879193, 0.56396161],
                [0.92250973, 0.54564224, 0.89690301, 0.76679512],
                [0.83668402, 0.18075749, 0.54652922, 2.        ],
                [0.48236452, 0.77258043, 0.61857768, 0.66614441]])
```

You can also use lists & Booleans to index NumPy arrays

```
my_array[[1,2,3]]
```

```
my_array[my_array > 1]
```

We can also use this to selectively operate on values in the array that meet our criteria:

```
my_array[my_array > 1] = my_array[my_array > 1] * 2
```

Useful NumPy functions

`np.zeros()`

`np.empty()`

`np.linspace()`

`np.arange()`

`np.reshape()`

`np.random.random()`

`np.vstack()`

`np.hstack()`

`np.save()`

`np.load()`

See here for a useful Numpy overview.

Key NumPy takeaways

- Import a library into a program using `import libraryname`
- Use the NumPy library to work with arrays in Python.
- The expression array.shape gives the shape of an array.
- Use `array[x, y]` to select a single element from a 2D array.
- Array indices start at 0, not 1.
- Use `low:high` to specify a slice that includes the indices from low to high-1.
- Use `np.mean(array)`, `np.max(array)`, and `np.min(array)` to calculate simple statistics.
- Use `np.mean(array, axis=0)` or `np.mean(array, axis=1)` to calculate statistics across the specified axis.

Objectives for today

- Install and import packages for Python
- Create NumPy arrays
- Execute methods & access attributes of arrays
- **Create & manipulate Pandas dataframes**
- Introduce the microarray data for today *(& a1!)*

Pandas is a useful module that creates “data frames”

- great for real-world, heterogeneous data
- similar to Excel spreadsheets (but way faster!)
- “numpy with labels”
- Smartly deals with missing data

Numpy:

	0	1	2
0			
1			
2			

Pandas:

	Height	Weight	Age
Amy			
Brad			
Caroline			

df = the dataframe we are operating on!

Useful Pandas methods

`df.mean()` Returns the mean of all columns

`df.corr()` Returns the correlation between columns in a data frame

`df.count()` Returns the number of non-null values in each data frame column

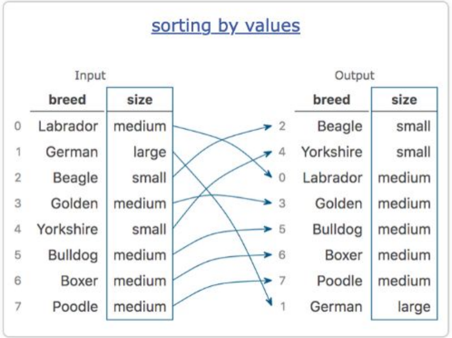
`df.max()` Returns the highest value in each column

`df.min()` Returns the lowest value in each column

`df.median()` Returns the median of each column

`df.std()` Returns the standard deviation of each column

For more useful functions, see this overview.

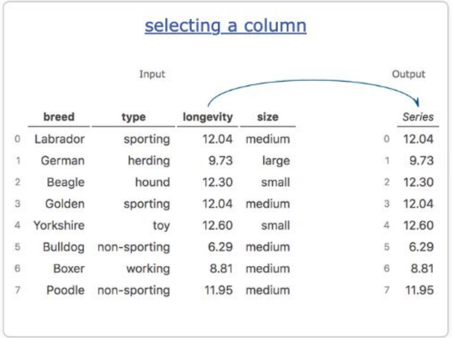

sorting by values

Input

	breed	size
0	Labrador	medium
1	German	large
2	Beagle	small
3	Golden	medium
4	Yorkshire	small
5	Bulldog	medium
6	Boxer	medium
7	Poodle	medium

Output

	breed	size
2	Beagle	small
4	Yorkshire	small
0	Labrador	medium
3	Golden	medium
5	Bulldog	medium
6	Boxer	medium
7	Poodle	medium
1	German	large

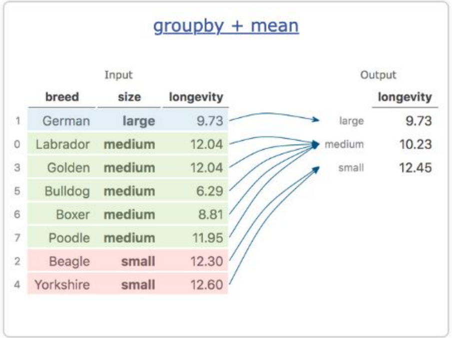

selecting a column

Input

	breed	type	longevity	size
0	Labrador	sporting	12.04	medium
1	German	herding	9.73	large
2	Beagle	hound	12.30	small
3	Golden	sporting	12.04	medium
4	Yorkshire	toy	12.60	small
5	Bulldog	non-sporting	6.29	medium
6	Boxer	working	8.81	medium
7	Poodle	non-sporting	11.95	medium

Output

	Series
0	12.04
1	9.73
2	12.30
3	12.04
4	12.60
5	6.29
6	8.81
7	11.95

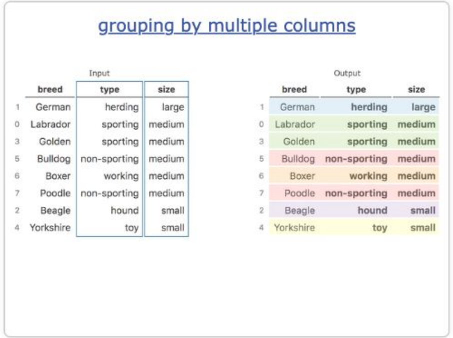

groupby + mean

Input

	breed	size	longevity
1	German	large	9.73
0	Labrador	medium	12.04
3	Golden	medium	12.04
5	Bulldog	medium	6.29
6	Boxer	medium	8.81
7	Poodle	medium	11.95
2	Beagle	small	12.30
4	Yorkshire	small	12.60

Output

	longevity
large	9.73
medium	10.23
small	12.45

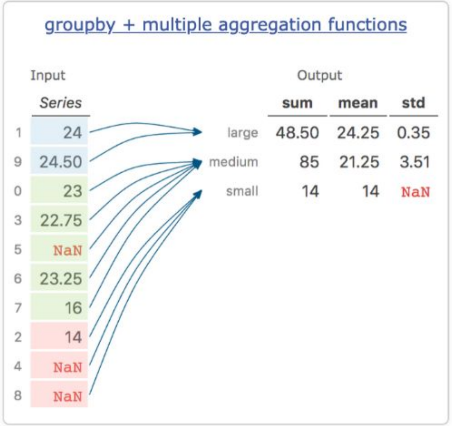

grouping by multiple columns

Input

	breed	type	size
1	German	herding	large
0	Labrador	sporting	medium
3	Golden	sporting	medium
5	Bulldog	non-sporting	medium
6	Boxer	working	medium
7	Poodle	non-sporting	medium
2	Beagle	hound	small
4	Yorkshire	toy	small

Output

	breed	type	size
1	German	herding	large
0	Labrador	sporting	medium
3	Golden	sporting	medium
5	Bulldog	non-sporting	medium
6	Boxer	working	medium
7	Poodle	non-sporting	medium
2	Beagle	hound	small
4	Yorkshire	toy	small

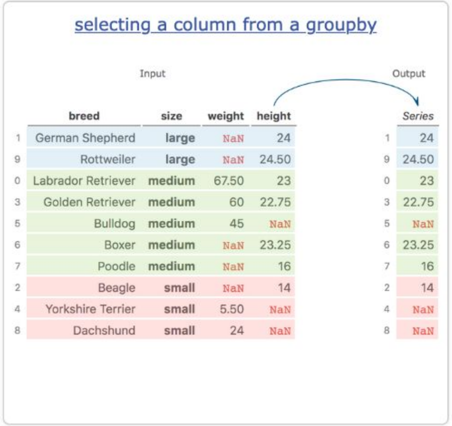

groupby + multiple aggregation functions

Input

	Series
1	24
9	24.50
0	23
3	22.75
5	NaN
6	23.25
7	16
2	14
4	NaN
8	NaN

Output

	sum	mean	std
large	48.50	24.25	0.35
medium	85	21.25	3.51
small	14	14	NaN

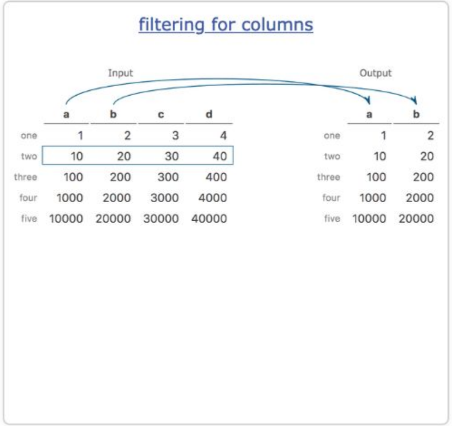

selecting a column from a groupby

Input

	breed	size	weight	height
1	German Shepherd	large	NaN	24
9	Rottweiler	large	NaN	24.50
0	Labrador Retriever	medium	67.50	23
3	Golden Retriever	medium	60	22.75
5	Bulldog	medium	45	NaN
6	Boxer	medium	NaN	23.25
7	Poodle	medium	NaN	16
2	Beagle	small	NaN	14
4	Yorkshire Terrier	small	5.50	NaN
8	Dachshund	small	24	NaN

Output

	Series
1	24
9	24.50
0	23
3	22.75
5	NaN
6	23.25
7	16
2	14
4	NaN
8	NaN

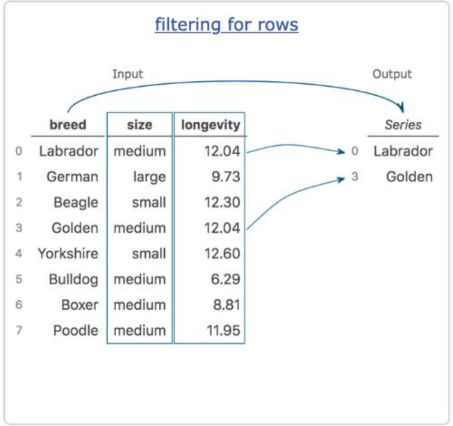

filtering for columns

Input

	a	b	c	d
one	1	2	3	4
two	10	20	30	40
three	100	200	300	400
four	1000	2000	3000	4000
five	10000	20000	30000	40000

Output

	a	b
one	1	2
two	10	20
three	100	200
four	1000	2000
five	10000	20000


filtering for rows

Input

	breed	size	longevity
0	Labrador	medium	12.04
1	German	large	9.73
2	Beagle	small	12.30
3	Golden	medium	12.04
4	Yorkshire	small	12.60
5	Bulldog	medium	6.29
6	Boxer	medium	8.81
7	Poodle	medium	11.95

Output

	Series
0	Labrador
3	Golden

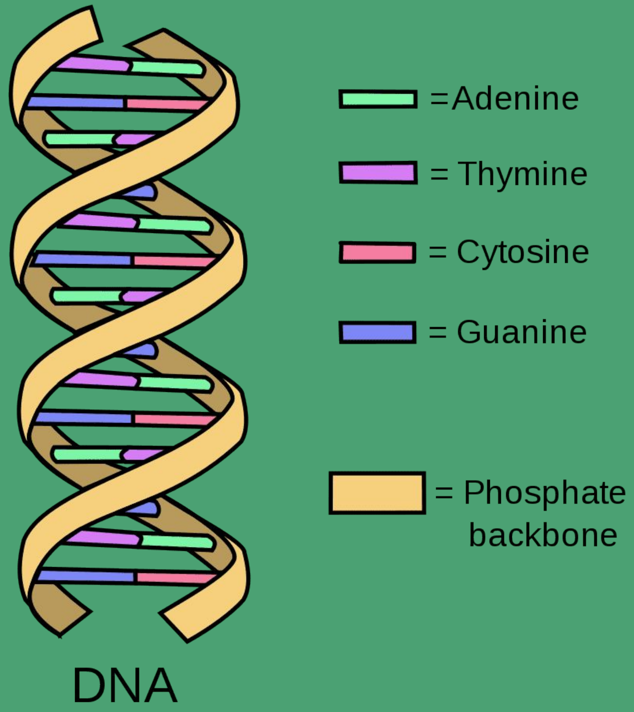
Pandas tutor for visualization

https://pandastutor.com/

Objectives for today

- Install and import packages for Python
- Create NumPy arrays
- Execute methods & access attributes of arrays
- **Create & manipulate Pandas dataframes**
- **Introduce the microarray data for today *(& a1!)***

Why should we care about the brain’s gene expression?



“ Characterizing the complete transcriptional architecture of the human brain will provide important information for **understanding the impact of genetic disorders on different brain regions and functional circuits**.

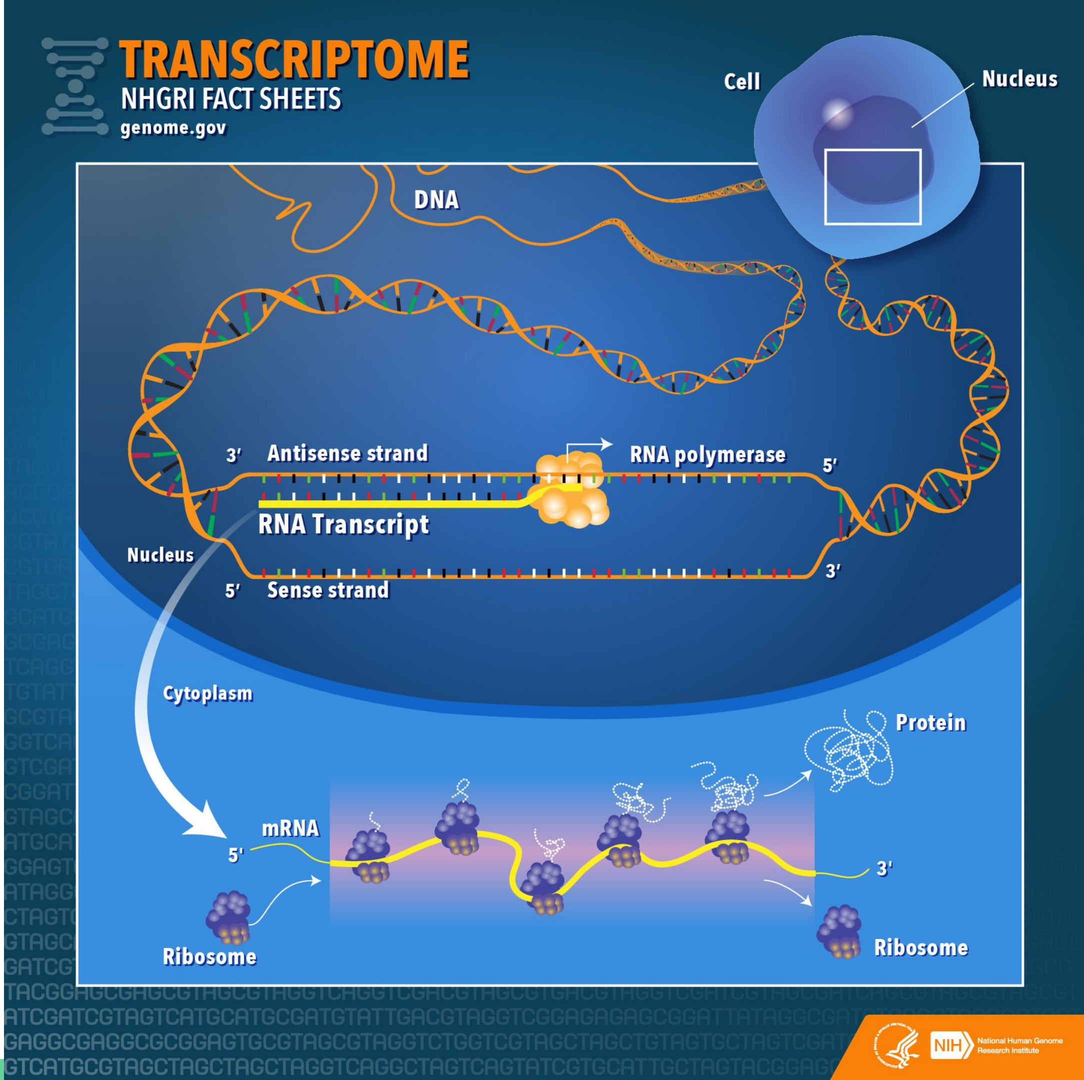
Furthermore, conservation and divergence in brain function between humans and other species provide **essential information for the understanding of drug action**, which is often poorly conserved across species.”

(Hawrylycz et al., 2012)

A few essential reminders

The central dogma: DNA is **transcribed** into mRNA, which is **translated** into protein

- During transcription, a poly(A) tail is added to the mRNA, distinguishing it from other types of RNA that are transcribed
- This poly(A) tail serves as a good way to recognize it with **primers**.



A few essential reminders *(continued)*

- Genes are composed of **introns** and **exons**
  - Exons remain in mRNA, and ultimately code for amino acids
- Most cells contain the same genes, but show different patterns of **transcription**, or **gene expression**
- Genes can have different *alleles*

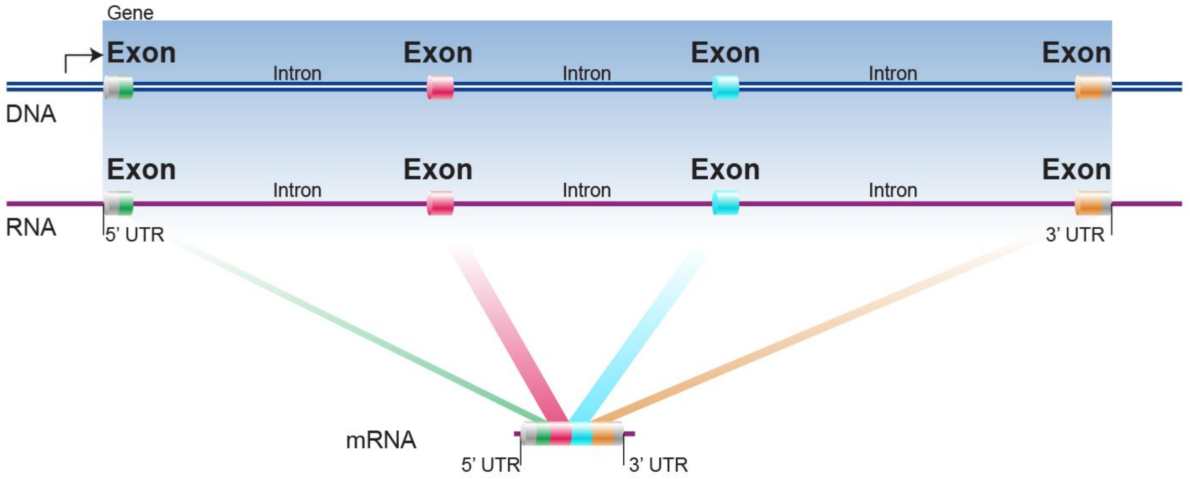


Image: genome.gov

$$\begin{tikzcd} \textbf{genome} \arrow[r] & \textbf{transcriptome} \arrow[r] & \textbf{proteome} \end{tikzcd}$$

coding & non-coding genes

about 20,000-25,000 protein coding genes in humans, ~3 billion DNA base pairs

small percentage of genes (~5%) that are transcribed into readouts (**transcripts**)

has different **splices** of genes

all of the protein translated from mRNA

dynamic & interacting, therefore difficult to study

Important to remember:
RNA != protein

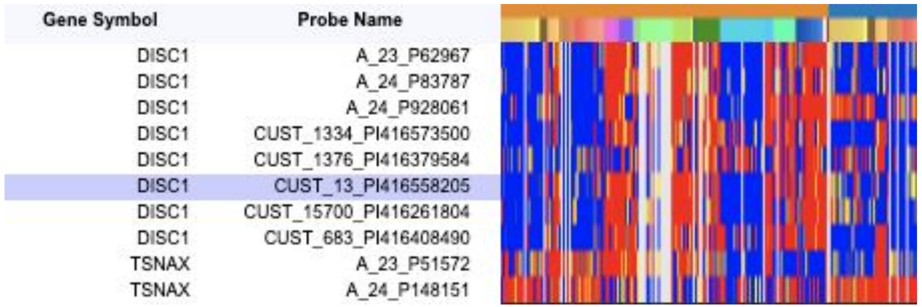
Different ways of measuring gene expression

In situ hybridization

- Probes for DNA or RNA ✓
- Spatial map of gene expression ✓
- Can assay a small number of genes
- Difficult to quantify (Lee et al. 2008)

Microarray

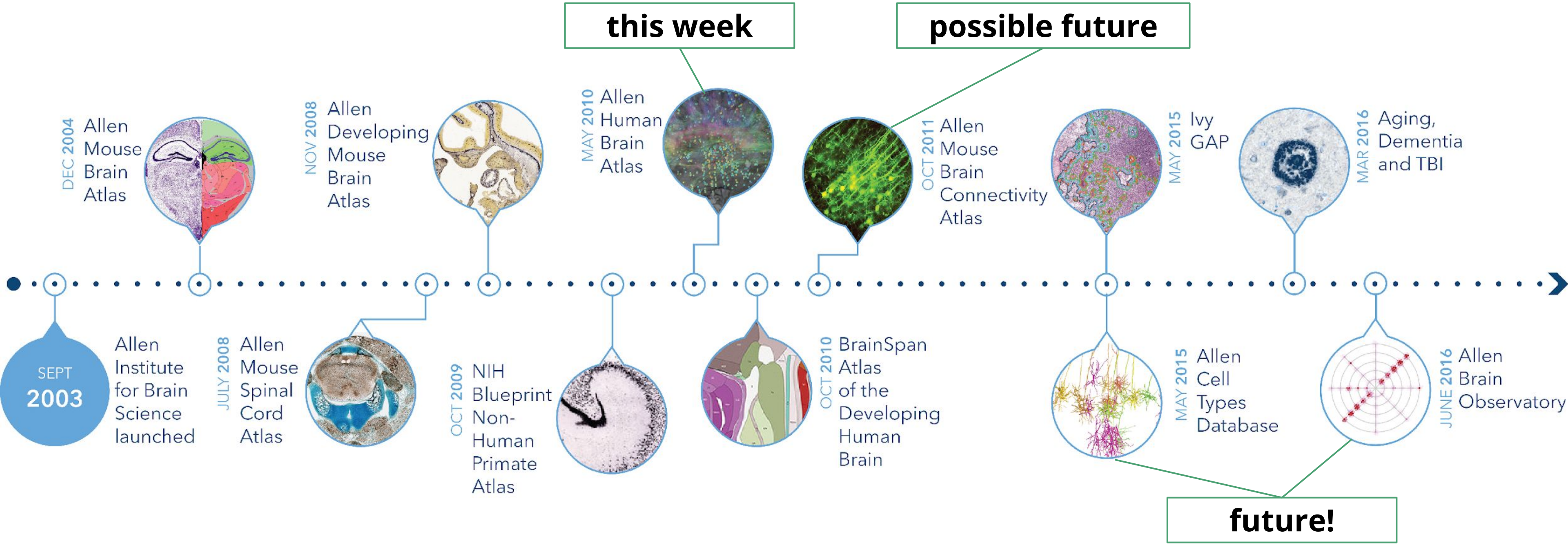
- Probes for DNA or RNA ✓
- Spatial map of gene expression ✓
- Assay 1000s of genes ✓
- Relative quantification ✓



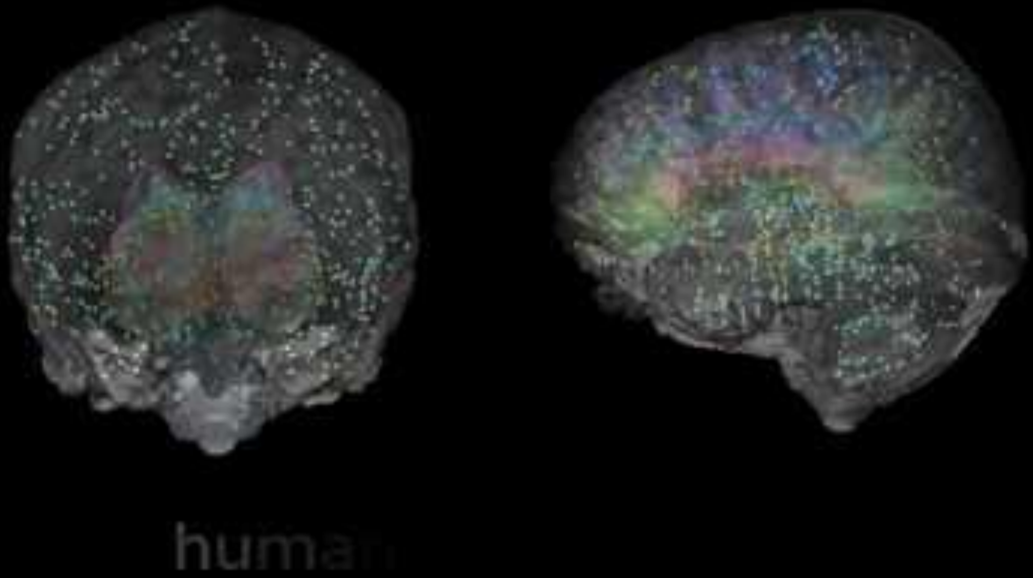
Gene Symbol	Probe Name
DISC1	A_23_P62967
DISC1	A_24_P83787
DISC1	A_24_P928061
DISC1	CUST_1334_PI416573500
DISC1	CUST_1376_PI416379584
DISC1	CUST_13_PI416558205
DISC1	CUST_15700_PI416261804
DISC1	CUST_683_PI416408490
TSNAX	A_23_P51572
TSNAX	A_24_P148151

What do we need to know when working with a dataset?

- **Method** of data collection
  - Was the data collected ethically?
  - Limitations/benefits of the data collection approach
  - Questions we can/cannot answer with the method
- Best practices for **data analysis**
- **Structure** of data set & corresponding metadata


this week
possible future
DEC 2004 Allen Mouse Brain Atlas
NOV 2008 Allen Developing Mouse Brain Atlas
MAY 2010 Allen Human Brain Atlas
OCT 2011 Allen Mouse Brain Connectivity Atlas
MAY 2015 Ivy GAP
MAR 2016 Aging, Dementia and TBI
SEPT 2003 Allen Institute for Brain Science launched
JULY 2008 Allen Mouse Spinal Cord Atlas
OCT 2009 NIH Blueprint Non-Human Primate Atlas
OCT 2010 BrainSpan Atlas of the Developing Human Brain
MAY 2015 Allen Cell Types Database
JUNE 2016 Allen Brain Observatory
future!

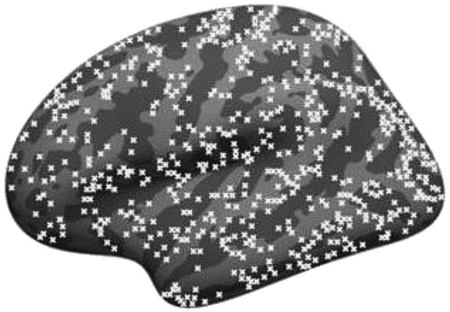
History & Contributions of the Allen Brain Institute

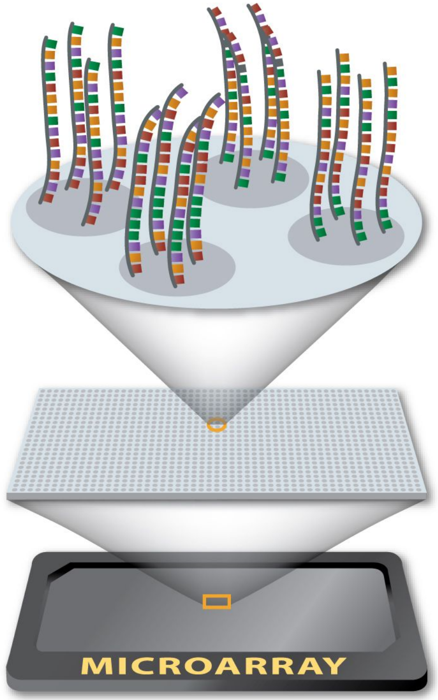


Video describing the acquisition of human microarray data

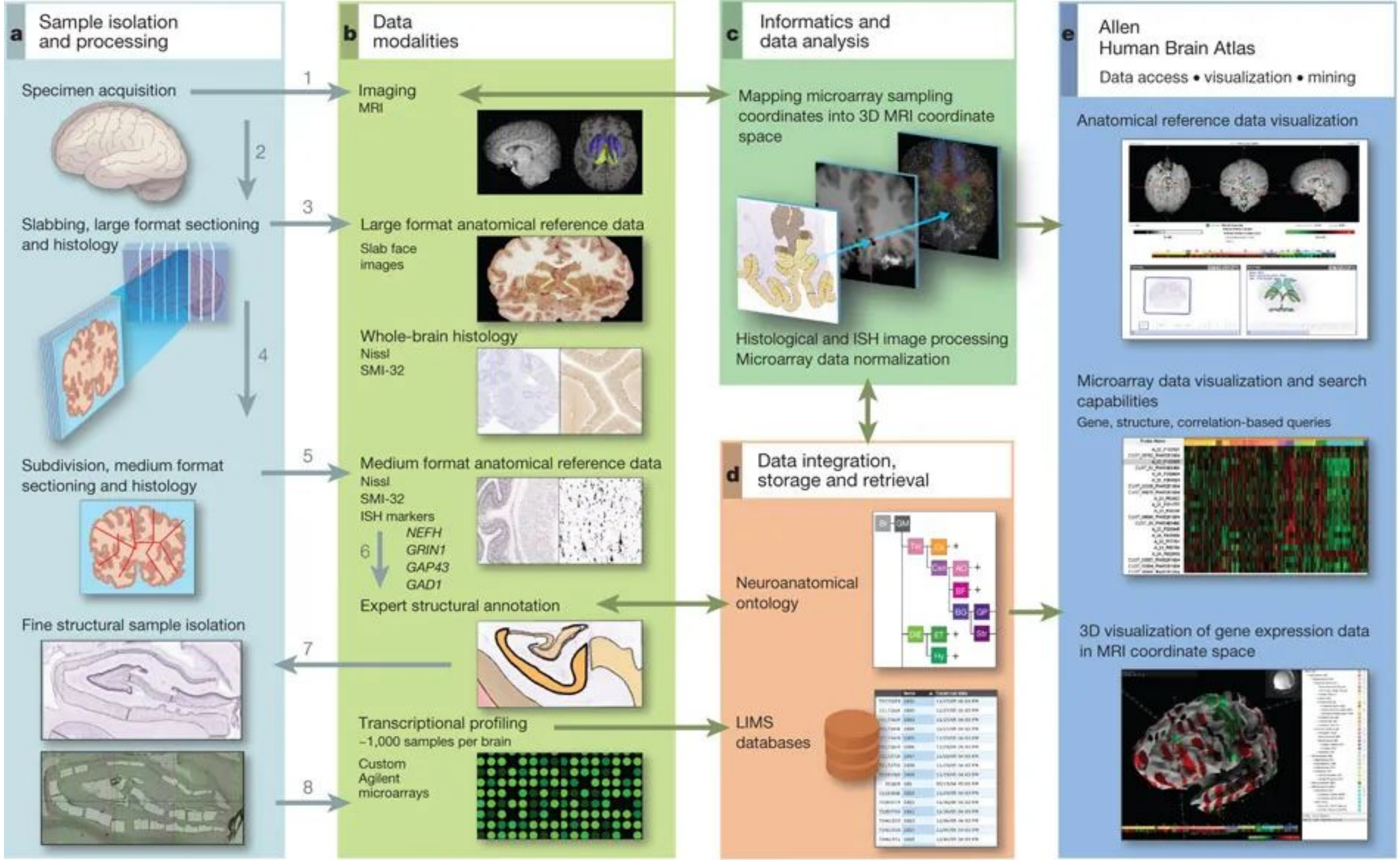
Summary of the ABA Human Microarray Dataset

- 6 different subjects (5 male, 1 female, ages 18-68 years)
  - No known neuropsychiatric or neuropathological history
  - *Details here*
- ~500 brain regions
- Sequenced with microarray chip

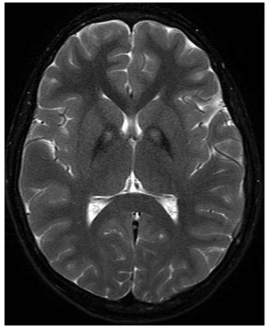


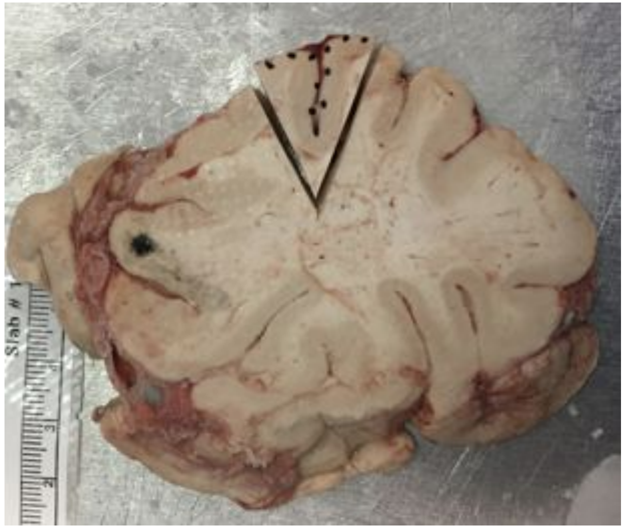


Images: Gryglewski et al. (2018) & Utah Genetics


a Sample isolation and processing
Specimen acquisition
Slabbing, large format sectioning and histology
Subdivision, medium format sectioning and histology
Fine structural sample isolation
b Data modalities
Imaging
MRI
Large format anatomical reference data
Slab face images
Whole-brain histology
Nissl
SMI-32
Medium format anatomical reference data
Nissl
SMI-32
ISH markers
NEFH
GRIN1
GAP43
GAD1
Expert structural annotation
Transcriptional profiling
~1,000 samples per brain
Custom Agilent microarrays
1
2
3
4
5
6
7
8
c Informatics and data analysis
Mapping microarray sampling coordinates into 3D MRI coordinate space
Histological and ISH image processing
Microarray data normalization
d Data integration, storage and retrieval
Neuroanatomical ontology
LIMS databases
e Allen Human Brain Atlas
Data access • visualization • mining
Anatomical reference data visualization
Microarray data visualization and search capabilities
Gene, structure, correlation-based queries
3D visualization of gene expression data in MRI coordinate space

Overview of process



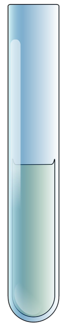


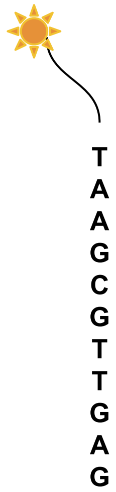
Post mortem brain donation

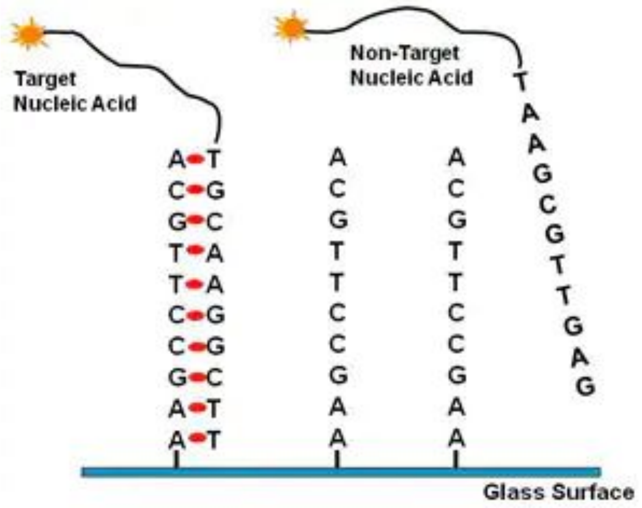
MRI scan on full brain
http://human.brain-map.org/mri_viewers/data

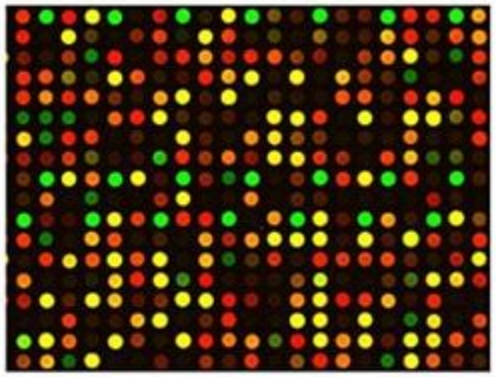
Brain macro- or microsectioned (laser)

Detailed protocol: https://www.protocols.io/view/human-tissue-slicing-and-dissections-for-nuclear-i-7aehibe/abstract
Full documentation: https://help.brain-map.org/display/humanbrain/Documentation









DNA Microarray

Extract mRNA from samples

Create cDNA (via reverse transcriptase) **attached to fluorophore**

Bind to a microarray chip with Whole Human Genome probe set

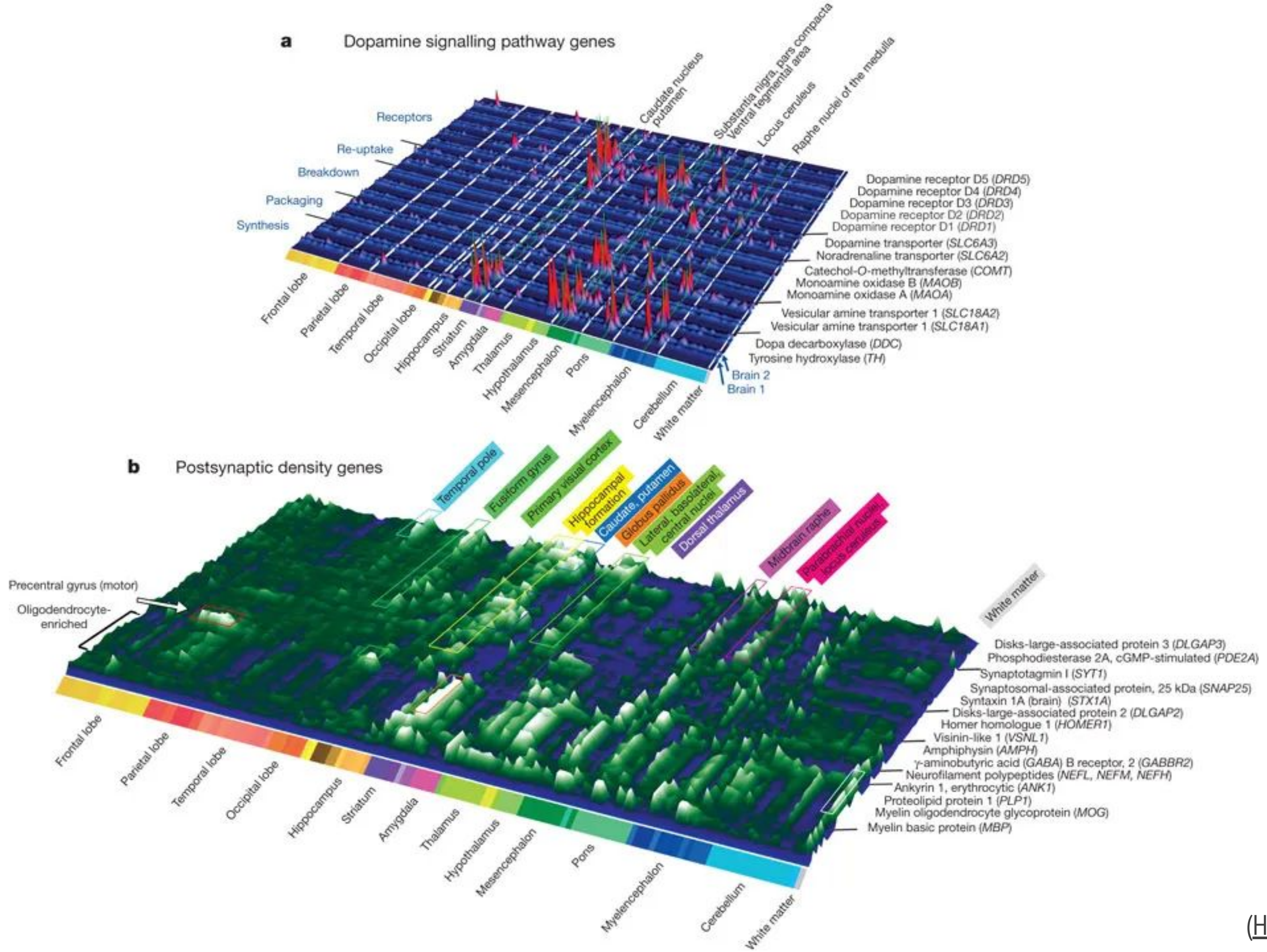
Positive controls: Pooled RNA samples from same brain, and other brains

Negative control: nuclease free deionized water (NFdH2O)

Normalize expression across all samples (from one subject)

and later, across batches of experiments.

Images: Agilent



(Hawrylycz et al., 2012

Resources

NumPy

NumPy quickstart — NumPy v1.26 Manual

NumPy: the absolute basics for beginners

Numerical & Scientific Computing with Python: Introduction into NumPy

Lecture-2-Numpy.ipynb

Analyzing Patient Data – Programming with Python

Pandas

A Quick Introduction to the “Pandas” Python Library

10 minutes to pandas — pandas 1.0.5 documentation

Pandas Tutor

Python Data Science with pandas