Machine Learning and Databases in Astronomy

Managing data & projects and an introduction to regression techniques

https://github.com/jbrinchmann/MLD2019

Why are you here?

The old view:

Why are you here?

https://imgs.xkcd.com/comics/machine_learning.png

~~Why are you here?~~

What is machine learning (in this course)?

https://imgs.xkcd.com/comics/machine_learning.png

What is machine learning (in this course)?

My definition:

Techniques to extract patterns/information/structure from data

There are many other definitions around, the Wikipedia one is fairly broad:

> *Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence.*

https://en.wikipedia.org/wiki/Machine_learning

This, from From "Deep Learning", Goodfellow, Bengio, Courville, is also good:

"Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimated complicated functions and a decreased emphasis on prov[id]ing confidence intervals around these functions"

Examples outside of astronomy:

Amazon recommends .....

Use the buying patterns of all customers to create characteristic directions (books) in a multi-dimensional space. Apply this to individual customers to predict their preferences.

Examples outside of astronomy:

Amazon recommends .....

Use the buying patterns of all customers to create characteristic directions (books) in a multi-dimensional space. Apply this to individual customers to predict their preferences.

Organisation of merchandise in a supermarket

Find clusters of products that are bought together. Ensure you don't discount both at the same time.

Examples outside of astronomy:

Amazon recommends .....

Use the buying patterns of all customers to create characteristic directions (books) in a multi-dimensional space. Apply this to individual customers to predict their preferences.

Organisation of merchandise in a supermarket

Find clusters of products that are bought together. Ensure you don't discount both at the same time.

Associating genome sequences

Identify groups of genomes, cross-correlate, look for structure

Examples outside of astronomy:

Amazon recommends .....

Use the buying patterns of all customers to create characteristic directions (books) in a multi-dimensional space. Apply this to individual customers to predict their preferences.

Organisation of merchandise in a supermarket

Find clusters of products that are bought together. Ensure you don't discount both at the same time.

Associating genome sequences

Identify groups of genomes, cross-correlate, look for structure

Identifying plant diseases

Train machines using the strategy of human experts - maybe they will outperform their teachers in the end (it happened for soy plants).

Examples outside of astronomy:

Amazon recommends .....

Use the buying patterns of all customers to create characteristic directions (books) in a multi-dimensional space. Apply this to individual customers to predict their preferences.

Organisation of merchandise in a supermarket

Find clusters of products that are bought together. Ensure you don't discount both at the same time.

Associating genome sequences

Identify groups of genomes, cross-correlate, look for structure

Identifying plant diseases

Train machines using the strategy of human experts - maybe they will outperform their teachers in the end (it happened for soy plants).

Self-driving cars

Deep learning used for visual identification

Examples inside astronomy:

Regression: Hubble's law, Tully-Fisher relation ++++

Clustering: Identifying asteroid classes from SDSS photometry

Hierarchical trees: Galaxy morphology classification, photometric redshifts

MCMC & friends: Most fields.

Density estimators: Most fields.

Support Vector Machines: Stellar classification

Gaussian process regression: Time-series, galaxy formation

Random trees: Photometric redshifts for galaxies

Principal Component Analysis: PSF estimation, post-starbursts, data reduction improvements +++

Self-organising maps: Photometric redshifts, Υ-ray source classification.

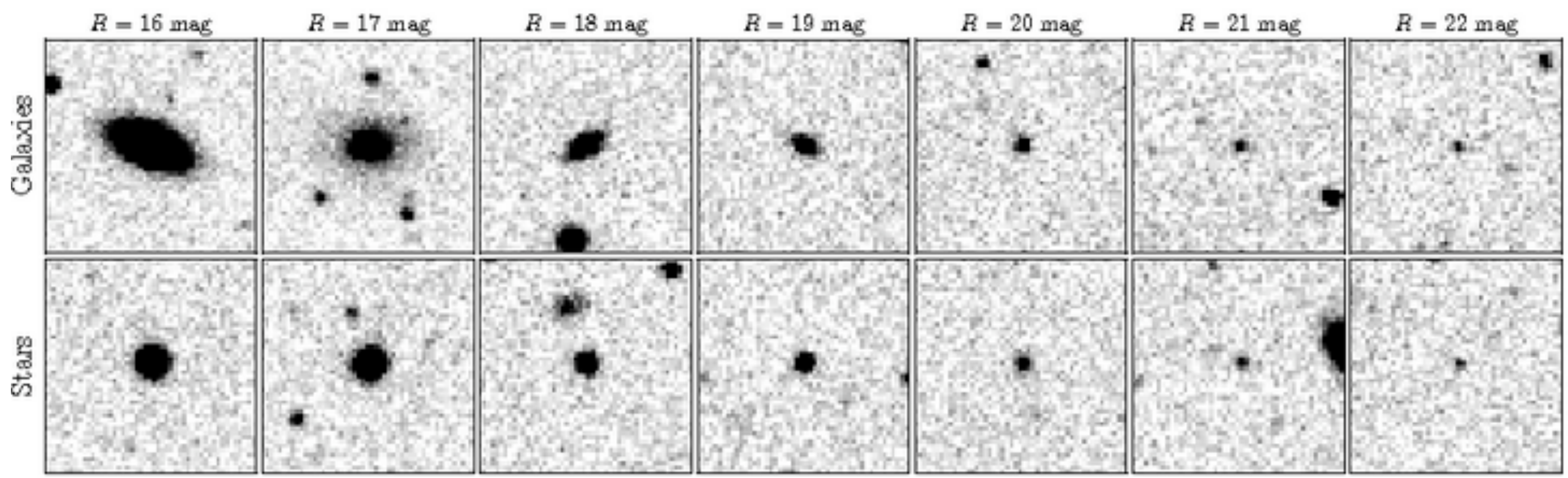
Neural nets: Photometric redshifts, stellar classification, galaxy morphology

Deep neural nets: galaxy morphology, image generation, gravitational lens finders, classification of lightcurves

What is machine learning (in this course)?

Thus machine learning can be a useful **tool** for astronomy and allows us to ask questions like:

- Is this blob of photons a galaxy or a star?

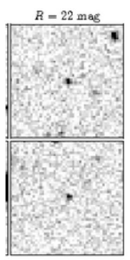


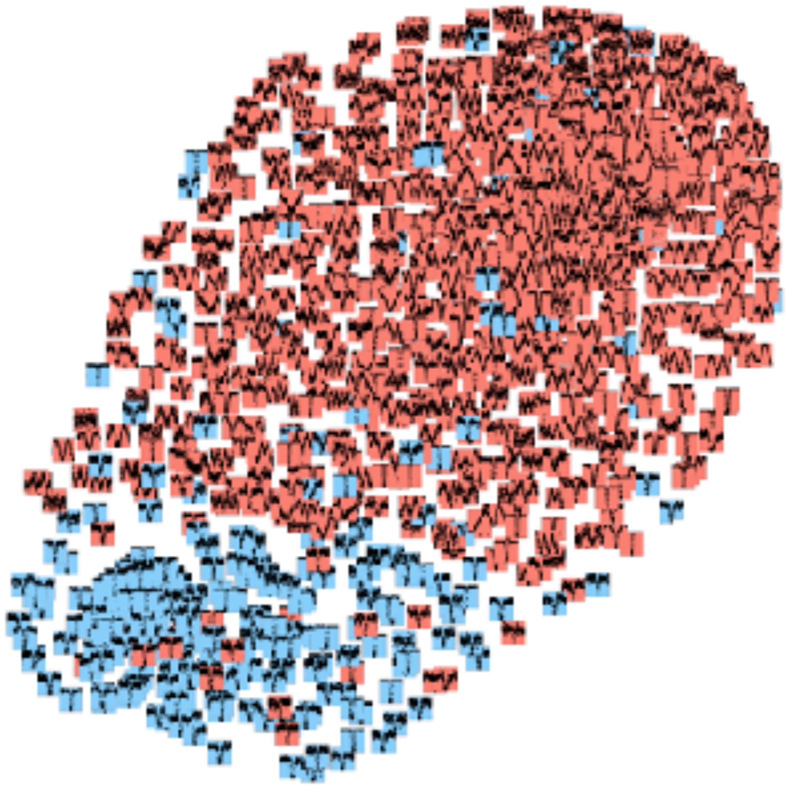
Miller et al (2017, arXiv:1703.07356)

What is machine learning (in this course)?

Thus machine learning can be a useful **tool** for astronomy and allows us to ask questions like:

- Is this blob of photons a galaxy or a star?
- Identify possible transiting exo-planets in Kepler light-curves

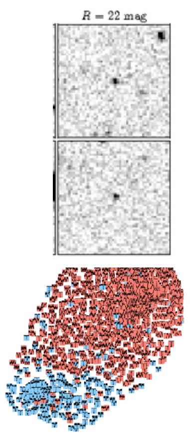




Shallue & Vanderburg (2018, AJ, 155)

What is machine learning (in this course)?

Thus machine learning can be a useful **tool** for astronomy and allows us to ask questions like:



- Is this blob of photons a galaxy or a star?
- Identify possible transiting exo-planets in Kepler light-curves
- What is the relationship between rotation velocity and luminosity in galaxies?
- How many classes of asteroids are there?
- etc etc.

Prediction vs discovery

There is often a distinction made between machine learning and data mining: machine learning aims to predict Y based on X in some “known” way, whereas data mining aims to discover unknown relationships within the data.

I will not make that distinction - both tasks are important in astronomy. And the methods/algorithms adopted are much the same in the two branches anyway.

What does it involve?

Data preparation

e.g.: Extract measurements from data
Check for quality/accuracy

What does it involve?

Data preparation

e.g.: Extract measurements from data
Check for quality/accuracy

Apply a machine learning technique

What does it involve?

Data preparation

e.g.: Extract measurements from data
Check for quality/accuracy

Sometimes optional

Inject data into a database

Create the database structure.

Read data using X and insert into database using SQL.

Get data from database

Apply a machine learning technique

The main methods

- Classification
  - Given a set of images, say, that we know the morphology of, we want to be able to classify a large set of other objects. Other examples: Classification of stellar spectra with GAIA, classification of transient sources, variable source detection etc.
- Estimation/prediction
- Grouping/clustering
- Dimension reduction

The main methods

- Classification
- Estimation/prediction
  - You have X, Y, Z, ... and you want to predict a value A. An example would be if you are given a set of emission lines and you want to estimate the metal abundance in the gas. Most fitting methods fall into this category. Another area of interest is to estimate distributions from a few observations - the aim of kernel estimation techniques.
- Grouping/clustering
- Dimension reduction

The main methods

- Classification
- Estimation/prediction
- Grouping/clustering
    - This is used to find groups objects with similar properties. One example is to find different asteroid families from orbital information. Another might be to find galaxy clusters, or simply to find outliers/weirdoes.
- Dimension reduction

The main methods

- Classification
- Estimation/prediction
- Grouping/clustering
- Dimension reduction
  - It is much easier to handle & visualise low-dimensional data. Dimension reduction techniques allow you to go from multi-dimensional data, with 1000s of dimensions, to a manageable set of variables. This can be used for model fitting (e.g. principal components analysis) and for exploratory data visualisation.

Why is this important?

Big surveys HENCE DATABASES

The last decade has seen a trend towards big surveys and massive theoretical calculations.
This will continue to produce vast amounts of data.

Large amounts of data HENCE STATISTICS

Large amounts of data offers a lot of potential but can be hard to work with/explore. This means that we will need other tools to make optimal use of the new data.

Sharing of data and resources HENCE VO/GRID/CLOUD..

It is now essential to share your data to get the most out of them and to have the highest impact, and many surveys already plan how to do this. But distributed computing is already a thing - e.g. using Amazon's services for calculations.

This is all true for observational, experimental and theoretical data!

Some specific examples

- Data rates are increasing - astronomers today produce data at about ~few Tb/night integrated over all (optical) telescopes. But LSST should produce ~15 Tb *per night*! [3 sq deg each 10-15 seconds]. How do you store/organise the data?

  Pre-processing & data bases!

- These large surveys produce large number of objects. (SDSS & 2MASS all contain > 100 million objects with perhaps 1000 attributes each). How do you check that all data are ok?

  Robust analysis, system engineering

- How do you analyse these kinds of data? To find the closest match using naïve search on 10^8 objects requires 10^{16} operations so correlation functions can be hard to calculate - and how do you find new relationships & events?

  Data mining/statistical methods/
  clever algorithms

Some numbers...

#Galaxies: 10^5 galaxies per square degree when going to m_I=25.5

(for every two magnitudes deeper you detect an order of magnitude more objects)

#Galaxies: 20,000 deg^2 “extragalactic sky” ⇒ ~10^9 sources

#Stars: $10^{5\text{-}6}$ stars per square degree per magnitude at low galactic latitudes

#Imaging data:

$$N_{\rm pix} = 1.44 \times 10^8 \left(\frac{\theta}{0.3''}\right)^2 \left(\frac{A}{1{\rm deg}^2}\right) \ {\rm pixels}$$

CFHT MegaCam: 36 CCDs with 2048 x 4612 pixels (340 megapixels)
~720 Mb per file for ~ 18000 pixels on the side (expanding to 1.4 Gb when converted to float...)

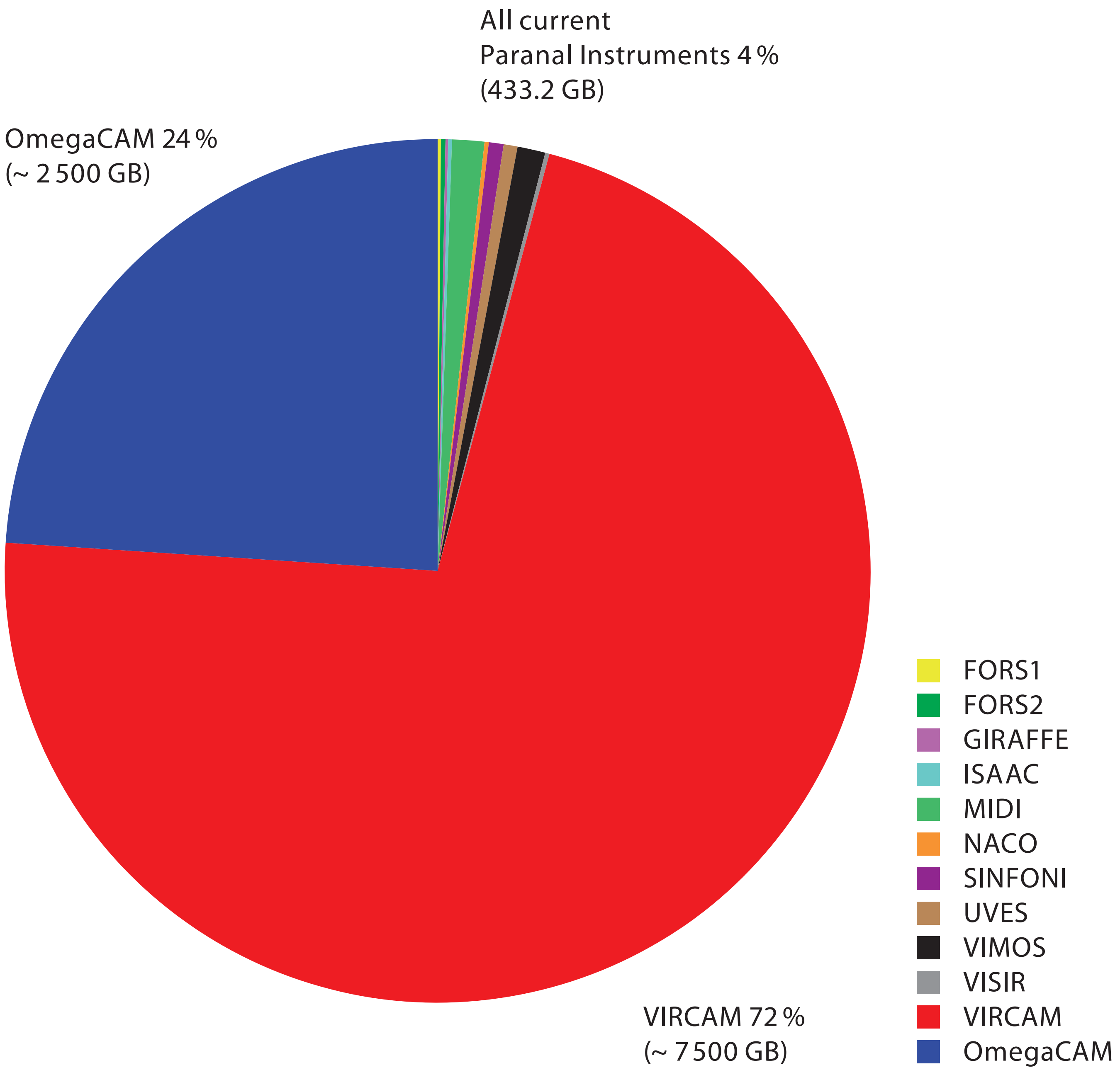
That is for one filter and one exposure - we normally need 4-5 filters and multiple exposures.

#Gaia: ~10^{21}-10^{22} Flops needed, future missions even more

Recall:
#seconds per year ~ 3×10^7
(top computers ~ 10^{15} Flops)

The ESO pie - or the changing world

The way the science changed over the last decade - and will continue to do so in the foreseeable future.



Arnaboldi et al (2007)

Key ingredients of machine learning

Examples:

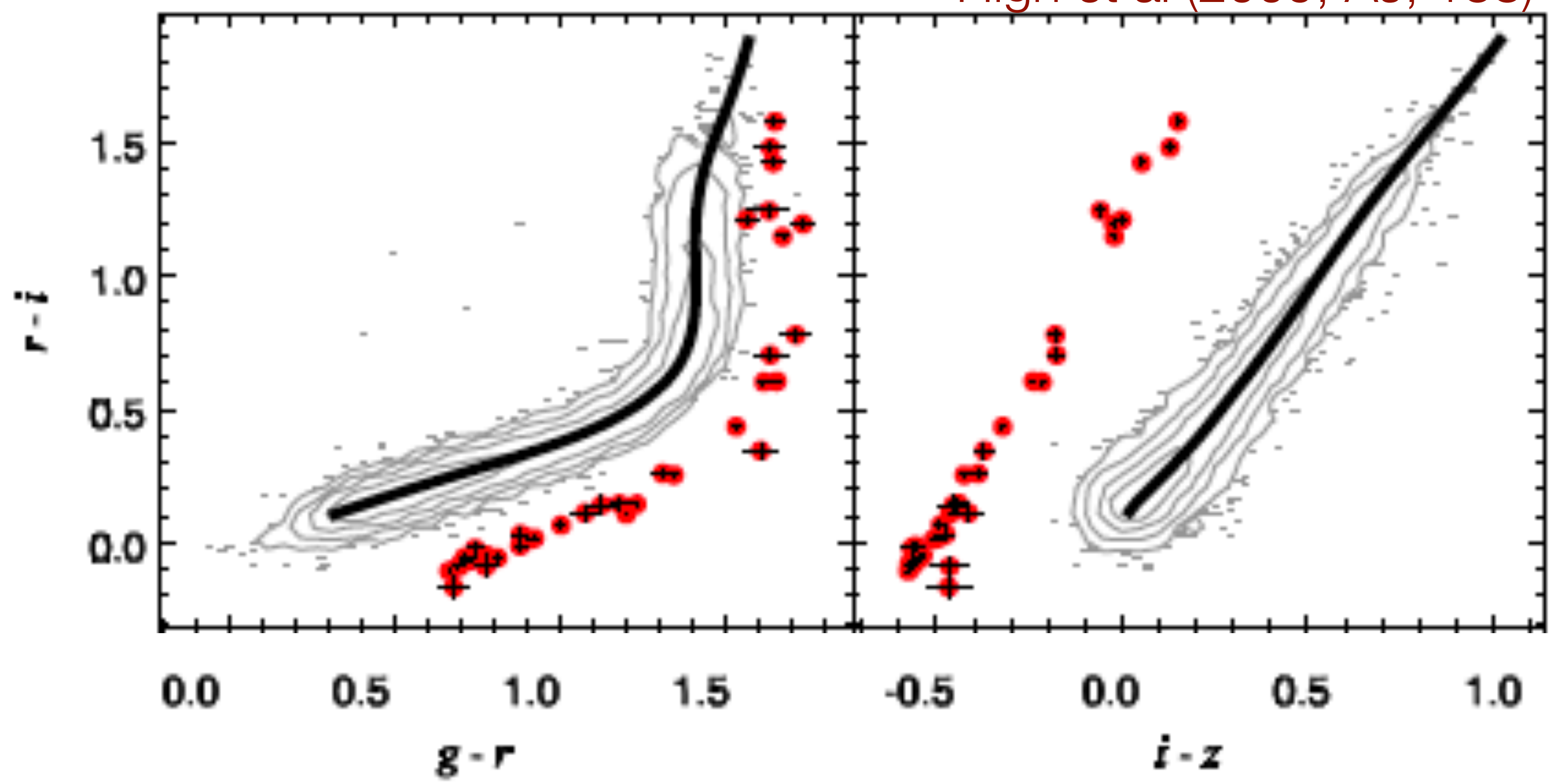
A machine learning method cannot “learn” unless it is provided with examples. These are composed of **features**.

Key ingredients of machine learning

Examples:

A machine learning method cannot “learn” unless it is provided with examples. These are composed of **features**.

High et al (2009, AJ, 138)

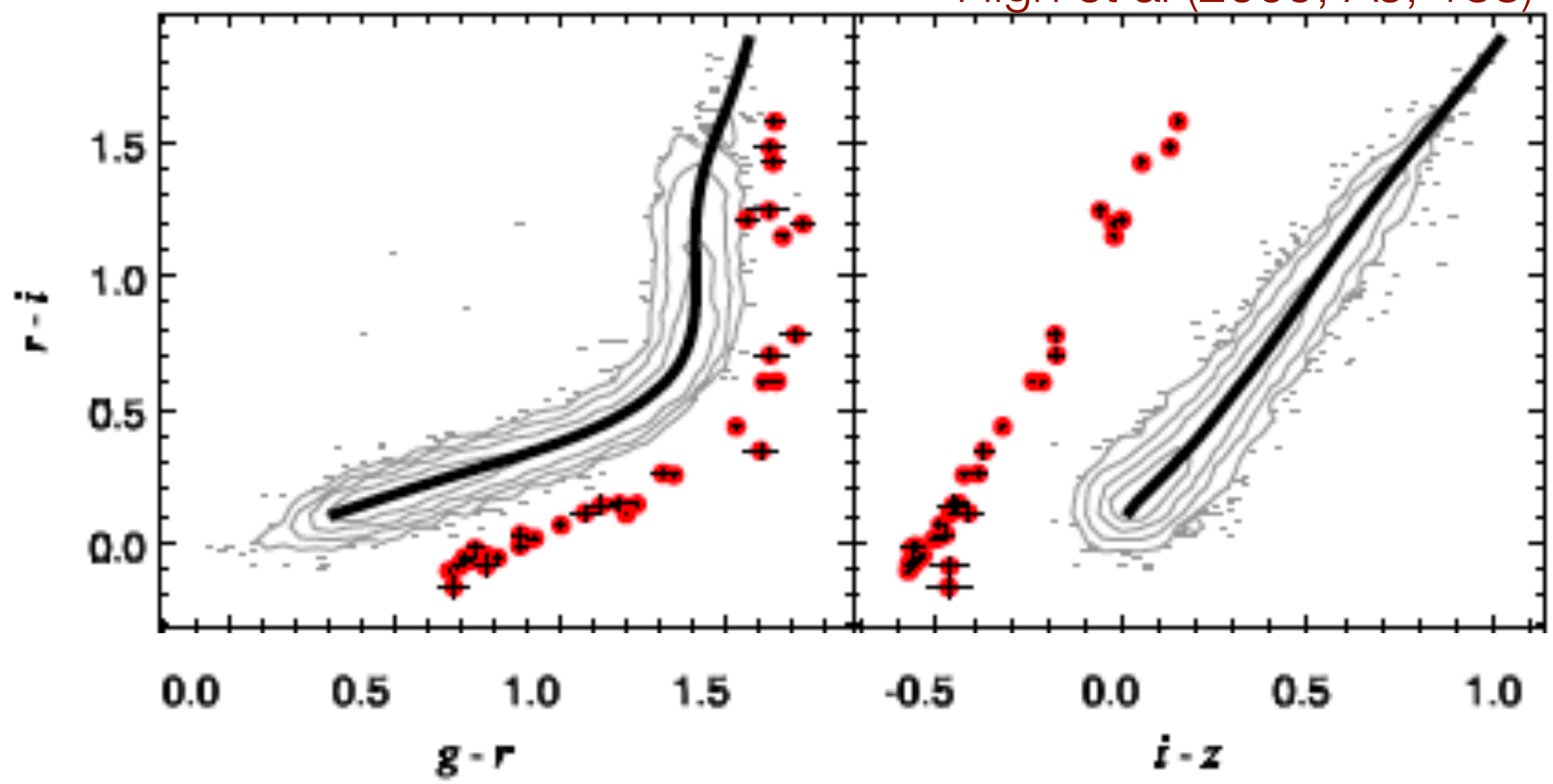


Key ingredients of machine learning

Examples:

A machine learning method cannot “learn” unless it is provided with examples. These are composed of **features**.

High et al (2009, AJ, 138)



Features: g-r, r-i, i-z

Example: [g-r, r-i, i-z]

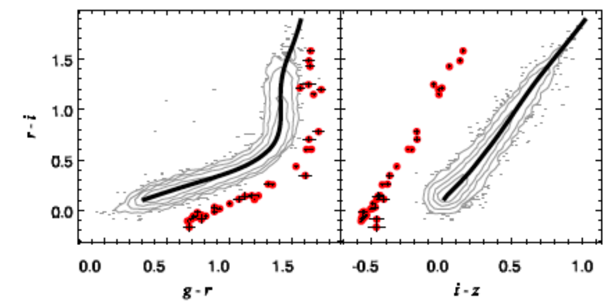
Key ingredients of machine learning

Examples:

A machine learning method cannot “learn” unless it is provided with examples. These are composed of **features**.

High et al (2009, AJ, 138)

Features: g-r, r-i, i-z
Example: [g-r, r-i, i-z]



Often summarised in a **design matrix**. Here each example makes up one row - e.g. for 4 points:

$$\begin{pmatrix} g_1 - r_1 & r_1 - i_1 & i_1 - z_1 \\ g_2 - r_2 & r_2 - i_2 & i_2 - z_2 \\ g_3 - r_3 & r_3 - i_3 & i_3 - z_3 \\ g_4 - r_4 & r_4 - i_4 & i_4 - z_4 \end{pmatrix}$$

Key ingredients of machine learning

Accuracy/error-rate

How well is our algorithm doing? A common measure - the mean squared error:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y - y_{\mathrm{pred}}\right)^2$$

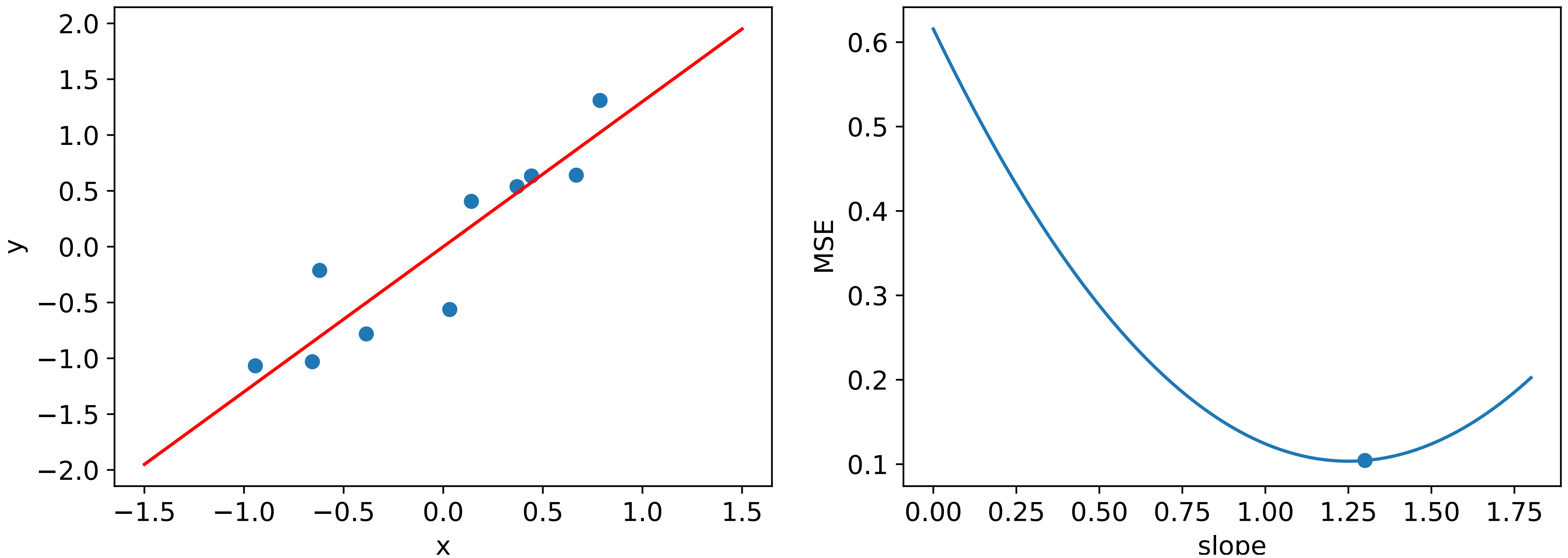
Key ingredients of machine learning

Accuracy/error-rate

How well is our algorithm doing? A common measure - the mean squared error:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y - y_{\mathrm{pred}})^2$$

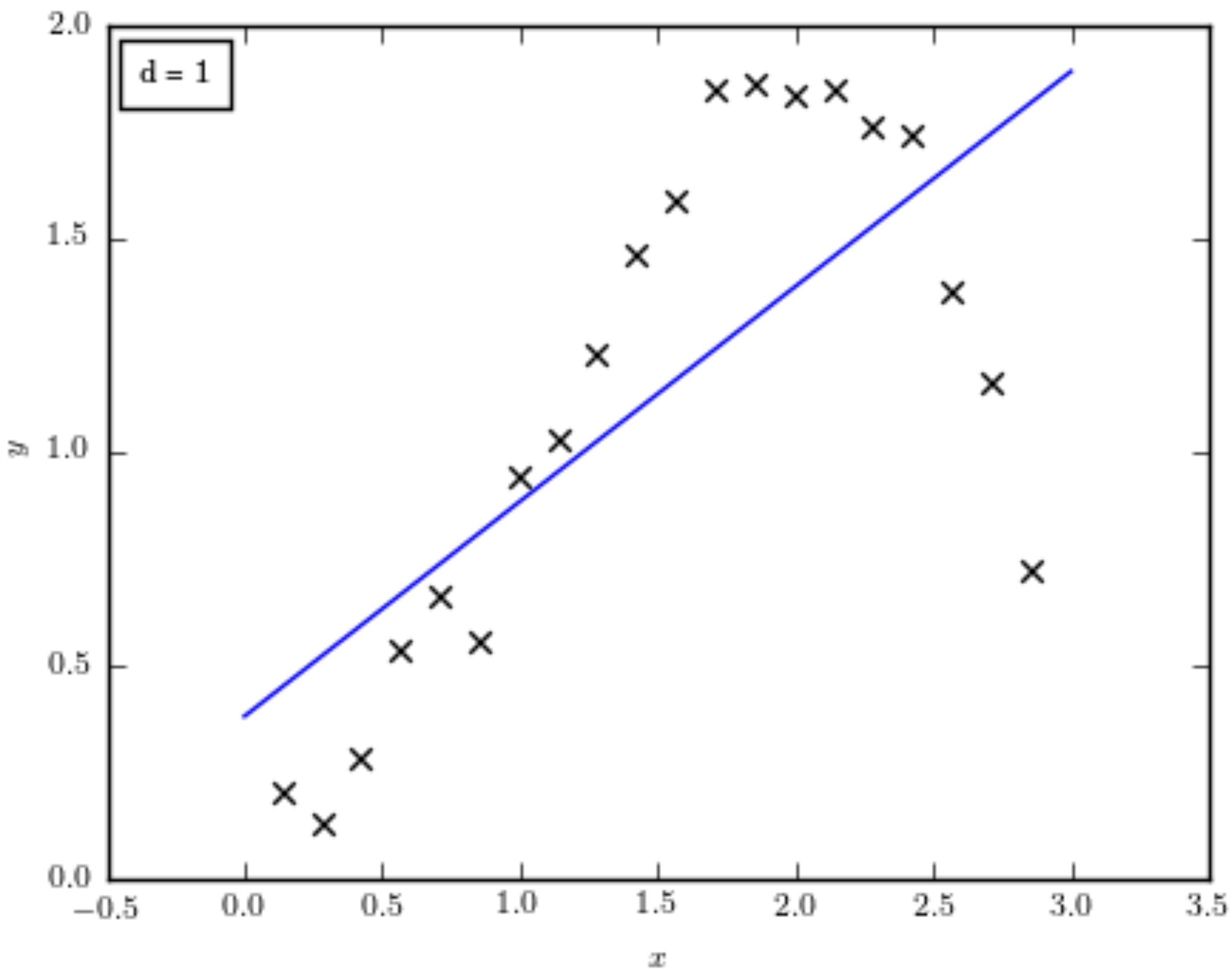
Try this on linear regression $y = \mathrm{slope} \times x$



A simple example - fitting a polynomial

Overfitting, underfitting

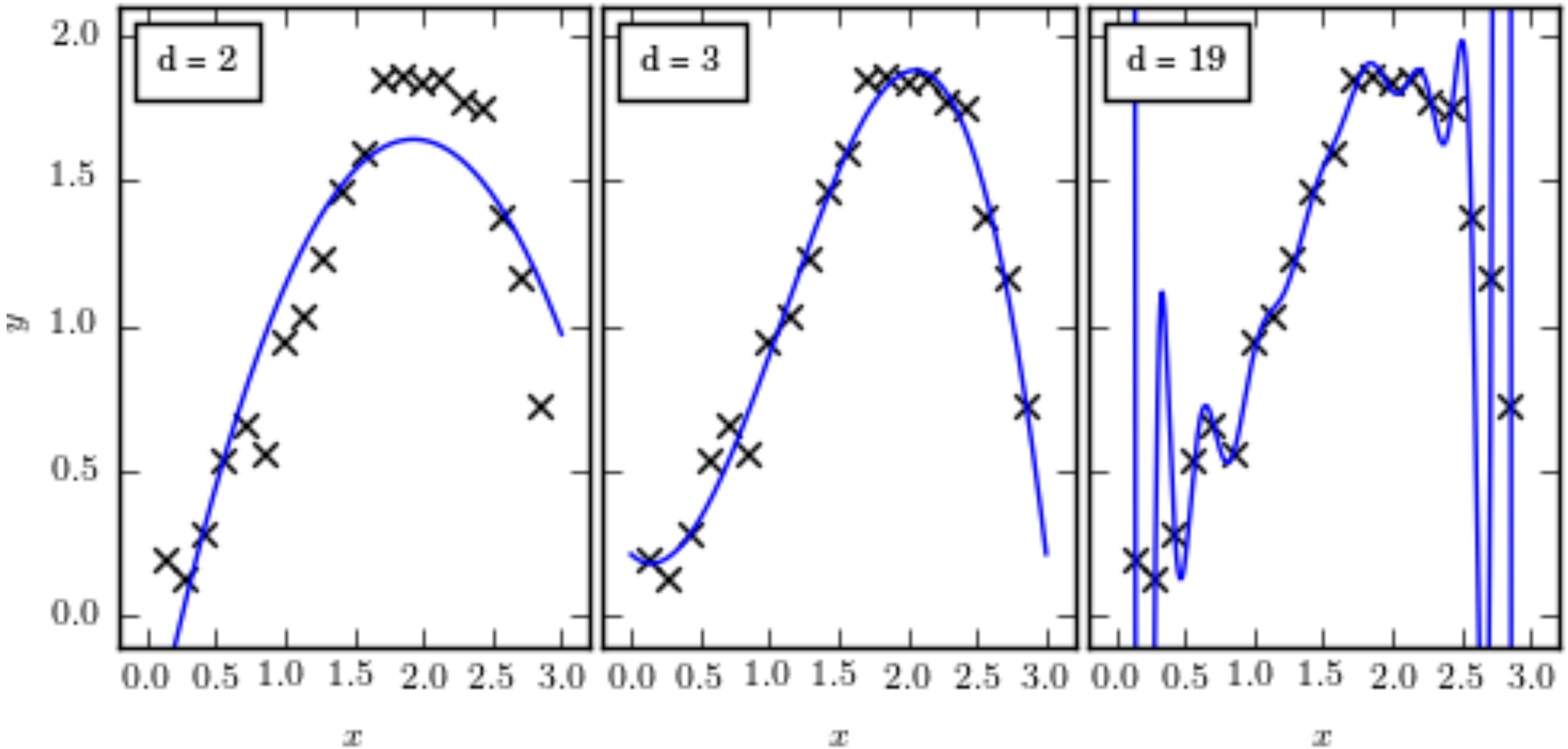
We want our method to **generalise** well



High bias!

Increasing the flexibility:

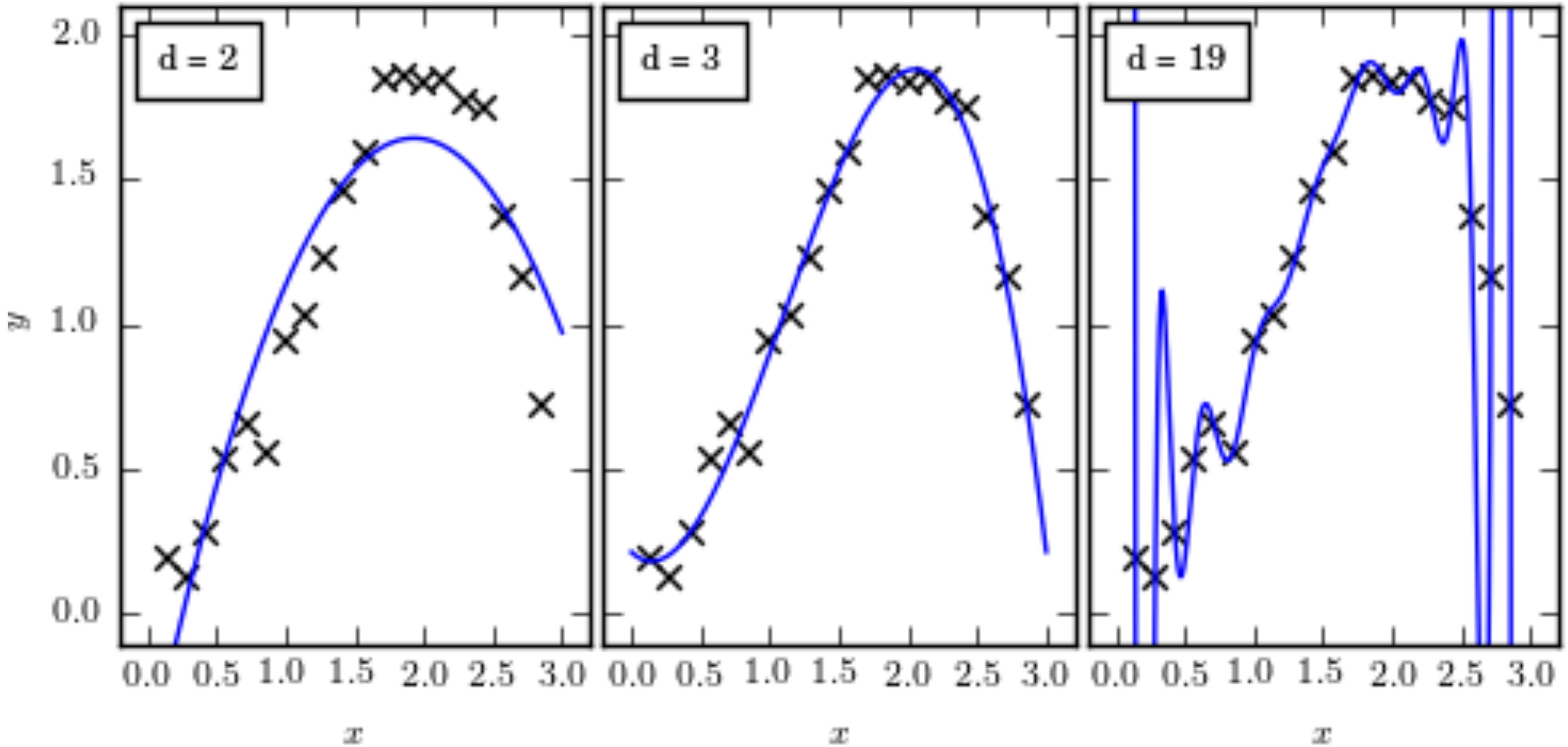
From astroML.org



Our eye would probably say a 3rd degree polynomial is ok.

Increasing the flexibility:

From astroML.org



Underfitting

Overfitting

Key ingredients of machine learning

Parameters, hyper-parameters

Consider a polynomial:

$$y = \sum_{i=0}^{M} a_i x^i$$

Here a_i are the parameters of the model and what our machine learning algorithm will give us.

But M - the maximum polynomial power is *also* a parameter. It is a different kind of parameter and we call this a **hyperparameter**.

How do you decide

We will return to this but we should think of dividing our data in three:

The training set

This is what we use to find the best-fitting parameters and we minimise the **training error** (e.g. MSE) on this.

The test set

We evaluate the generalisation error of our algorithm on this sample. These data are *not* used for the fitting. We refer to this as **test error**.

The validation set

We use this to determine the optimal settings of the hyper-parameters.

Plan for the course

- 1. Managing data and simple regression.
  - Covering git and SQL
  - Introducing machine learning through regression techniques.
- 2. Visualisation and inference methods
  - Visualisation of data, do's and don't's
  - Classical inference
  - Bayesian inference
  - MCMC

Practical class

- 3. Density estimation and model choice
  - Estimating densities, parametric & non-parametric
  - Bias-variance trade-off
  - Cross-validation
  - Classification

Practical class

- 4. Dimensional reduction
  - Standardising data.
  - Principal Component Analysis
  - Manifold learning

Practical class

- 5. Ensemble methods, neural networks, deep learning
  - Local regression methods
  - Random forests and other boosting methods
  - Neural networks & deep learning

Practical class

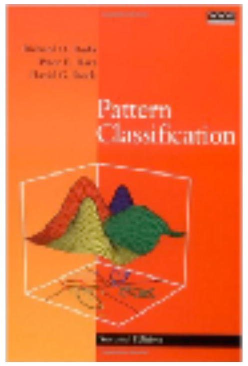
Literature

No obligatory book, but I will roughly follow:

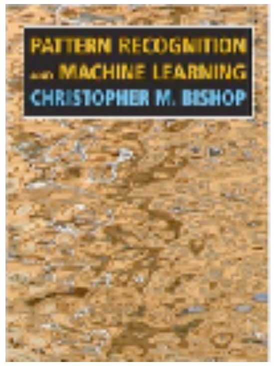
Statistics, Data Mining, and Machine Learning in Astronomy - *Ivezic, Connolly, VanderPlas & Gray*

with some inspiration from:
Deep Learning - *Goodfellow, Bengio & Courville*

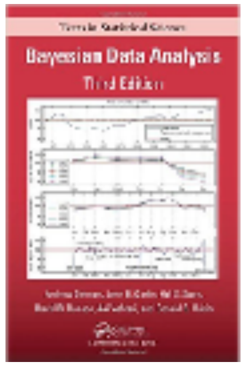
Other useful books among many others:



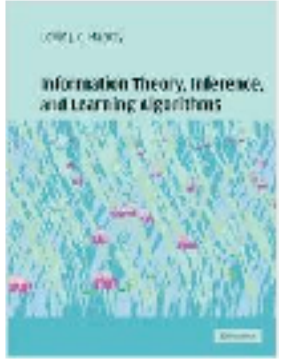
Pattern Classification - *Duda, Hart & Stork*



Pattern Recognition and Machine Learning - *Bishop*



Bayesian Data Analysis - *Gelman*

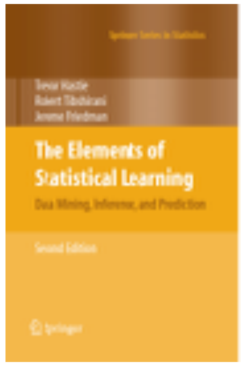


Information Theory, Inference and Learning Algorithms - *MacKay*

Online

Introduction to Statistical Learning - *James et al*

Online



Elements of Statistical Learning - *Hastie et al*

Online

Language choice

In this course we will use **Python** and in particular the scikit-learn (sklearn) toolkit for machine learning. (and **SQL**)

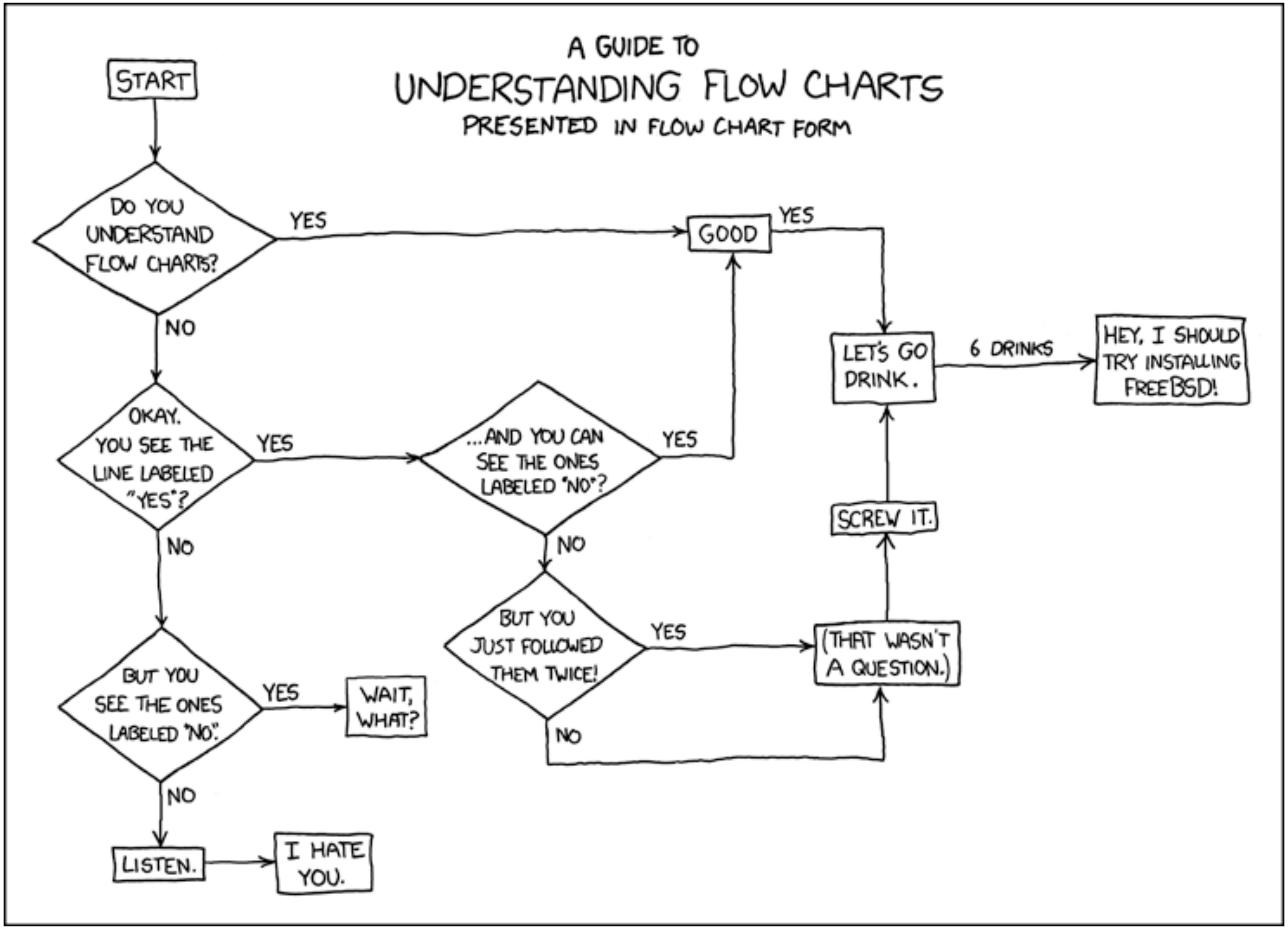
Other languages are widely used in machine learning and you could easily come across:

- **R**. The statistical computing language is very widely used and many new algorithms are implemented in R first. Well worth learning.
- **C/C++**. Often required for speed. Many frameworks are written in these languages (e.g. Torch, TensorFlow)
- **Java**. Influential deep learning frameworks like Deeplearning4j are coded in Java and it is a popular deployment platform.
- **Matlab**. A traditional choice for algorithm development thus you might come across Matlab code that does exactly what you want.

Practicalities - working in data science

Good enough practices in scientific computing

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510



https://xkcd.com/518/

Good enough practices in scientific computing

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

The topics of their paper:

- Data management: saving both raw and intermediate forms, documenting all steps, creating tidy data amenable to analysis.
- Software: writing, organizing, and sharing scripts and programs used in an analysis.
- Collaboration: making it easy for existing and new collaborators to understand and contribute to a project.
- Project organization: organizing the digital artifacts of a project to ease discovery and understanding.
- Tracking changes: recording how various components of your project change over time.
- Manuscripts: writing manuscripts in a way that leaves an audit trail and minimizes manual merging of conflicts

Data management

- ✓ Save the raw data.
- ✓ Ensure that raw data are backed up in more than one location.
- ✓ Create the data you wish to see in the world.
- ✓ Create analysis-friendly data.
- ✓ Record all the steps used to process data.
- ✓ Anticipate the need to use multiple tables, and use a unique identifier for every record.
- ✓ Submit data to a reputable DOI-issuing repository so that others can access and cite it.

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Data management

- ✓ Save the raw data.
- ✓ Ensure that raw data are backed up in more than one location.
- ✓ Create the data you wish to see in the world.
- ✓ Create analysis-friendly data.
- ✓ Record all the steps used to process data.
- ✓ Anticipate the need to use multiple tables, and use a unique identifier for every record.
- ✓ Submit data to a reputable DOI-issuing repository so that others can access and cite it.

Backup & storage

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Data management

- ✓ Save the raw data.
- ✓ Ensure that raw data are backed up in more than one location.
- ✓ Create the data you wish to see in the world.
- ✓ Create analysis-friendly data.
- ✓ Record all the steps used to process data.
- ✓ Anticipate the need to use multiple tables, and use a unique identifier for every record.
- ✓ Submit data to a reputable DOI-issuing repository so that others can access and cite it.

Backup & storage

Data reduction,
Data organisation

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Data management

- ✓ Save the raw data.
- ✓ Ensure that raw data are backed up in more than one location.

Backup & storage

- ✓ Create the data you wish to see in the world.
- ✓ Create analysis-friendly data.
- ✓ Record all the steps used to process data.

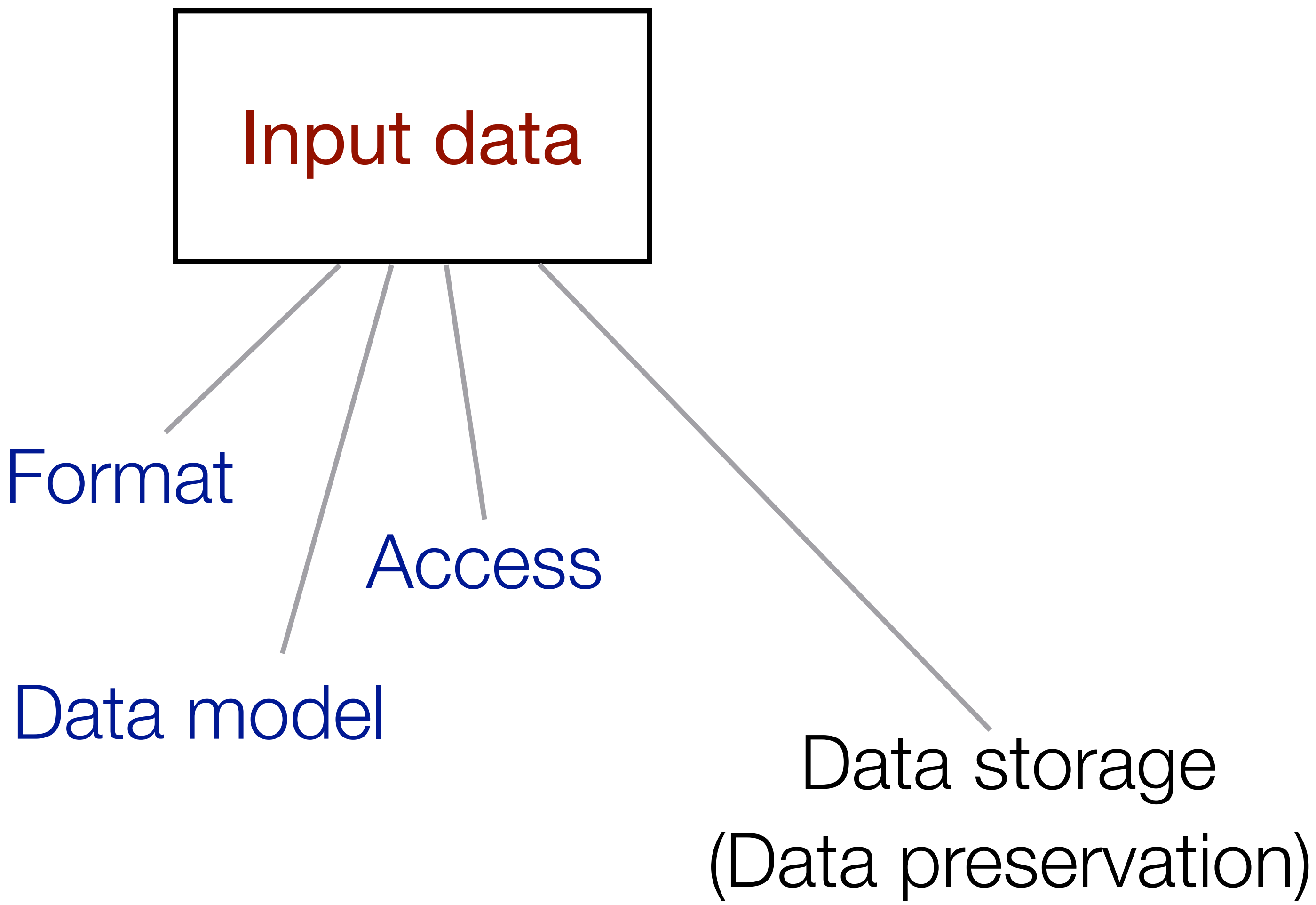
Data reduction,
Data organisation

- ✓ Anticipate the need to use multiple tables, and use a unique identifier for every record.
- ✓ Submit data to a reputable DOI-issuing repository so that others can access and cite it.

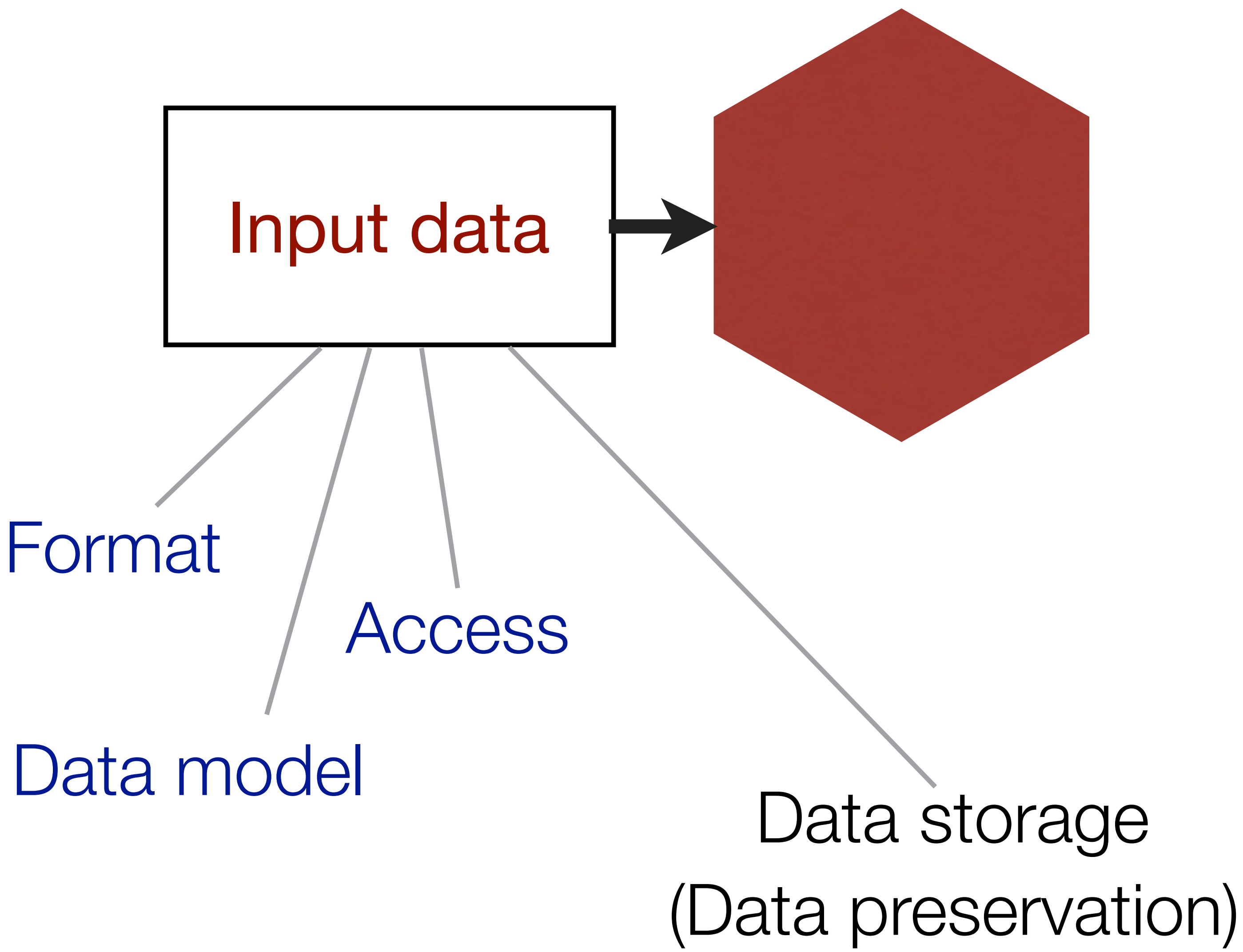
Think about your data in advance and plan for sharing with the world.

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

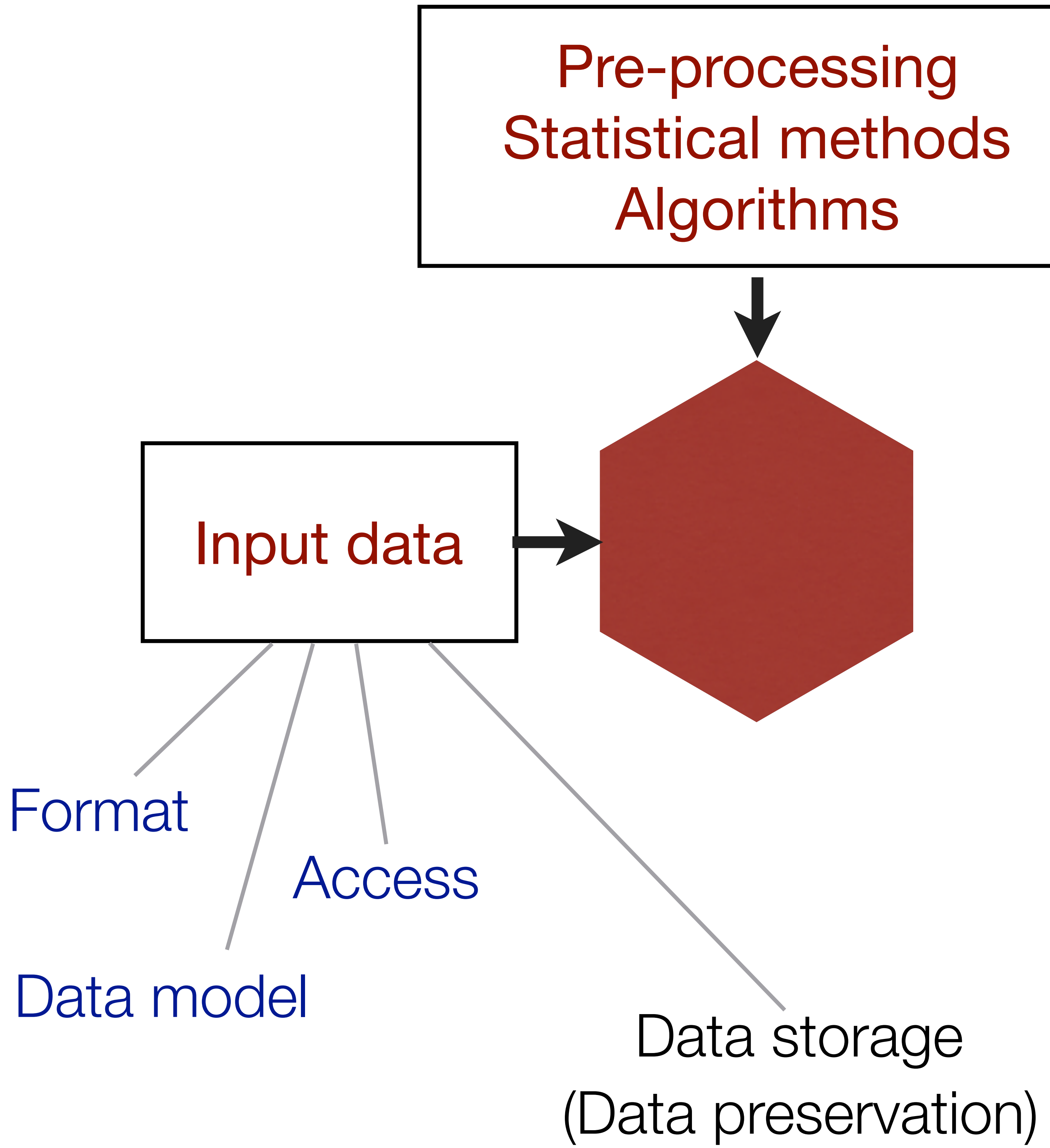
Data Mining/Machine learning Outline



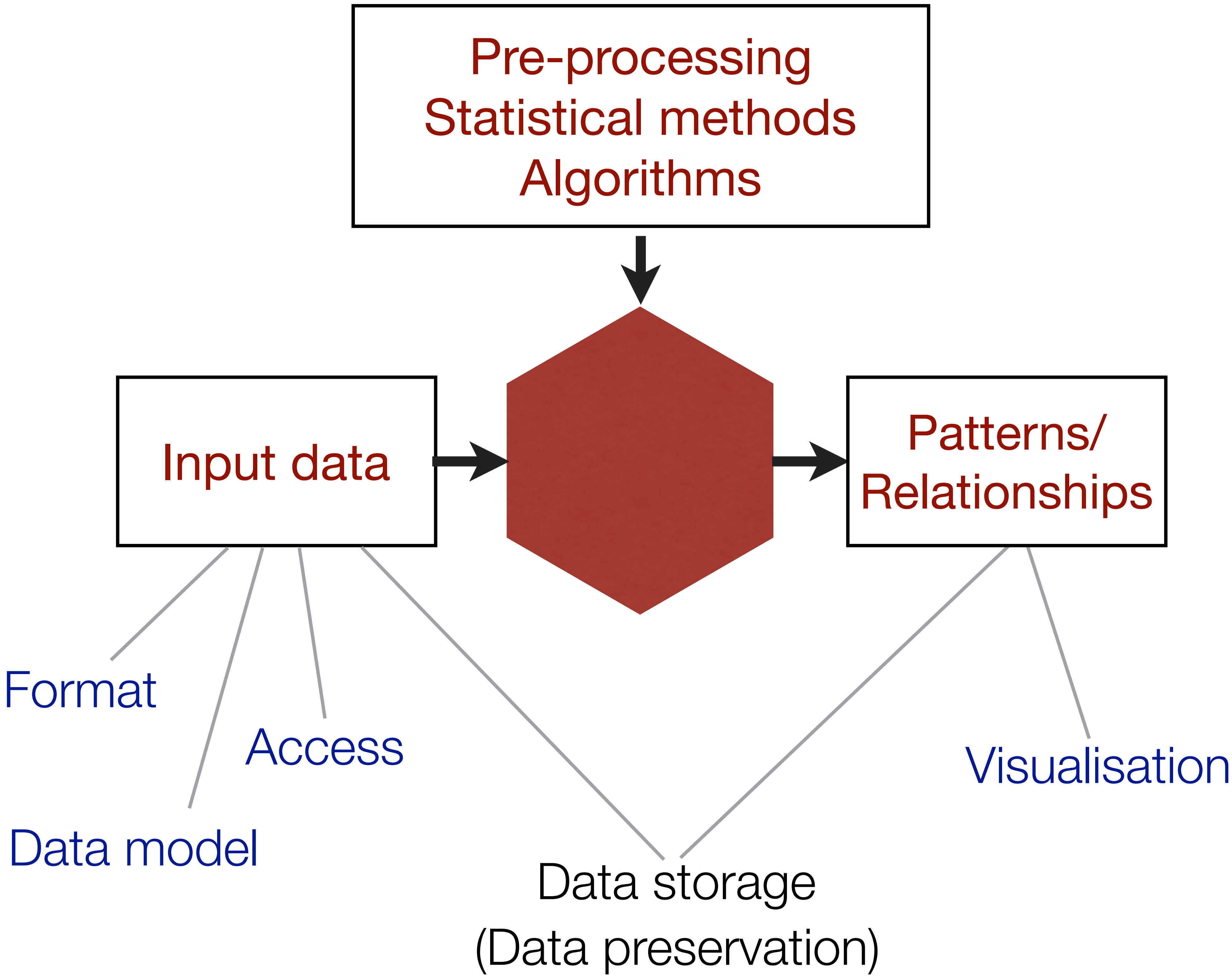
Data Mining/Machine learning Outline



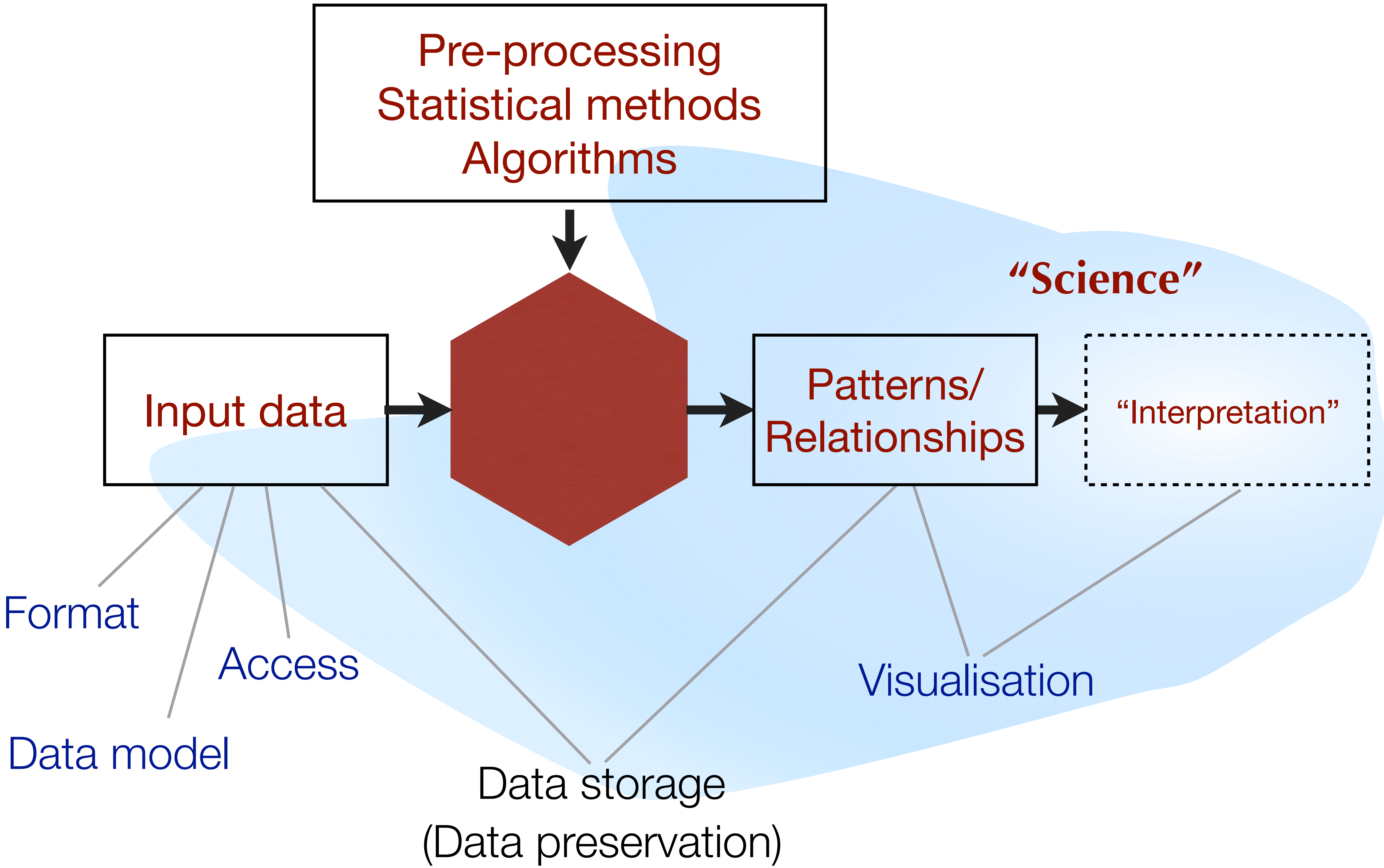
Data Mining/Machine learning Outline



Data Mining/Machine learning Outline



Data Mining/Machine learning Outline



Software

- ✓ Place a brief explanatory comment at the start of every program.
- ✓ Decompose programs into functions.
- ✓ Be ruthless about eliminating duplication.
- ✓ Always search for well-maintained software libraries that do what you need.
- ✓ Test libraries before relying on them.
- ✓ Give functions and variables meaningful names.
- ✓ Make dependencies and requirements explicit.
- ✓ Do not comment and uncomment sections of code to control a program's behaviour.
- ✓ Provide a simple example or test data set.
- ✓ Submit code to a reputable DOI-issuing repository.

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Collaboration

- ✓Create an overview of your project.
- ✓Create a shared "to-do" list for the project.
- ✓Decide on communication strategies.
- ✓Make the license explicit.
- ✓Make the project citable.

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Collaboration

- ✓Create an overview of your project.
- ✓Create a shared "to-do" list for the project.
- ✓Decide on communication strategies.
- ✓Make the license explicit.
- ✓Make the project citable.

Plan your project

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Collaboration

- ✓Create an overview of your project.
- ✓Create a shared "to-do" list for the project.
- ✓Decide on communication strategies.
- ✓Make the license explicit.
- ✓Make the project citable.

Plan your project

Think about how you want to share information

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Project organisation

- ✓Put each project in its own directory, which is named after the project.
- ✓Put text documents associated with the project in the doc directory.
- ✓Put raw data and metadata in a data directory and files generated during cleanup and analysis in a results directory.
- ✓Put project source code in the src directory.
- ✓Put external scripts or compiled programs in the bin directory.
- ✓Name all files to reflect their content or function.

Probably the most “personal” of recommendations - a good idea to follow but not the only way to do it!

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Keeping track of changes

- ✓ Back up (almost) everything created by a human being as soon as it is created.
- ✓ Keep changes small.
- ✓ Share changes frequently.
- ✓ Create, maintain, and use a checklist for saving and sharing changes to the project.
- ✓ Store each project in a folder that is mirrored off the researcher's working machine.
- ✓ Add a file called CHANGELOG.txt to the project's docs subfolder.
- ✓ Copy the entire project whenever a significant change has been made.
- ✓ Use a version control system.

Prone to errors

← Use this!

Wilson et al (2017) - https://doi.org/10.1371/journal.pcbi.1005510

Version control using git

Managing your work - version control

When you carry out a complex task, you are advised to use a version control (VC) system. We will use **git**.

A VC helps keep track of changes made to your documents/code/whatever and allows you to branch out to try something new.

There are multiple possible ways to use git - local repositories and online repositories. Here we will use an online repository: github [github.com]

(there are many others, e.g. bitbucket.org)

Added benefit: important inside & outside academia!

But really - why should I bother?

- ✓Keeps track of your work.
- ✓The history allows you to check what you changed x days ago.
- ✓Adds structure to your work.
- ✓Simplifies collaboration on code.
- ✓Simplifies sharing of code with the world.
- ✓Keeps a backup of your code!
- ✓Allows you to implement new features while not breaking a released code (‘branching’)

A bird's eye view of how we use VCs

I need to implement a new function in my code

1. Check if there have been updates to the code (if I work with someone else) — git status, git pull

2. Implement all or parts of the new functionality. Save.

3. Add the changed file to a list of changes (git keeps track of what you changed) — git add

4. Commit the modification with a small note to say what you have done — git commit

5. Send it all away — git push

Making a local repository

Create a working directory where your project will live, and go into it

```
> mkdir Project
> cd Project
```

Initialise a git repository:

```
> git init
```

Put a file in the repository - this is the simplest way

```
> touch README
```

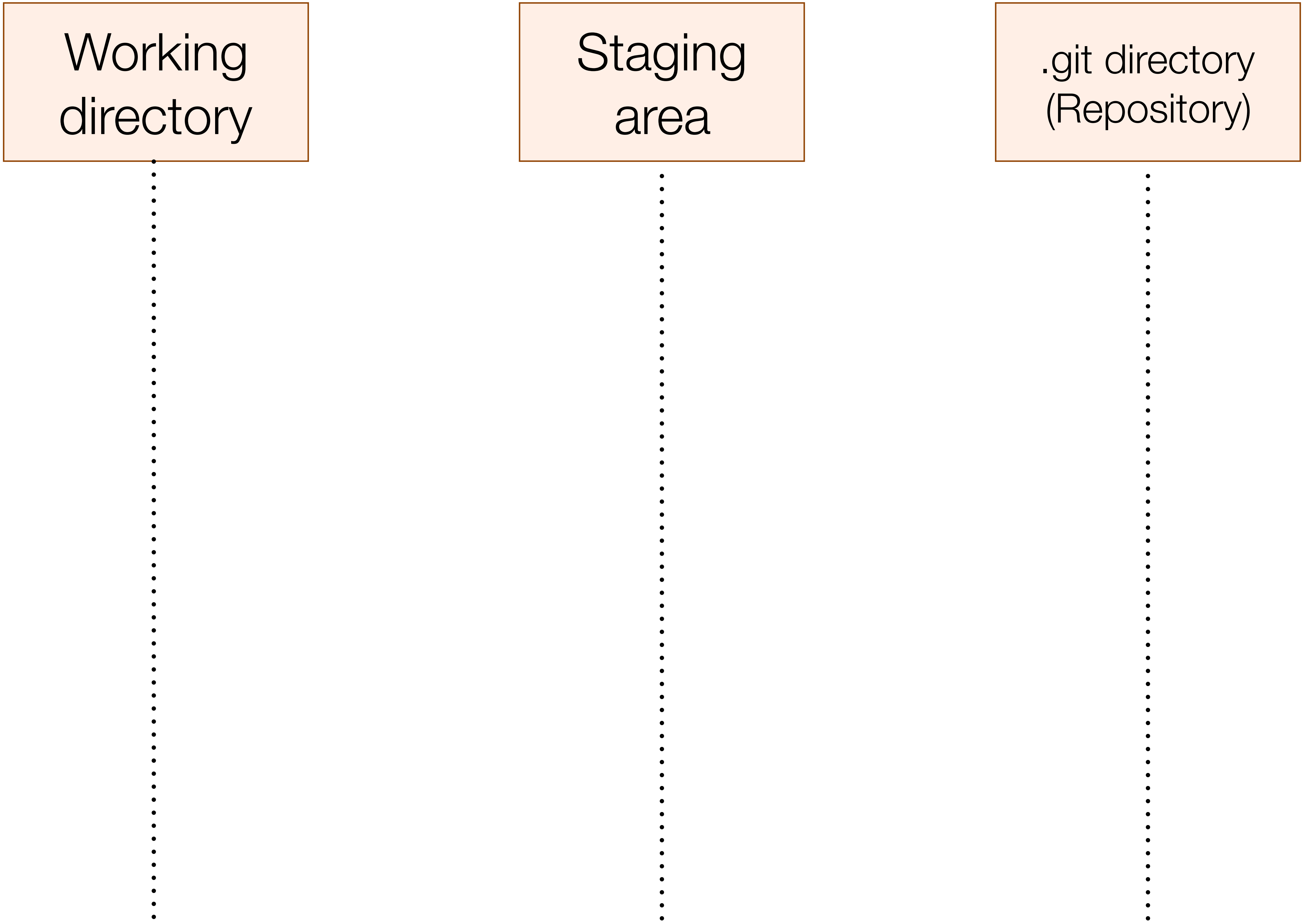
And check the status

```
> git status
```

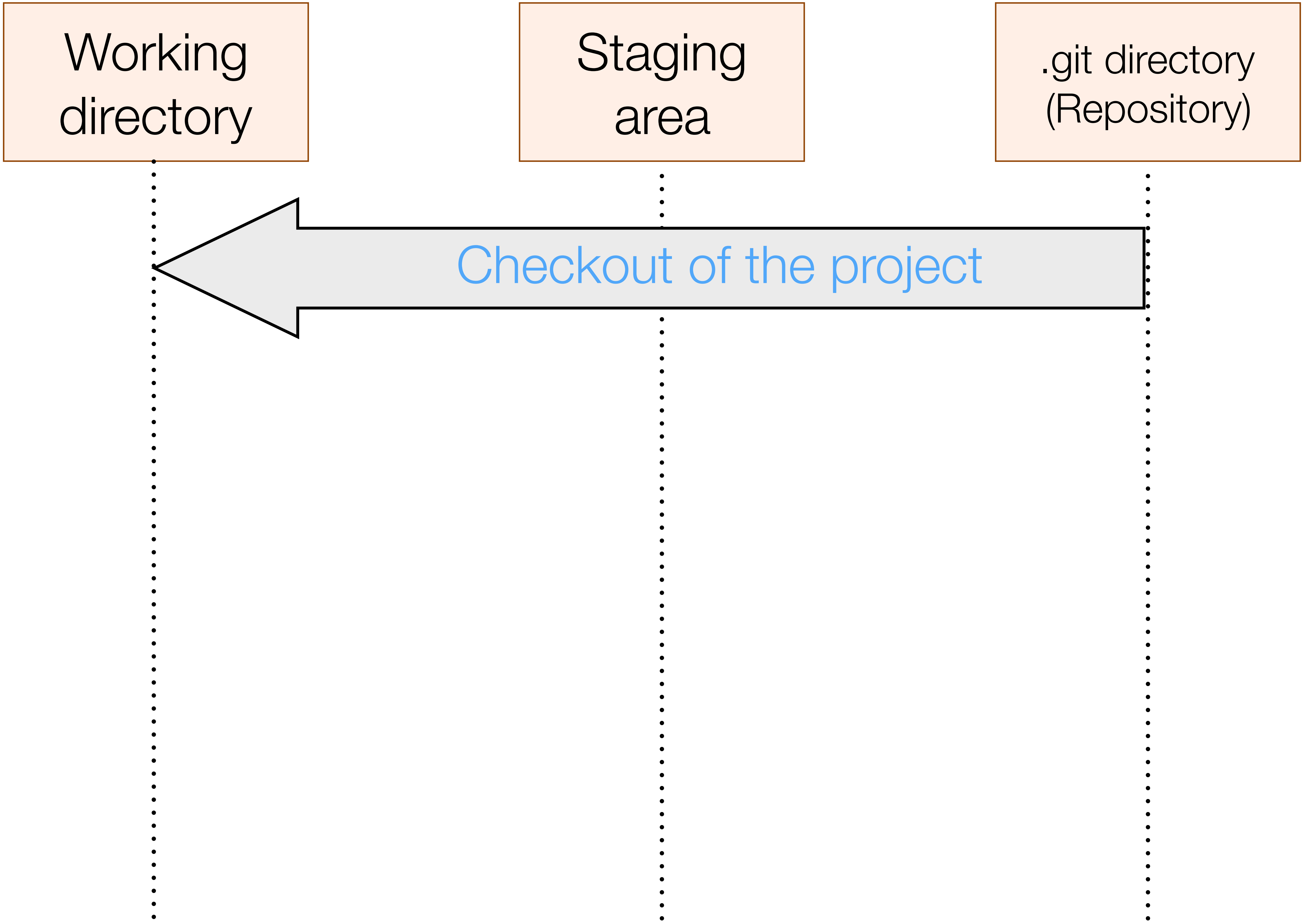
Making a local repository - still!

```
> git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README

nothing added to commit but untracked files present (use "git add" to track)
```

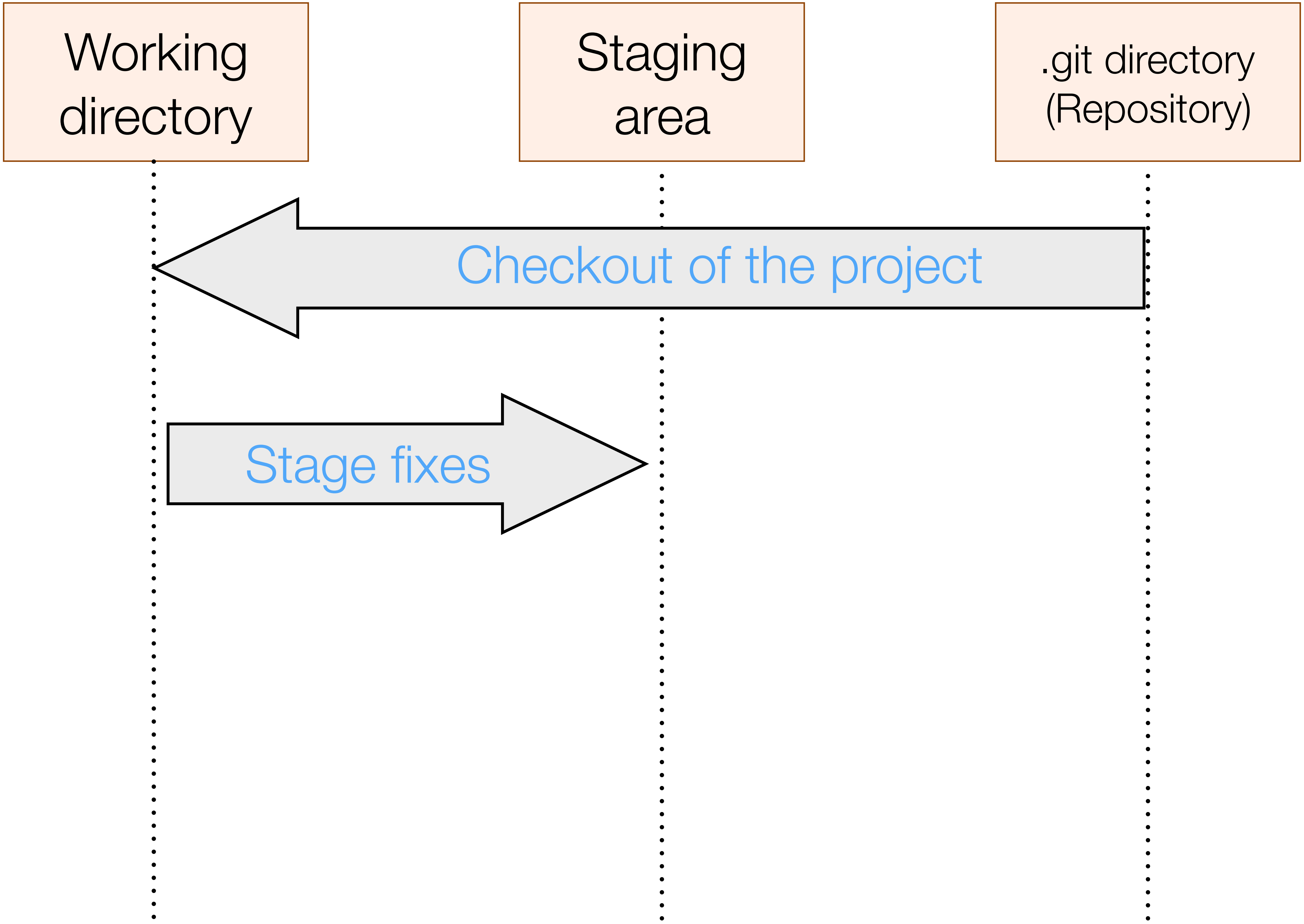
Staging, committing & pushing



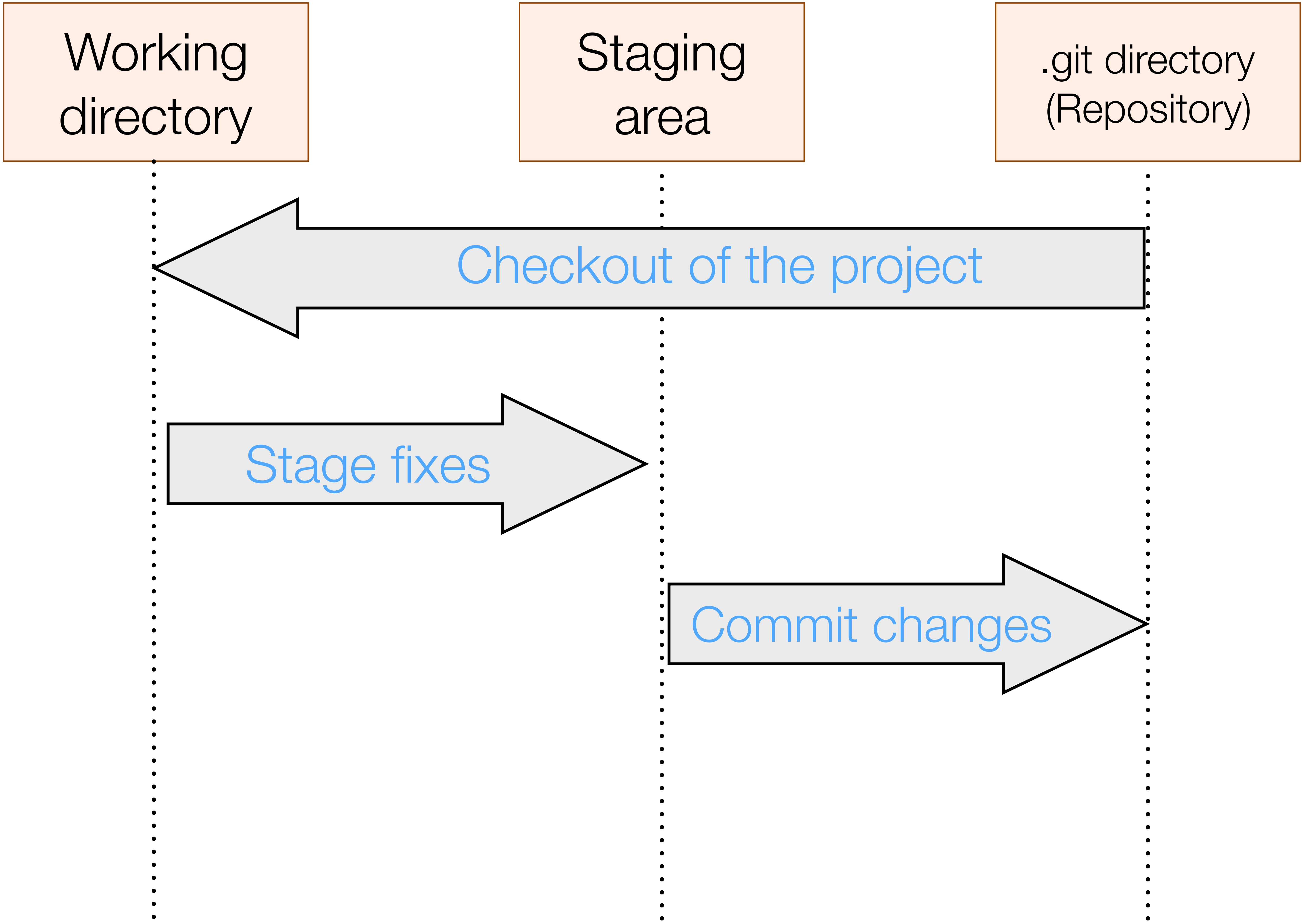
Staging, committing & pushing



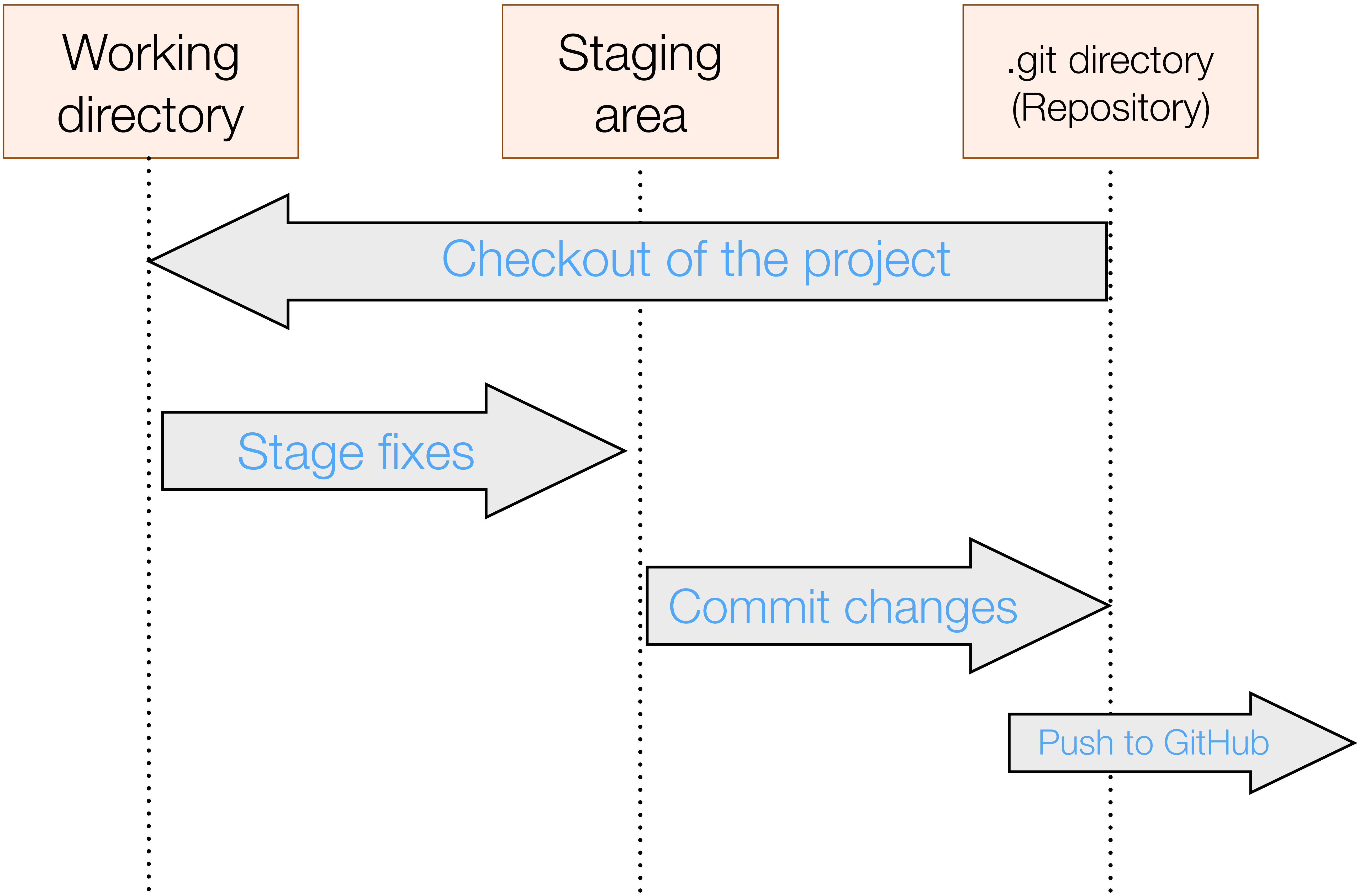
Staging, committing & pushing



Staging, committing & pushing



Staging, committing & pushing



Making a local repository - still!

Add the files

```
> git add README
```

Making a local repository - still!

Add the files

```
> git add README
```

Commit the changes

```
> git commit -m "First commit"
```

Making a local repository - still!

Add the files

```
> git add README
```

Comments! Important!

Commit the changes

```
> git commit -m "First commit"
```

Making a local repository - still!

Add the files

```
> git add README
```

Comments! Important!

Commit the changes

```
> git commit -m "First commit"
```

---

moving files

```
> git mv <oldfile> <newfile>
```

Oops, get the old version of the file

```
> git checkout --  filename
```

More: https://www.codementor.io/citizen428/git-tutorial-10-common-git-problems-and-how-to-fix-them-aajv0katd

Using git - checking out a project

You need to know the address of the project:

```
https://github.com/jbrinchmann/MyRepo1.git
```

Then you check it out:

```
git clone https://github.com/jbrinchmann/MyRepo1.git
```

You now have this repository in the MyRepo1 directory

To get a new version (if someone has changed something) - go into the MyRepo1 directory and type:

```
git pull
```

git - workflow with GitHub

Make GitHub repository	`https://github.com/jbrinchmann`
Check it out	`git clone <address>`
Edit a file on your computer — add the file if new	`git add square.py`
Commit your changes	`git commit -m “Fixed exponent bug”`
Push the changes to the repository	`git push`

Another reason for using it:

You need it for the course!

For you to do:

Check the repository at
https://github.com/jbrinchmann/MLD2019

Here you will find a small document with math/ statistics reminders which I expect you to have read!

I will also place some problems that we will work with in the practical session, on this site.

Databases

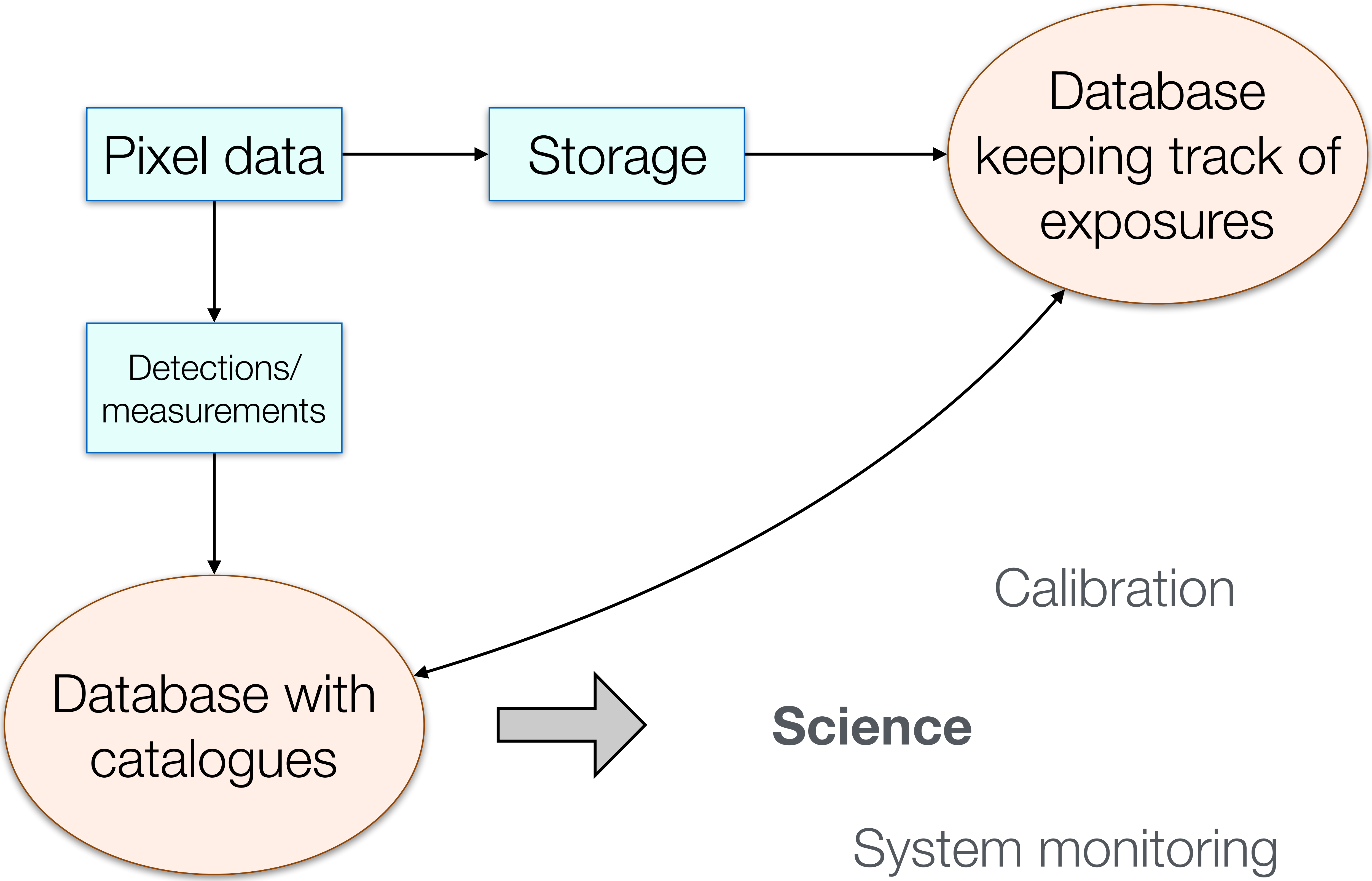
Sloan Digital Sky Survey - the reference database in astronomy

Millennium, EAGLE, Illustris - cosmological hydrodynamical simulations

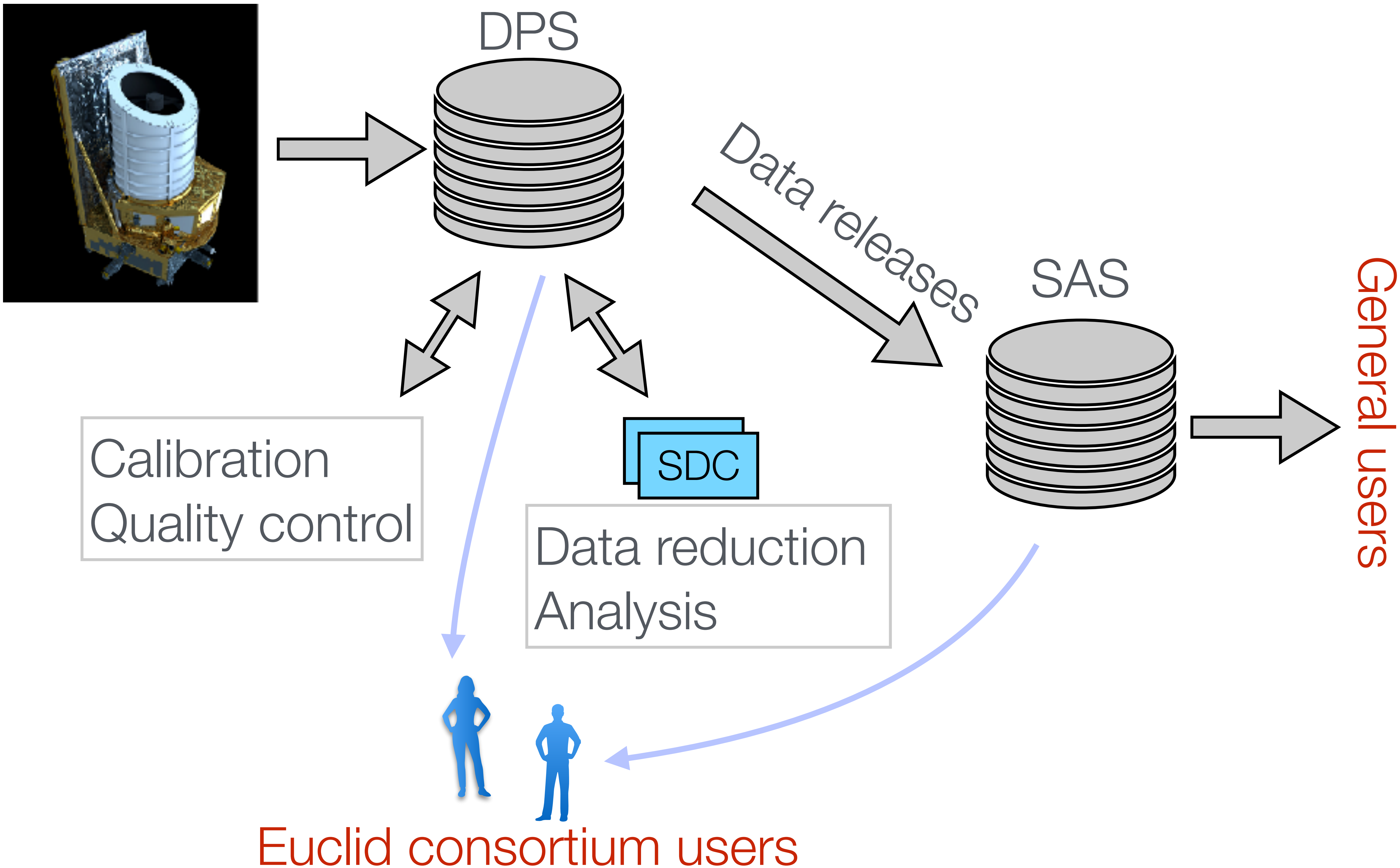
Kepler, TESS, GAIA

GALEX, 2MASS, etc etc.

The role of databases - a modern view

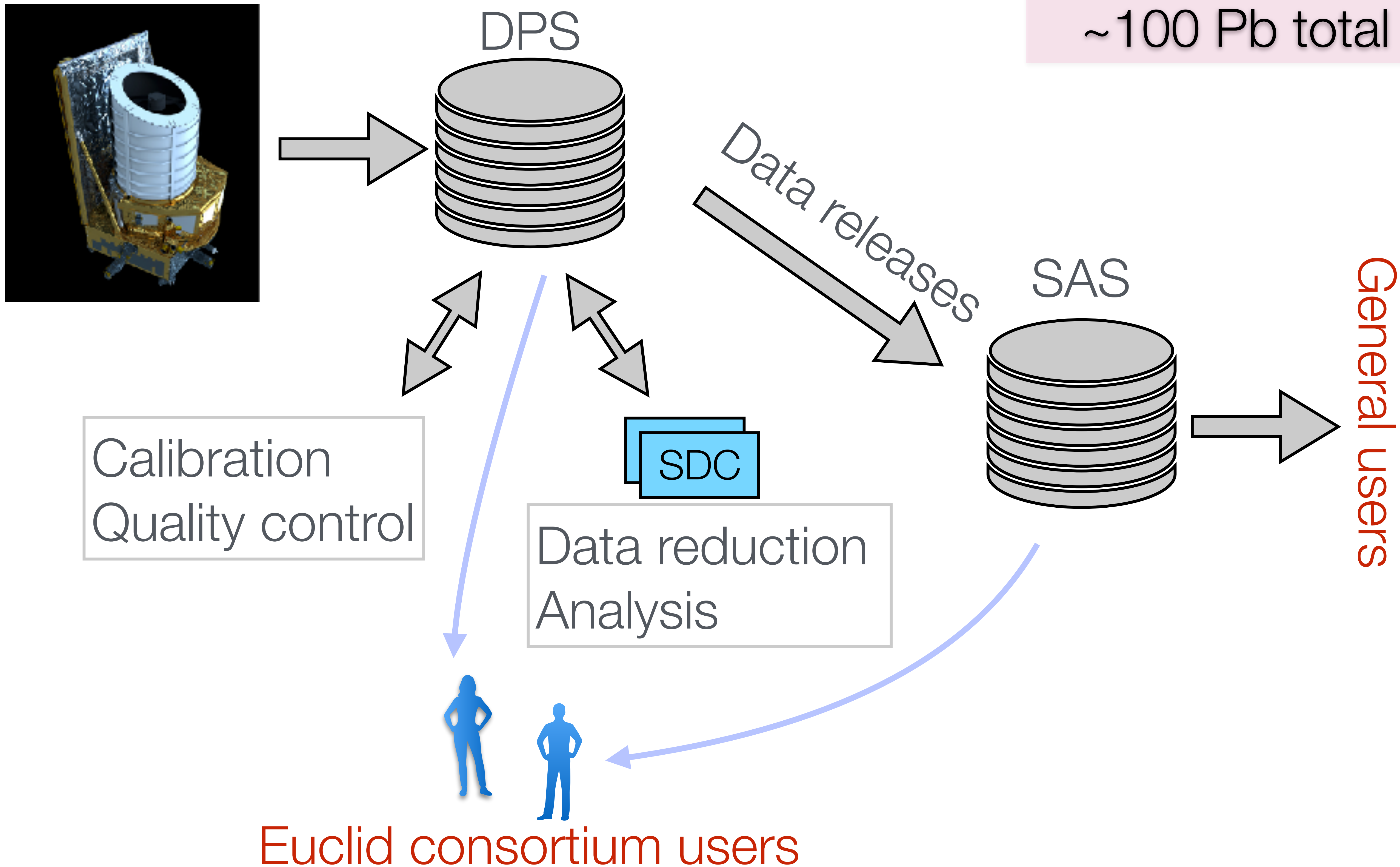


Euclid as an example



Euclid as an example

Total amount:
300 Tb Raw data
~100 Pb total



SQL - Structured Query Language

The standard way to interact with databases.

What we are doing next - and where it fits in

- The aim is always the data - today we will look a bit at how we can use a database to handle data.
- We will see how we can
  - Find data in a table
  - Combine tables
  - in the practical class: Create tables in a data-base
- There will be some technical detail today - this needs to be learned, but you should always keep in mind what your goal is: To do science with the data.

An example problem: keeping track of observations

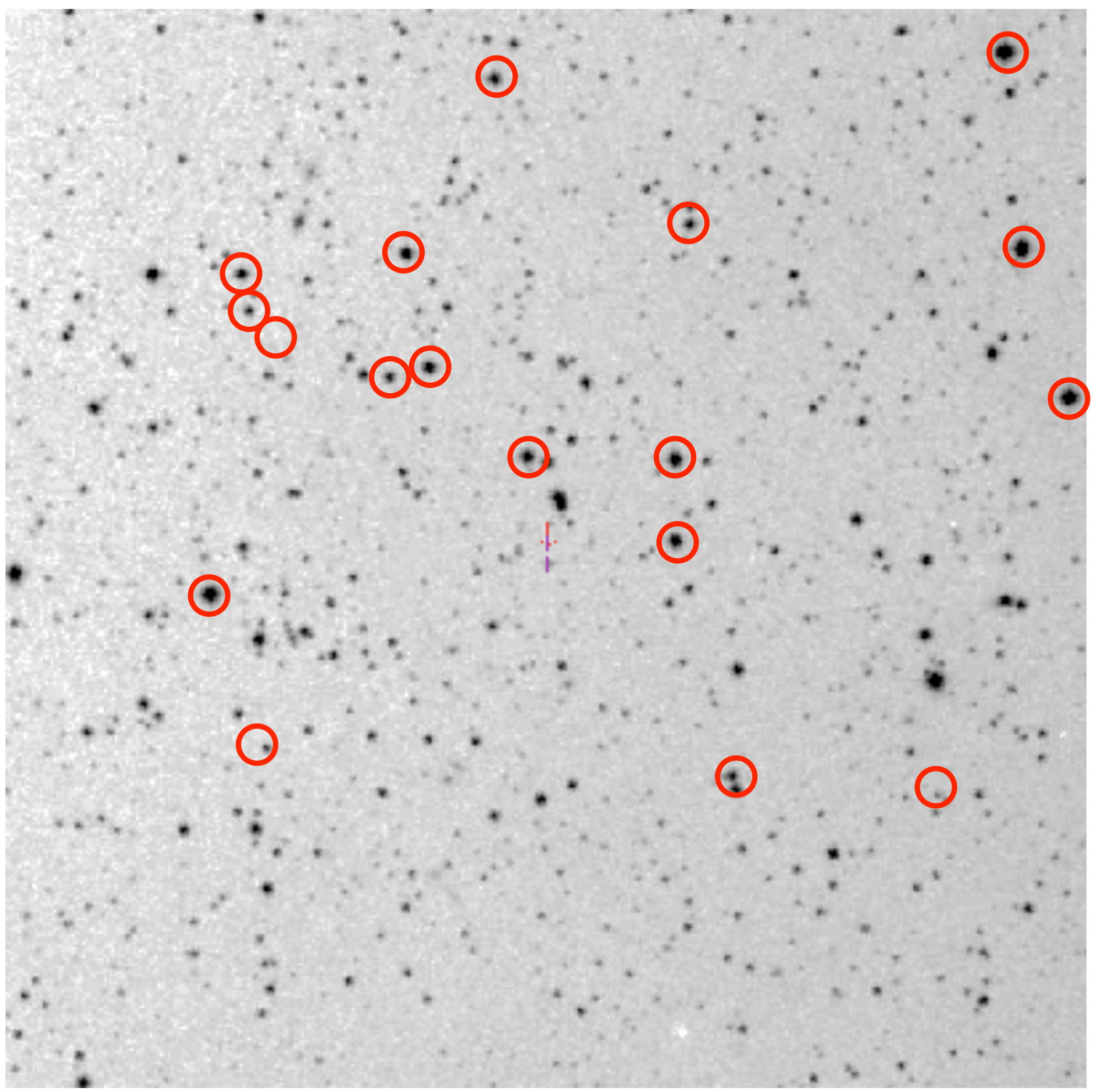
Observations:

```
# Field      Date          Exptime   Quality   WhereStored
StF-043  92.9885764     23.2       1         /disks/yaeps-1/StF-043.fits
StF-044  97.3323764     30.2       1         /disks/yaeps-1/StF-044.fits
StF-045  93.5532134     29.5       0.5       /disks/yaeps-1/StF-045.fits
```

An example problem: keeping track of observations

Observations:

```
# Field        Date          Exptime    Quality    WhereStored
StF-043   92.9885764      23.2         1          /disks/yaeps-1/StF-043.fits
StF-044   97.3323764      30.2         1          /disks/yaeps-1/StF-044.fits
StF-045   93.5532134      29.5         0.5        /disks/yaeps-1/StF-045.fits
```



Within each field we will detect a number of stars.

An example problem: keeping track of observations

Observations:

```
# Field       Date           Exptime   Quality   WhereStored
StF-043  92.9885764     23.2       1         /disks/yaeps-1/StF-043.fits
StF-044  97.3323764     30.2       1         /disks/yaeps-1/StF-044.fits
StF-045  93.5532134     29.5       0.5       /disks/yaeps-1/StF-045.fits
```

Stars:

```
# Star     Ra         Dec        g      r
 S1  198.8475000 10.5034722 14.5   15.2
 S2  198.5654167 11.0231944 15.3   15.4
 S5  198.9370833  9.9168889 16.4   15.8
 S7  199.2516667 10.3486944 14.6   14.1
```

If, for each star I keep information about the observations, I can waste a LOT of space. => Relational databases.

Relational databases:

Table A - lastnames

1	Smith
2	Kovacs
3	Tanahaka
.....	

Table B - course grades

1	2.3	5.4	6.2	7.8
3	8.5	7.4	9.2	8.8
7	2.2	7.0	8.2	7.5
9	8.0	7.0	8.0	7.5

Relational databases:

Table A - lastnames

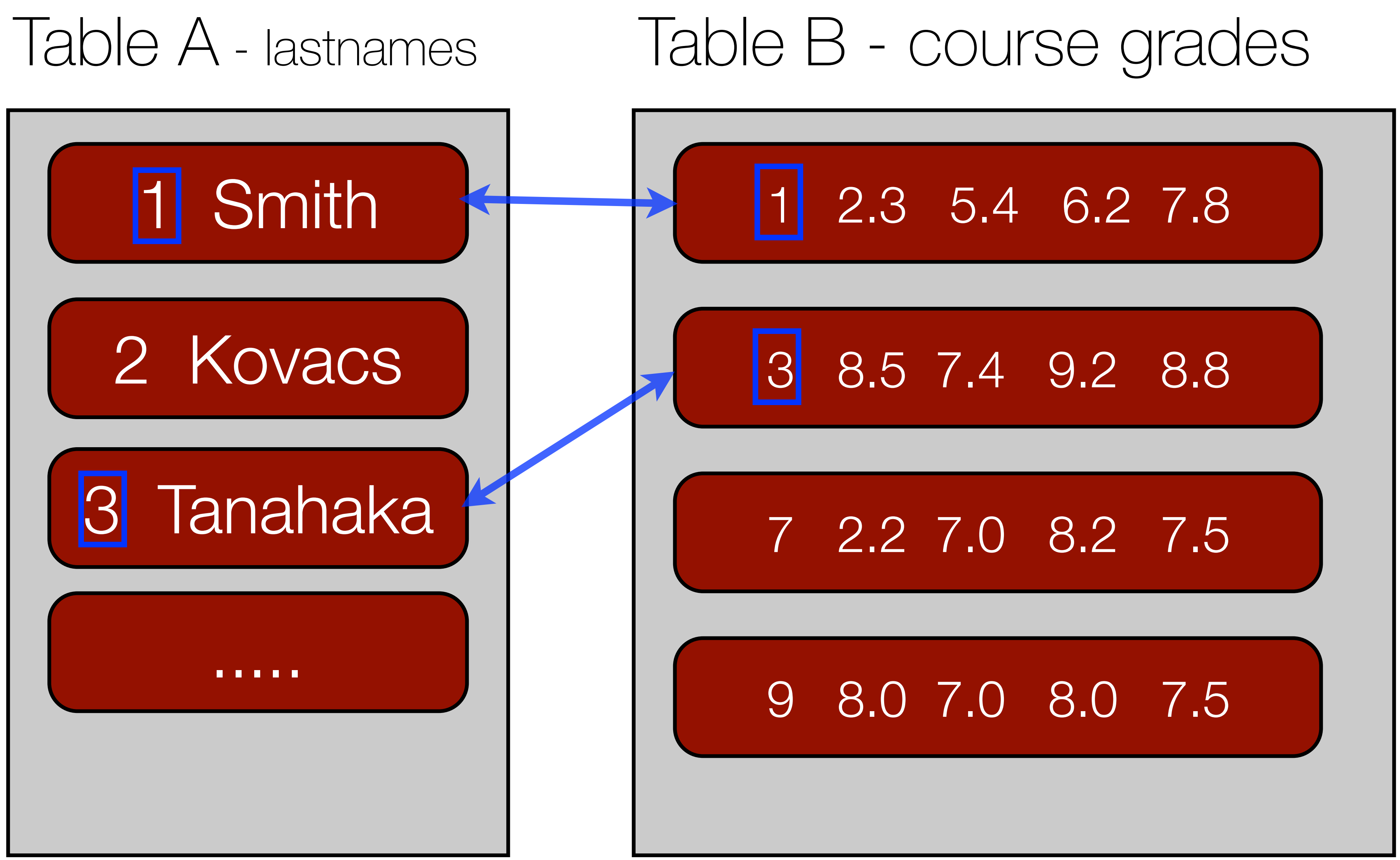
1	Smith
2	Kovacs
3	Tanahaka
.....	

Table B - course grades

1	2.3	5.4	6.2	7.8
3	8.5	7.4	9.2	8.8
7	2.2	7.0	8.2	7.5
9	8.0	7.0	8.0	7.5

While a relational database is formally defined with no reference to tables, it is useful to think of it as a collection of tables where (some) rows can be related.

Relational databases:



While a relational database is formally defined with no reference to tables, it is useful to think of it as a collection of tables where (some) rows can be related.

Relational databases - the observing example

Table: Observations

# Field	Date	Exptime	Quality	WhereStored
StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

Table: Stars

# Star	Ra	Dec	g	r
S1	198.8475000	10.5034722	14.5	15.2
S2	198.5654167	11.0231944	15.3	15.4
S5	198.9370833	9.9168889	16.4	15.8
S7	199.2516667	10.3486944	14.6	14.1

How should we link these?

One possibility: Create an ID column

Relational databases - the observing example

Table: Observations

# ID	Field	Date	Exptime	Quality	WhereStored
1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

Table: Stars

# FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

Relational databases - the observing example

Table: Observations

# ID	Field	Date	Exptime	Quality	WhereStored
1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

Table: Stars

# FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

Note that there are two IDs in the Stars table:

The ID of each star (StarID) **Primary key**

The ID of the field it was observed id (FieldID) **Foreign key**

Relational databases - the observing example

Table Observations

ID	Field	Date	Exptime	Quality	WhereStored
1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

Table Stars

FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

We would like to ask questions like:

1. Give me all stars brighter than r=14.5
2. How many stars have 0.1 < g-r < 0.4?
3. When did we observe S2?
4. Where is the FITS image stored for star S5?
5. Give me a list of all stars observed on the same FieldID

Choice of database solution

For concreteness I will use **sqlite** as my database as well as CasJobs

Lightweight & convenient

Advantages of sqlite: Light-weight, no need for complex setup, supports most of SQL. Easy to use for local work. Very widely used (e.g. Firefox, Chrome) and bindings for many languages.

Disadvantages: Not a client-server solution. Not all of SQL is supported and some features (e.g. ALTER TABLE) are only partially available.

Alternatives: MySQL, Oracle, PostgreSQL, Microsoft SQL Server.

Outline of creation of databases

- Determine the format for each table in your database - its *schema*. Insert this to create your table.

```
CREATE TABLE IF NOT EXISTS Stars (<schema>);
```

- Import data into each table.

```
.separator ,
.import YAEPS.stars-table-sqlite.dat Stars
```

The devil is in the details!

Querying databases - SQL

1. Give me all stars brighter than r=14.5

# FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

In SQL we do:

```
SELECT *
FROM Stars
WHERE r < 14.5
```

In sqlite:

```
sqlite> SELECT * FROM Stars WHERE r < 14.5;
4,2,S7,199.2516667,10.3486944,14.6,14.1
```

not the prettiest formatting (mysql is nicer) but good enough.

Breaking down the SELECT query:

1. Give me all stars brighter than r=14.5

Breaking down the SELECT query:

1. Give me all stars brighter than r=14.5

```
SELECT *
```

Return all columns for all the rows that matches the constraints. We can also specify specific columns.

Breaking down the SELECT query:

1. Give me all stars brighter than r=14.5

```
SELECT *
```

Return all columns for all the rows that matches the constraints. We can also specify specific columns.

```
FROM Stars
```

Use the table named Stars for this query.

Breaking down the SELECT query:

1. Give me all stars brighter than r=14.5

```
SELECT *
```

Return all columns for all the rows that matches the constraints. We can also specify specific columns.

```
FROM Stars
```

Use the table named Stars for this query.

```
WHERE r < 14.5
```

Only return those **rows** that satisfy the criteri(on/a) that we specify.

Choosing columns to output

FieldID	StarID	Star	Ra	Decl	g	r
1	1	S1	198.8475	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

```
SELECT Star, g, r
FROM Stars
```

Choosing columns to output

FieldID	StarID	Star	Ra	Decl	g	r
1	1	S1	198.8475	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

```
SELECT Star, g, r
FROM Stars
```

```
sqlite> SELECT Star, g, r FROM Stars;
S1,14.5,15.2
S2,15.3,15.4
S5,16.4,15.8
S7,14.6,14.1
```

Choosing columns to output

FieldID	StarID	Star	Ra	Decl	g	r
1	1	S1	198.8475	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

```
SELECT Star, g, r
FROM Stars
WHERE r < 14.5
```

But what if I want some combination of columns?

Choosing columns to output

FieldID	StarID	Star	Ra	Decl	g	r	g-r
1	1	S1	198.8475	10.5034722	14.5	15.2	-0.7
1	2	S2	198.5654167	11.0231944	15.3	15.4	-0.1
3	3	S5	198.9370833	9.9168889	16.4	15.8	0.6
2	4	S7	199.2516667	10.3486944	14.6	14.1	0.5

```
SELECT Star, g, r, g-r as gr
FROM Stars
WHERE FieldID = 1
```

Choosing columns to output

FieldID	StarID	Star	Ra	Decl	g	r	g-r
1	1	S1	198.8475	10.5034722	14.5	15.2	-0.7
1	2	S2	198.5654167	11.0231944	15.3	15.4	-0.1
3	3	S5	198.9370833	9.9168889	16.4	15.8	0.6
2	4	S7	199.2516667	10.3486944	14.6	14.1	0.5

```
SELECT Star, g, r, g-r as gr
FROM Stars
WHERE FieldID = 1
```

```
sqlite> SELECT Star, g, r, g-r as gr FROM Stars WHERE FieldID
= 1;
S1,14.5,15.2,-0.699999999999999
S2,15.3,15.4,-0.0999999999999996
```

Getting a few values - SQL flavours...

When you have very large tables, you often want to try out statements or just get few examples back. This is easy but depends on the SQL flavour you use:

Microsoft SQL (used in SDSS):

```
SELECT TOP 2 r FROM STARS
```

MySQL and sqlite:

```
SELECT r FROM STARS LIMIT 2
```

Oracle (used in AstroWISE):

```
SELECT r FROM STARS WHERE ROWNUM < 2
```

Recall:

Table Observations

# ID	Field	Date	Exptime	Quality	WhereStored
1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

Table Stars

# FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

Now let us go back to our questions:

3. When did we observe S2?
4. Where is the FITS image stored for star S5?
5. Give me a list of all stars observed on the same FieldID

# ID	Field	Date	Exptime	Quality	WhereStored
1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

# FieldID	StarID	Star	Ra	Dec	g	r
1	1	S1	198.8475000	10.5034722	14.5	15.2
1	2	S2	198.5654167	11.0231944	15.3	15.4
3	3	S5	198.9370833	9.9168889	16.4	15.8
2	4	S7	199.2516667	10.3486944	14.6	14.1

In these cases we need to be able to link information between two tables. In SQL we do this using JOINs

First a theoretical view:

Two sets of values: $\{x_i\}$ $\{y_j\}$ (the elements can be vectors/matrices etc)

Possible ways to combine:

Union: $\{x_i, y_j | i=1, n; j=1, m\}$ elements must be the same

Cross-join: $\{(x_i, y_j) | i=1, n; j=1, m\}$ ie. all possible pairs

Left Outer join: $\{(x_i, y_i)$ if y_i exists, (x_i, NULL) otherwise$\}$

Right Outer join: $\{(x_i, y_i)$ if x_i exists, (NULL, y_i) otherwise$\}$

Inner join: $\{(x_i, y_i)$ if y_i exists$\}$

All these are supported in SQL.

UNION

It must make sense to glue the tables together!

```
Select TOP 10
ra, dec
FROM SpecPhoto
WHERE ra > 120
AND DEC < 0
```

Table 1

```
UNION
```

```
Select TOP 10
ra, dec
From SpecPhoto
WHERE ra < 10
AND DEC > 0
```

Table 2

Try it in SDSS!

UNION

It must make sense to glue the tables together!

```
Select TOP 10
ra, dec
FROM SpecPhoto
WHERE ra > 120
AND DEC < 0
```

Table 1

Ra	Dec

```
UNION
```

```
Select TOP 10
ra, dec
From SpecPhoto
WHERE ra < 10
AND DEC > 0
```

Table 2

Try it in SDSS!

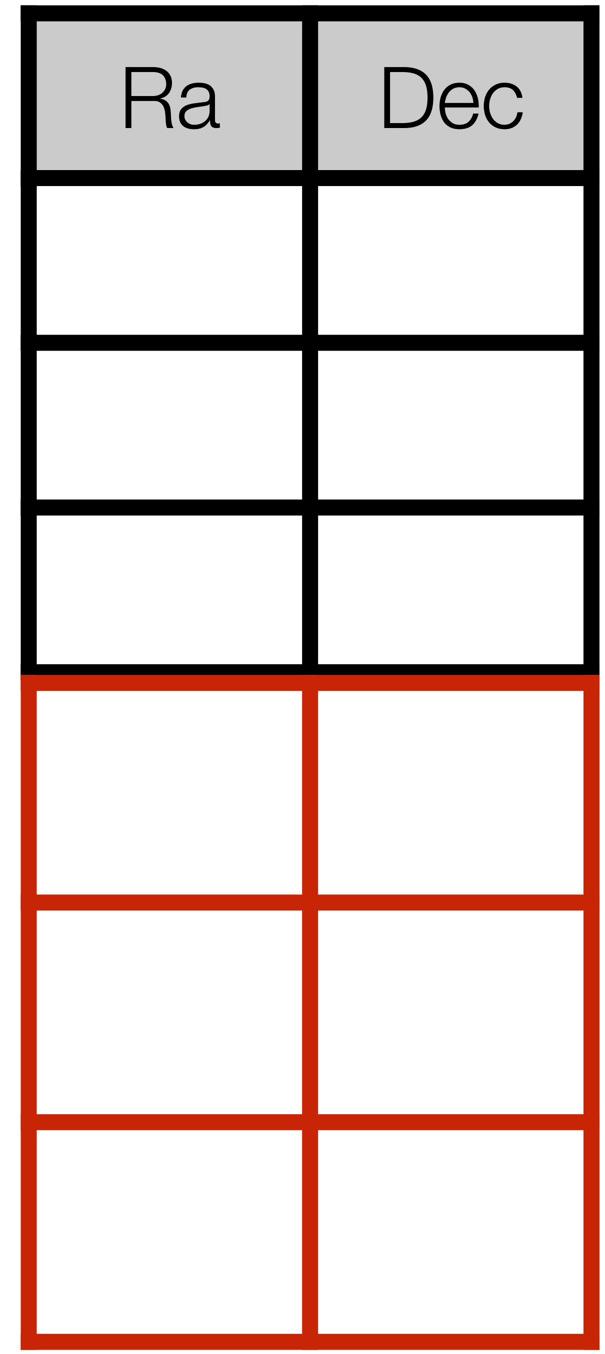
UNION

It must make sense to glue the tables together!

```
Select TOP 10
ra, dec
FROM SpecPhoto
WHERE ra > 120
AND DEC < 0

UNION

Select TOP 10
ra, dec
From SpecPhoto
WHERE ra < 10
AND DEC > 0
```

Table 1

Table 2



Try it in SDSS!

How do you combine?

ID	1
u	19.3

ID	2
u	17.5

ID	3
u	20.5

ID	1
EW(	75Å

ID	3
EW(	0.5Å

How do you combine?

ID	u	EW(Ha)
1	19.3	75Å
3	20.5	0.5Å

How do you combine?

ID	u	EW(Ha)
1	19.3	75Å
3	20.5	0.5Å

```
SELECT P.u, S.EW
FROM Photo as P
JOIN  Spectro as S
ON P.ID=S.ID
```

How do you combine?

ID	1
u	19.3
ID	2
u	17.5
ID	3
u	20.5

ID	1
EW(	75Å
ID	3
EW(	0.5Å

JOIN

ID	u	EW(Ha)
1	19.3	75Å
3	20.5	0.5Å

```
SELECT P.u, S.EW
FROM Photo as P
JOIN  Spectro as S
ON P.ID=S.ID
```

OR

```
SELECT P.u, S.EW
FROM Photo as P,
Spectro as S
WHERE P.ID=S.ID
```

How do you combine?

ID	1
u	19.3
ID	2
u	17.5
ID	3
u	20.5

ID	1
EW(	75Å
ID	3
EW(	0.5Å

JOIN →

ID	u	EW(Ha)
1	19.3	75Å
3	20.5	0.5Å

```
SELECT P.u, S.EW
FROM Photo as P
JOIN  Spectro as S
ON P.ID=S.ID
```

Explicit INNER JOIN

OR

```
SELECT P.u, S.EW
FROM Photo as P,
Spectro as S
WHERE P.ID=S.ID
```

Implicit INNER JOIN

(or *old-style* INNER JOIN)

Explicit vs implicit JOINs

JOIN ... ON a=b
or
WHERE a = b

Mostly up to you - there should be no significant difference between the two.

The main disadvantage of an implicit JOIN is that you have less control of the order things are done if you have more than two tables.

I personally prefer explicit JOINs because they show more clearly what your intention is and if you have a problem because your query runs too slowly, you can more easily figure out the execution order.

ID	u	EW(Ha)
1	19.3	75Å
2	17.5	NULL
3	20.5	0.5Å

But if we want to keep **all** possible pairs we need an **OUTER JOIN**

ID	u	EW(Ha)
1	19.3	75Å
2	17.5	NULL
3	20.5	0.5Å

But if we want to keep **all** possible pairs we need an **OUTER JOIN**

```
SELECT P.u, S.z
FROM Photo as P
LEFT OUTER JOIN  Spectro as S
ON P.ID=S.ID
```

ID	u	EW(Ha)
1	19.3	75Å
2	17.5	NULL
3	20.5	0.5Å

Returning to our questions:

3. When did we observe S2?

```
# ID Field        Date       Exptime   Quality   WhereStored
  1 StF-043  92.9885764     23.2        1        /disks/yaeps-1/StF-043.fits
  2 StF-044  97.3323764     30.2        1        /disks/yaeps-1/StF-044.fits
  3 StF-045  93.5532134     29.5        0.5      /disks/yaeps-1/StF-045.fits
```

```
# FieldID StarID  Star      Ra          Dec        g      r
   1        1      S1   198.8475000 10.5034722 14.5   15.2
   1        2      S2   198.5654167 11.0231944 15.3   15.4
   3        3      S5   198.9370833  9.9168889 16.4   15.8
   2        4      S7   199.2516667 10.3486944 14.6   14.1
```

Our link is FieldID in Stars to ID in Observations

Returning to our questions:

3. When did we observe S2?

#	ID	Field	Date	Exptime	Quality	WhereStored
	1	StF-043	92.9885764	23.2	1	/disks/yaeps-1/StF-043.fits
	2	StF-044	97.3323764	30.2	1	/disks/yaeps-1/StF-044.fits
	3	StF-045	93.5532134	29.5	0.5	/disks/yaeps-1/StF-045.fits

#	FieldID	StarID	Star	Ra	Dec	g	r
	1	1	S1	198.8475000	10.5034722	14.5	15.2
	1	2	S2	198.5654167	11.0231944	15.3	15.4
	3	3	S5	198.9370833	9.9168889	16.4	15.8
	2	4	S7	199.2516667	10.3486944	14.6	14.1

Our link is FieldID in Stars to ID in Observations

```
select s.Star, o.Field, o.Date
from
  stars as s
  JOIN Observations as o
  ON s.fieldID = o.ID
Where Star = 'S2'
```

Returning to our questions:

3. When did we observe S2?

```
select s.Star, o.Field, o.Date
from
  stars as s
  JOIN Observations as o
  ON s.fieldID = o.ID
Where Star = 'S2'
```

We must specify what table to get a quantity from.

JOIN the tables explicitly

Choose our star

Useful: We can do **multiple** stars by changing the WHERE statement to:

Returning to our questions:

3. When did we observe S2?

```
select s.Star, o.Field, o.Date
from
  stars as s
  JOIN Observations as o
  ON s.fieldID = o.ID
Where Star = 'S2'
```

We must specify what table to get a quantity from.

JOIN the tables explicitly

Choose our star

Useful: We can do **multiple** stars by changing the WHERE statement to:

```
Where Star IN ('S2', 'S1')
```

Returning to our questions:

3. When did we observe S2?

```
select s.Star, o.Field, o.Date
from
  stars as s
  JOIN Observations as o
  ON s.fieldID = o.ID
Where Star = ‘S2’
```

We must specify what table to get a quantity from.

JOIN the tables explicitly

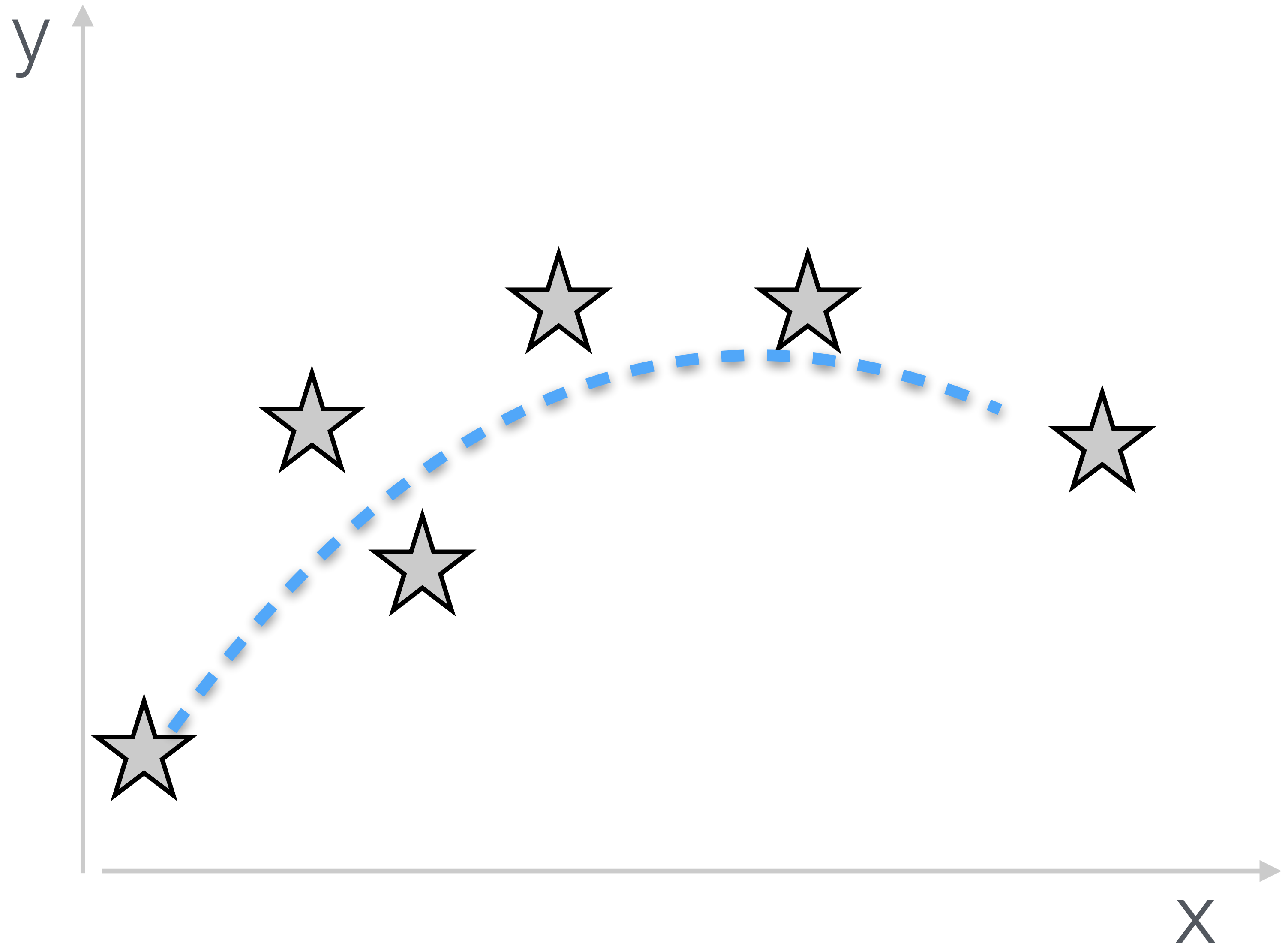
Choose our star

Useful: We can do **multiple** stars by changing the WHERE statement to:

```
Where Star IN (‘S2’, ‘S1’)
```

That’s it for now - more advanced topics for SQL are included at the end of the slides for self-study

Regression



Standard linear regression

In this case we typically have p observables at each of N points (predictors) and want to predict a response variable, y_i at each x_i

A standard way to fit this is to minimise the residual sum of squares (RSS) or N times MSE:

$$\mathrm{RSS} = \sum_i (y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the estimate of y_i. Which for linear regression is:

$$\hat{y}_i = \theta_0 + \sum_{j=1}^{p} \theta_j x_{ij}$$

Common formulation - the design matrix

The problem to solve is then often written:

$$Y = \mathrm{M}\boldsymbol{\theta}$$

Where Y is $(y_1, y_2, \ldots, y_N)$ and $\boldsymbol{\theta}=(\theta_1, \theta_2, \ldots, \theta_p)$

M is known as the design matrix as mentioned earlier and is

$$\mathrm{M} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \multicolumn{5}{c}{\dotfill} \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,p} \end{pmatrix}$$

Common formulation - the design matrix

$$Y = \mathsf{M}\boldsymbol{\theta}$$

If we also introduce the covariance matrix of uncertainties on Y:

$$\mathsf{C} = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{pmatrix}$$

the general solution of the linear regression is given by

$$\boldsymbol{\theta} = \left(\mathsf{M}^T \mathsf{C}^{-1} \mathsf{M}\right)^{-1} \left(\mathsf{M}^T \mathsf{C}^{-1} Y\right)$$

with uncertainties on the parameters given by

$$\Sigma_{\boldsymbol{\theta}} = \left(\mathsf{M}^T \mathsf{C}^{-1} \mathsf{M}\right)^{-1}$$

Basis functions

Note that a regression would still be linear if we transformed all the predictors with a function:

$$y_i = \theta_0 + \sum_{j=1}^{N} \theta_j \phi_j(x_i)$$

This is known as basis function regression and can for instance be done using BasisFunctionRegression in astroML.linear_model.

Linear regression - one way to do it in Python:

```
M, T = pickle_from_file('T-vs-colour-regression.pkl')
```

```
from astroML.linear_model import LinearRegression

model = LinearRegression(fit_intercept=True)
result = model.fit(M, T/1e4)
Tpred = model.predict(M)

result.coef_   # Coefficients of the fit.
```

Note that the intercept is the first element of the coefficient array. So if M is N_{obj}x4, `coef_` will be 5 elements long.

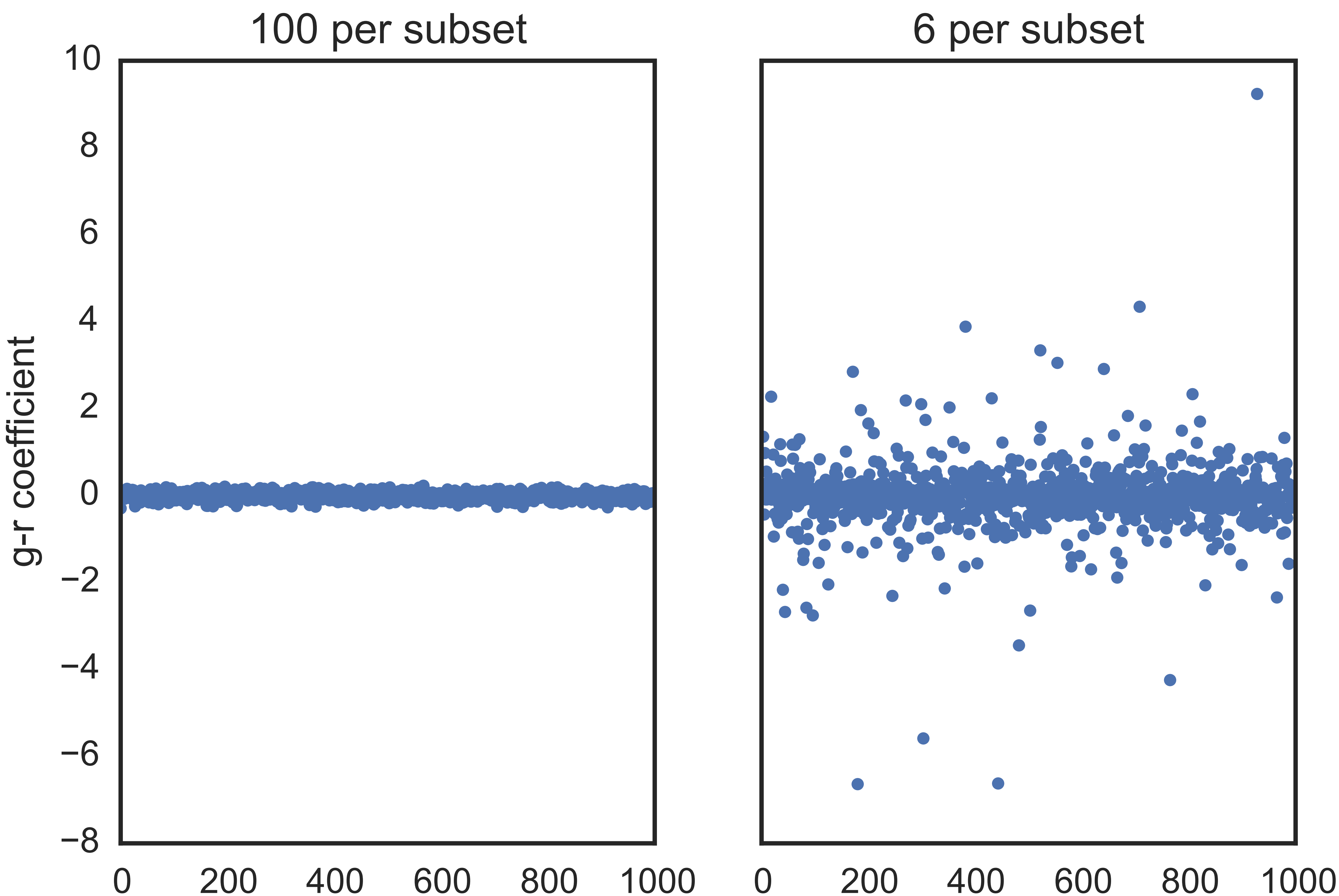
Regularising linear regression - ridge regression

When N is comparable to p, linear regression typically has large variance. To combat this we might want to trade a bit of bias for a lower variance.

As an example we can fit this:

$$T = \theta_0 + \theta_1(u-g) + \theta_2(g-r) + \theta_3(r-i) + \theta_5(i-z)$$

to the sspp dataset in astroML, where T is the effective temperature of the stars and the colours should be obvious.



Fits of 1000 random subsets with a linear regressor and showing just one coefficient.

Regularising linear regression - ridge regression

When N is comparable to p, linear regression typically has large variance. To combat this we might want to trade a bit of bias for a lower variance.

We do this by **regularising** the solution and minimise:

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here we limit the *size* of the parameter vector $\boldsymbol{\beta}$.

This does introduce a **regularisation parameter**: λ

Next time we will look at systematic ways to determine the regularisation parameter.

Regularising linear regression - ridge regression

We do this by **regularising** the solution and minimise:

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here we limit the *size* of the parameter vector $\boldsymbol{\beta}$.

When applied to linear regression, this leads to **ridge regression**.

Ridge regression - how to

```
from sklearn.linear_model import Ridge
model = Ridge(alpha=0.05, normalize=True)

result = model.fit(M, T/1e4)
Tpred = model.predict(M)

res.coef_   # Coefficients of the fit.
```

Note: alpha = λ in my (and others') notation.

Very similar to LinearRegression - with one exception:

The normalize keyword.

Ridge regression - normalisation

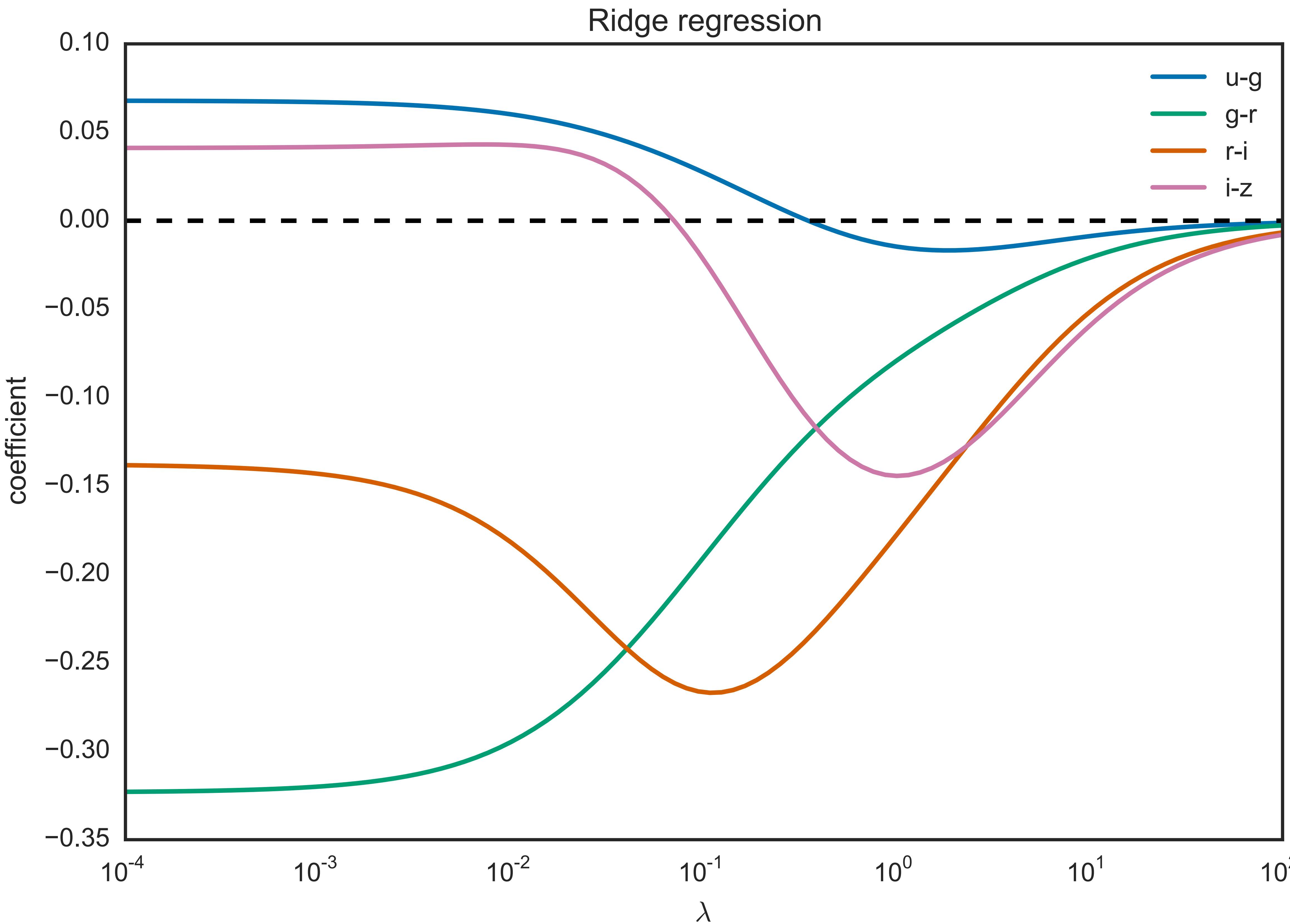
Ridge regression can also be seen to want to minimise

$$\sum_{i=1}^{N}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_{ij}\right)^2$$

subject to

$$\sum_{j=1}^{p}\theta_j^2 \leq s$$

Obviously that length will depend on the **units** of x_{ij}. It is therefore common to “**whiten**” or **standardize** x by dividing by its standard deviation (ideally robustly).


Ridge regression
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
coefficient
10^-4
10^-3
10^-2
10^-1
10^0
10^1
10^2
λ
u-g
g-r
r-i
i-z

Ridge regression - degrees of freedom

There are of course p parameters, but because of the constraints in ridge regression, the effective number of degrees of freedom is not p - rather it is a smaller number depending on λ.

As far as I know this is not available through sklearn, but you can calculate it from the SVD of X:

$$\mathsf{X} = \mathsf{UDV}^T$$

If the diagonal entries in D are d_j, the d.o.f. is:

$$\mathrm{df} = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

Regularising linear regression - LASSO

Ridge regression minimises the l_2 norm of the coefficients. The Lasso (Least Absolute Shrinkage and Selection Operator) minimises the l_1 norm.

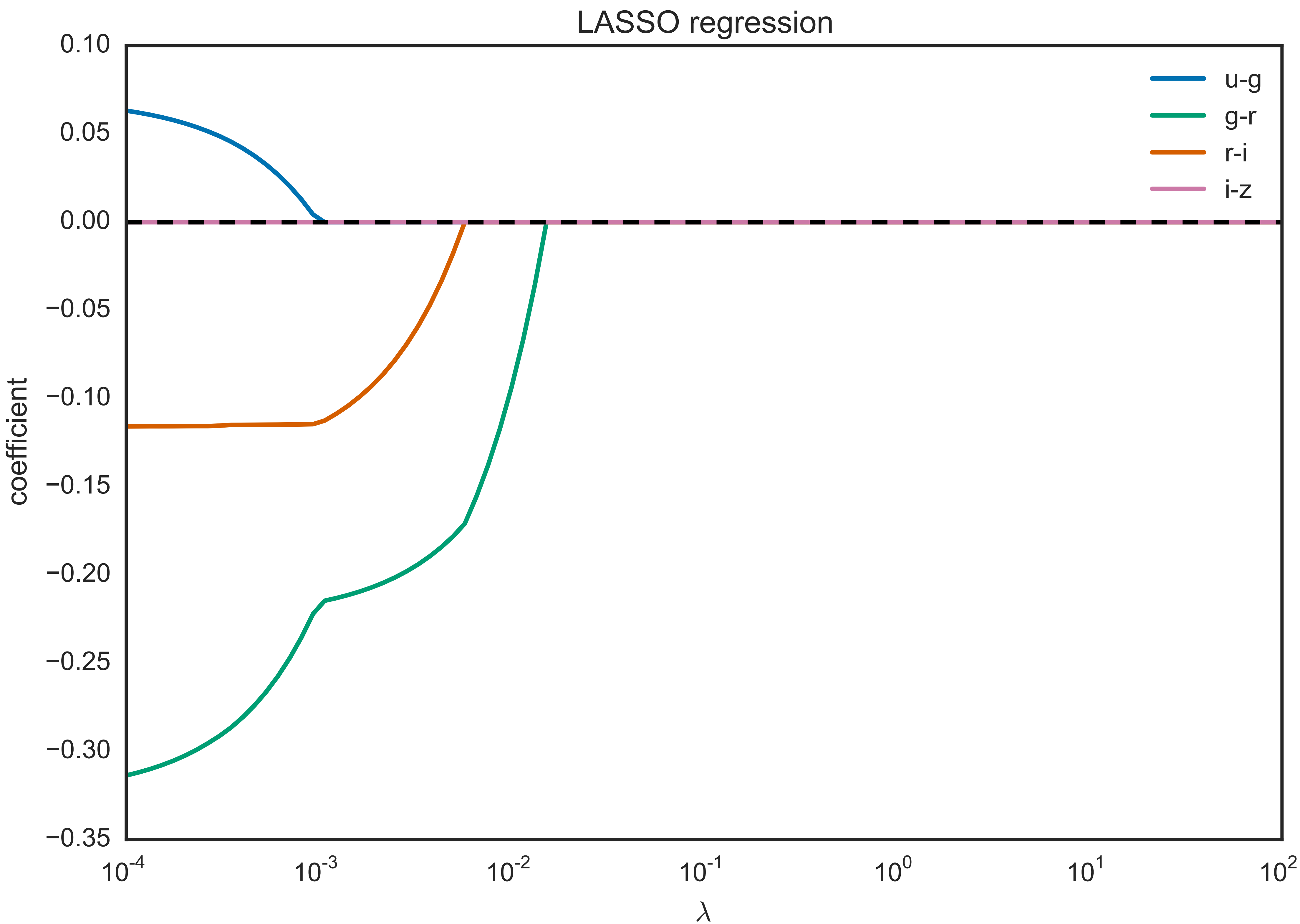
The minimisation is now of

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} |\theta_j|$$

Lasso regression - how to

```
from sklearn.linear_model import Lasso
model = Lasso(alpha=alpha, normalize=True)

result = model.fit(M, T/1e4)
Tpred = model.predict(M)

res.coef_   # Coefficients of the fit.
```

So just like ridge regression - but the result is somewhat different


LASSO regression
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
coefficient
10^-4
10^-3
10^-2
10^-1
10^0
10^1
10^2
λ
u-g
g-r
r-i
i-z

Variable selection with the Lasso

So some coefficients end up being set to zero!

This fact means that the lasso performs ***variable selection***.

This feature of the lasso can be phrased to say that it returns **sparse models**.

Basically it can be interpreted to say which variables (predictors) are most important for predicting the output.

Subset selection

The most rigorous selection of variables is what is called ***subset selection***. This basically considers all possible combinations of parameters:

1. Set M_0 to be null model with no predictors
2. for k=1,2…p
   - Fit all $\binom{p}{k}$ models that contain exactly k predictors
   - Pick the model among these that has the lowest RSS and call this M_k.
3. Select the best model among these M by CV or similar.

No ready made routine for this - but similar tools are available in sklearn.feature_selection.

So what do I do now?

Check out the course git repository

Read the math reminder document

Go through the Python & topcat "problem set"
(try to solve it before looking at the solution suggestion)

Try out the creation of sqlite databases

Practicalities:
creating a table

We will work with databases in the next practical.

At the end of the lecture notes the details of **creation** of databases are provided. Read through this before the practical but I will not go through in *detail* today!

Further topics in SQL - self-study

Sorting in SQL - ORDER BY

Sorting your output on some variable is simple:

```
SELECT Field, Quality
FROM Observations
ORDER BY Quality
```

What is the first field?

Sorting in SQL - ORDER BY

Sorting your output on some variable is simple:

```
SELECT Field, Quality
FROM Observations
ORDER BY Quality
```

What is the first field?

```
sqlite> SELECT Field, Quality FROM Observations ORDER BY Quality;
StF-051,0.0
StF-045,0.5
StF-048,0.7
StF-043,1.0
StF-044,1.0
StF-046,1.0
StF-047,1.0
StF-049,1.0
StF-050,1.0
```

When you don’t need everything

You want to make a histogram of the magnitude distribution of stars in the SDSS. There are **260,562,744** stars - do you need to download them all?

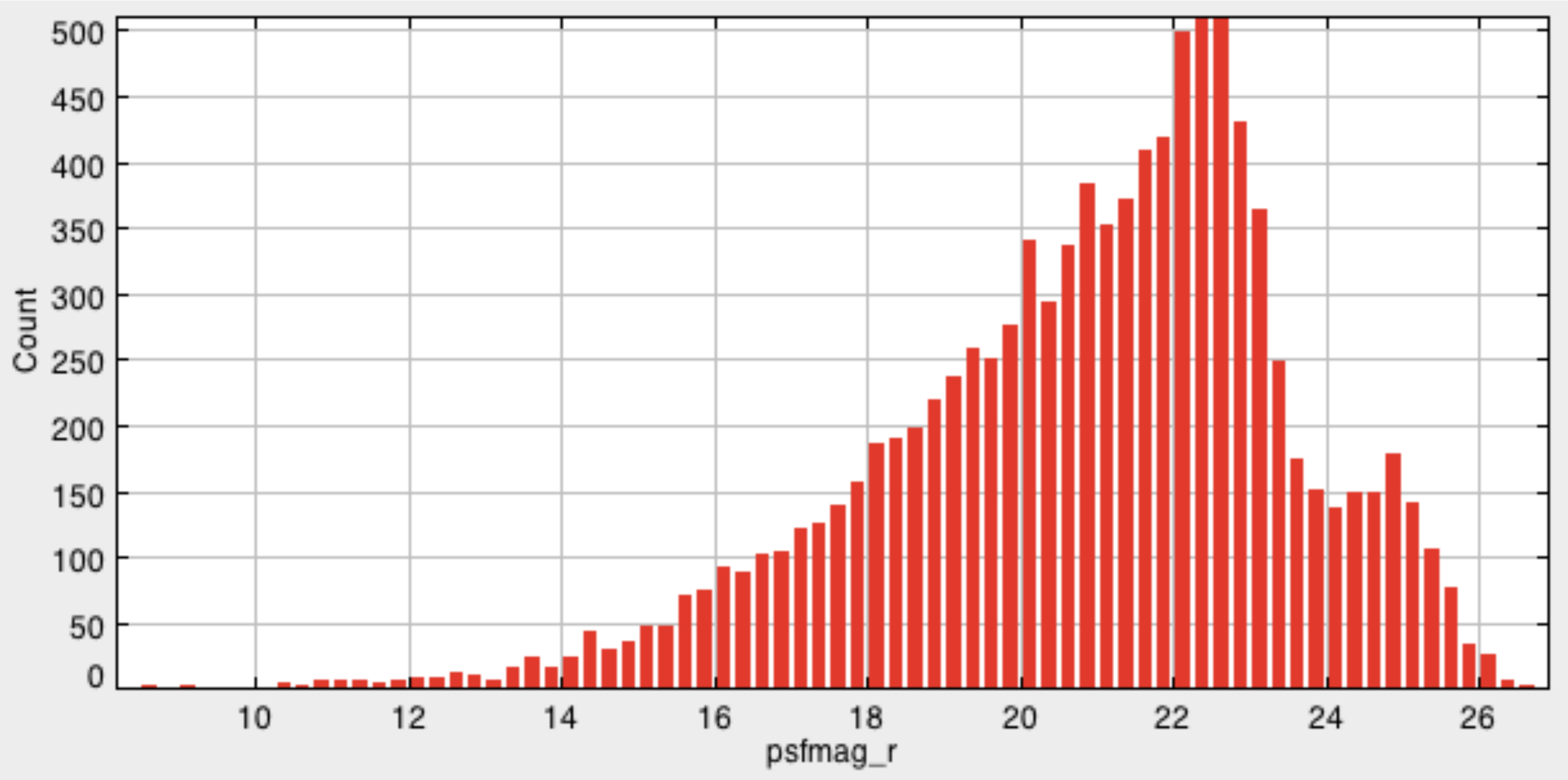
No.

When you don't need everything

You want to make a histogram of the magnitude distribution of stars in the SDSS. There are **260,562,744** stars - do you need to download them all?

No.

But you can if you want - here are the first 10,000. It can take a lot of time though!



Grouping your output

A better solution is sometimes to let the database server do the work. To do that we need to group our output.

Let us look at a simple case - we want to count the number of stars per field.

FieldI	StarI	Star	Ra	Decl	g	r	
1	1	S1	198.8475	0.503472	14.5	15.2	2
1	2	S2	98.565416	1.023194	15.3	15.4	
3	3	S5	98.937083	9.916888	16.4	15.8	1
2	4	S7	99.251666	0.348694	14.6	14.1	1

Grouping your output

Counting the number of stars per field.

FieldID	StarID	Star	Ra	Decl	g	r	
1	1	S1	198.8475	0.503472	14.5	15.2	2
1	2	S2	98.565416	1.023194	15.3	15.4	
3	3	S5	98.937083	9.916888	16.4	15.8	1
2	4	S7	99.251666	0.348694	14.6	14.1	1

The statement we need is: GROUP BY

```
SELECT FieldID, COUNT(*) as NperField
FROM Stars
GROUP BY FieldID
```

Accumulative functions - a side-step

SQL has certain functions that are ‘accumulative’:

COUNT
SUM
AVG
MIN
MAX

```
SELECT COUNT(*) as Nstars
FROM Stars
```

FieldI	StarI	Star	Ra	Decl	g	r
1	1	S1	198.8475	0.503472:	14.5	15.2
1	2	S2	98.565416	1.023194	15.3	15.4
3	3	S5	98.937083	9.916888	16.4	15.8
2	4	S7	99.251666	0.348694	14.6	14.1

GROUP BY - operating on accumulative functions

On their own these functions work on everything in one bunch - often not what you want. This is the role of **GROUP BY**.

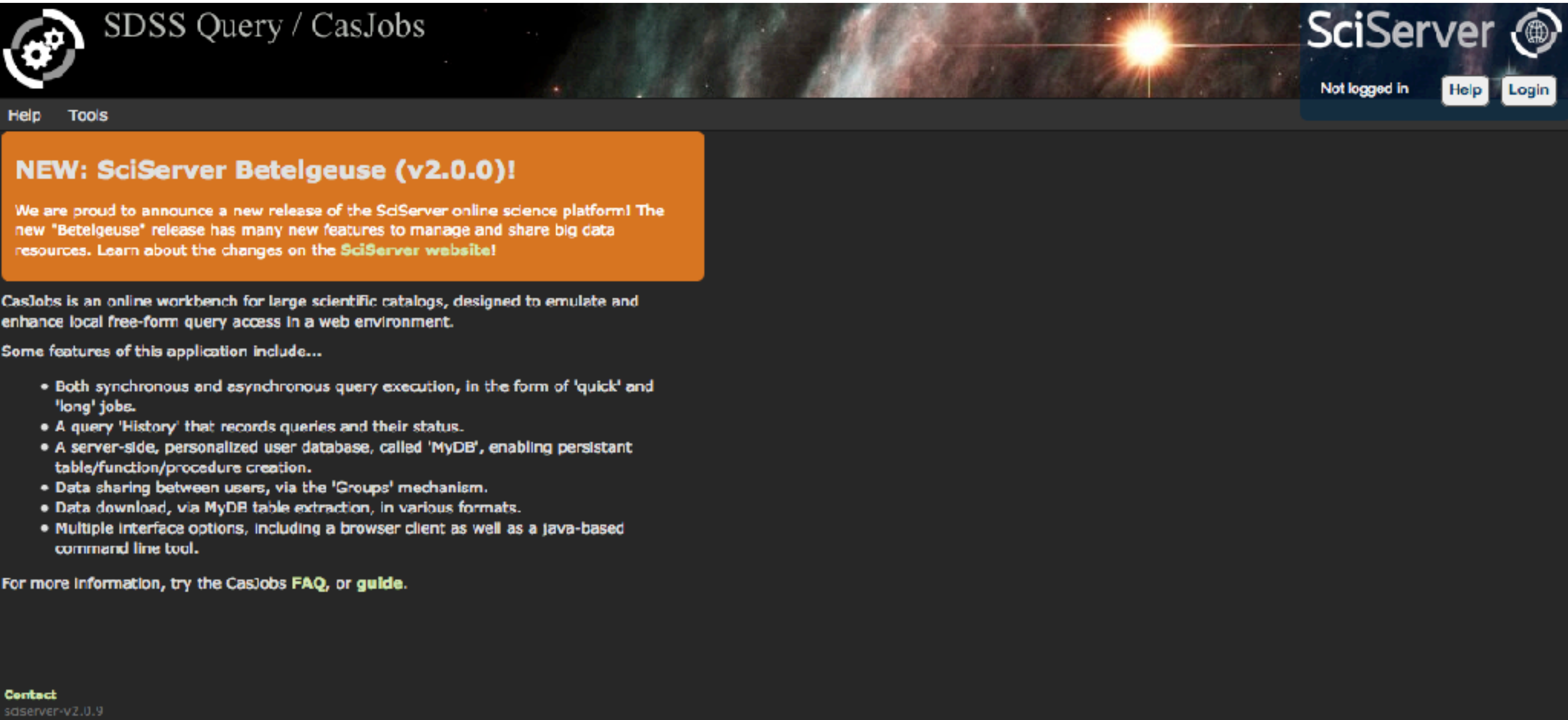
```
select fieldID, avg(r) as 'mean r'
from stars
group by FieldID;
```

Big data easily: SDSS & SQL

This also allows us to get to Kepler, TESS, GALEX and more in the same way.

The SDSS database

http://skyserver.sdss.org/casjobs/default.aspx



Create an account here when you can!

Ordering & Groupings

Counting objects in bins:

```
SELECT .2*(.5+floor(g.r/.2)) as mag,
       count(*) as num
FROM GALAXY as g
GROUP BY .2*(.5+floor(g.r/.2))
ORDER BY mag
```

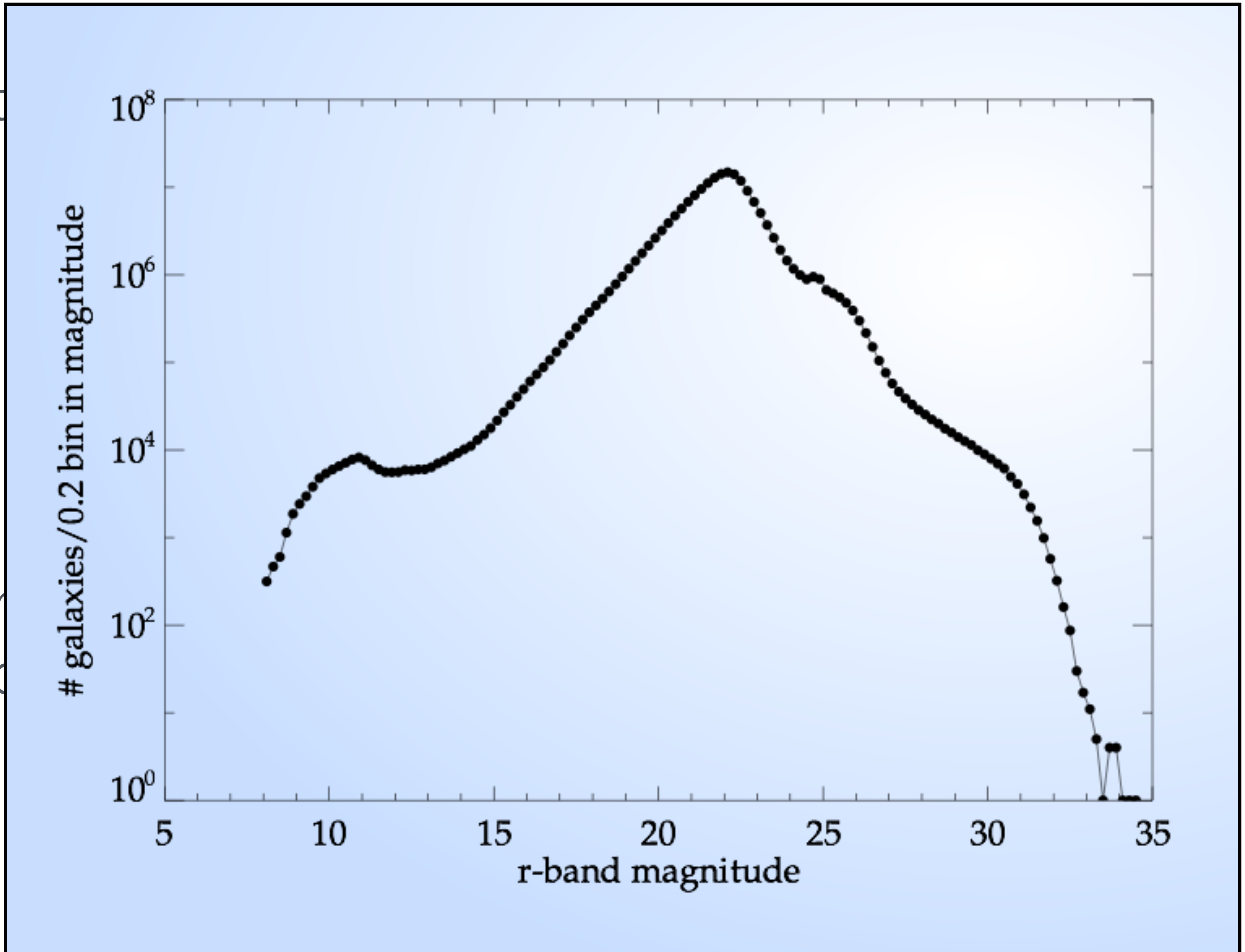
In general if you want to make C bins per unit for variable x, you use:

$$\frac{1}{C}\left(0.5 + \lfloor Cx \rfloor\right)$$

Ordering & Groupings

Count

In ger
unit fo



Creating intermediate tables

The result of a SELECT query is another table - we can therefore use this into another query.

As an example we can use this to create a histogram of the first 10,000 stars in the SDSS Star table:

```
select FLOOR(psfMag_r*10+0.5)/10
as rmag, COUNT(*) as N
  FROM
    (select TOP 10000 psfMag_r
    FROM Star) smallcat
GROUP BY (FLOOR(psfMag_r*10+0.5))
ORDER BY rmag
```

Selects from

What is SQL

- It is a declarative language to create database tables, and inserting/updating, deleting and querying information from these tables. It is often said to have two constituent parts:

- Data definition language (DDL): Define database structure (through schemas) and manage data access.

- Data manipulation language (DML): creation, deletion, updating and querying of content.

The next steps

Read through the final part of the lecture notes & start trying out sqlite3. In the practical we will try out several of these commands on tables you create yourself & also using SDSS.

Creation of databases & using them from Python

Creating a sqlite database

Let us set up a simple database to start with:

```
> sqlite3 MLD19.db
SQLite version 3.8.10.2 2015-05-20 18:17:19
Enter ".help" for usage hints.
sqlite> .tables
sqlite> .exit
```

This should give you nothing because there are no tables.
We need to make one.

Step 2 - inserting a table

Get: YAEPS.stars-table-sqlite.dat and sqlite3-make-stars-table.sql from Blackboard. **Edit** the latter to reflect the location of YAEPS.stars-table-sqlite.dat

When that is done:

```
> sqlite3 MLD19.db
SQLite version 3.8.10.2 2015-05-20 18:17:19
Enter ".help" for usage hints.
sqlite> .read sqlite3-make-stars-table.sql
sqlite> .tables
Stars
```

Magic?

First we need to create a schema

To do this we need to understand our data:

```
# StarID FieldID  Star      Ra          Dec        g      r
    1         1     S1  198.8475000 10.5034722 14.5   15.2
    2         1     S2  198.5654167 11.0231944 15.3   15.4
    3         3     S5  198.9370833  9.9168889 16.4   15.8
    4         2     S7  199.2516667 10.3486944 14.6   14.1
```

Schema

Column	Type	SQL type	Other
StarID	Integer	INT	PrimaryKey, Unique
FieldID	Integer	INT	ForeignKey
Star	String	varchar(10)	Length < 10
Ra	Real number	DOUBLE	
Dec	Real number	DOUBLE	
g	Real number	DOUBLE	
r	Real number	DOUBLE	

Creating databases and tables

Our star table is created with:

```
CREATE TABLE IF NOT EXISTS Stars (
    StarID INT,
    FieldID INT,
    Star varchar(10),
    ra DOUBLE,
    decl DOUBLE,
    g FLOAT,
    r FLOAT,
    UNIQUE(StarID),
    PRIMARY KEY(StarID),
    FOREIGN KEY(FieldID) REFERENCES Observations(ID)
    );
```

You can type this on the command line, but it is easier to store it in a file!

Creating databases and tables

Our star table is created with:

```
CREATE TABLE IF NOT EXISTS Stars (
    StarID INT,
    FieldID INT,
    Star varchar(10),
    ra DOUBLE,
    decl DOUBLE,
    g FLOAT,
    r FLOAT,
    UNIQUE(StarID),
    PRIMARY KEY(StarID),
    FOREIGN KEY(FieldID) REFERENCES Observations(ID)
    );
```

Notice the definition of strings

You can type this on the command line, but it is easier to store it in a file!

Creating databases and tables

Our star table is created with:

```
CREATE TABLE IF NOT EXISTS Stars (
    StarID INT,
    FieldID INT,
    Star varchar(10),
    ra DOUBLE,
    decl DOUBLE,
    g FLOAT,
    r FLOAT,
    UNIQUE(StarID),
    PRIMARY KEY(StarID),
    FOREIGN KEY(FieldID) REFERENCES Observations(ID)
    );
```

This indicates columns where each row has to have a unique value

You can type this on the command line, but it is easier to store it in a file!

Creating databases and tables

Our star table is created with:

```
CREATE TABLE IF NOT EXISTS Stars (
    StarID INT,
    FieldID INT,
    Star varchar(10),
    ra DOUBLE,
    decl DOUBLE,
    g FLOAT,
    r FLOAT,
    UNIQUE(StarID),
    PRIMARY KEY(StarID),
    FOREIGN KEY(FieldID) REFERENCES Observations(ID)
    );
```

This can optionally be used to indicate the primary key in this database

You can type this on the command line, but it is easier to store it in a file!

Creating databases and tables

Our star table is created with:

```
CREATE TABLE IF NOT EXISTS Stars (
    StarID INT,
    FieldID INT,
    Star varchar(10),
    ra DOUBLE,
    decl DOUBLE,
    g FLOAT,
    r FLOAT,
    UNIQUE(StarID),
    PRIMARY KEY(StarID),
    FOREIGN KEY(FieldID) REFERENCES Observations(ID)
    );
```

Optional: Use this to indicate foreign keys in the table. This can be useful for clarity at least.

You can type this on the command line, but it is easier to store it in a file!

Creating databases and tables

well, we need some more (this is database specific:)

```
.separator ,
.import YAEPS.stars-table-sqlite.dat Stars
```

These quick import routines are not part of SQL so vary from database to database (but they are handy!)

Creating databases and tables

well, we need some more (this is database specific:)

```
.separator ,
.import YAEPS.stars-table-sqlite.dat Stars
```

These quick import routines are not part of SQL so vary from database to database (but they are handy!)

In MySQL for instance it would be:

```
LOAD DATA INFILE 'YAEPS.stars-table-sqlite.dat' INTO
TABLE Stars
  FIELDS TERMINATED BY ‘,’;
```

here I can also add IGNORE 1 LINES at the end if I want to skip a header line.

Getting rid of a table & altering it

Everyone makes mistakes. Sometimes the best is to just get rid of a table. It is easy:

```
DROP TABLE Galaxy;
```

Getting rid of a table & altering it

Everyone makes mistakes. Sometimes the best is to just get rid of a table. It is easy:

```
DROP TABLE Galaxy;
```

It is also possible to modify (alter) a table - but not that this does not fully work in sqlite!

```
ALTER TABLE Galaxy ADD rmag  FLOAT
```

Adding column

```
ALTER TABLE Galaxy DROP COLUMN rmag
```

Deleting column

```
ALTER TABLE Galaxy ALTER rmag DOUBLE
```

Changing the type of a column

Putting data into the database - row by row

Adding data one row at a time is done using INSERT:

```
INSERT INTO Galaxy VALUES (1,
12.334, 14.433);
```

GalaxyI ra decl

GalaxyI	ra	decl
1	12.334	14.433

You can also insert only some values but then you have to say what columns they are for:

```
INSERT INTO Galaxy (GalaxyID,
decl) VALUES (2, 17.5);
```

GalaxyI	ra	decl
1	12.334	14.433
2	NULL	17.5

Note: You can also insert multiple rows if you copy from one table to another.

Putting data into the database - in one go

INSERT is quite slow, in part because the database is reorganised after each insert. This is fine for small jobs but not for 100,000s of entries. For this case we use LOAD DATA.

```
LOAD DATA INFILE <filename> INTO TABLE Stars
  FIELDS TERMINATED BY ',' IGNORE 1 LINES;
```

(optional)

will load from a file where each column is separated by a comma (the default is TAB), and rows by newline but it will skip the first line. The column types must match the Table definition - so in this case a file that can be loaded would be:

```
# StarID FieldID Star Ra Dec     g    r
1,1,S1,198.8475000,10.5034722,14.5,15.2
2,1,S2,198.5654167,11.0231944,15.3,15.4
```

This is fine for those situations where your data are already known - for instance if you downloaded a catalogue.

Updating data

Most astronomical data can change (calibration files can be updated, measurement techniques improved etc.)

We often then want to create a new table, but sometimes you want to update a row instead. This is done in SQL using the UPDATE command.

```
UPDATE Galaxy SET ra=11.3 WHERE decl=17.5
```

This is ok, in particular where information is acquired after most of the table is assembled.

Next: A more fancy criterion

2. How many stars have $0.1 < g-r < 0.4$?

```
SELECT *
FROM Stars
WHERE g-r BETWEEN 0.1 AND 0.4
```

or

```
SELECT *
FROM Stars
WHERE g-r > 0.1
  AND g-r < 0.4
```

Try this now for your newly created table - remember to put a ; at the end of each line!

Reminder:

```
# StarID FieldID  Star     Ra          Dec         g       r
    1         1      S1   198.8475000 10.5034722 14.5    15.2
    2         1      S2   198.5654167 11.0231944 15.3    15.4
    3         3      S5   198.9370833  9.9168889 16.4    15.8
    4         2      S7   199.2516667 10.3486944 14.6    14.1
```

Schema

Column	Type	SQL type	Other
StarID	Integer	INT	PrimaryKey, Unique
FieldID	Integer	INT	ForeignKey
Star	String	varchar(10)	Length < 10
Ra	Real number	DOUBLE	
Dec	Real number	DOUBLE	
g	Real number	DOUBLE	
r	Real number	DOUBLE	

Create a new table - now for the Observations

```
# FieldID  Field       Date      Exptime   Quality          WhereStored
   1       StF-043  92.9885764    23.2        1       /disks/yaeps-1/StF-043.fits
   2       StF-044  97.3323764    30.2        1       /disks/yaeps-1/StF-044.fits
   3       StF-045  93.5532134    29.5        0.5     /disks/yaeps-1/StF-045.fits
```

Column	Type	SQL type	Other
FieldID			
Field			
Date			
Exptime			
Quality			
WhereStored			

In practice:

Get YAEPS.observations-table-sqlite.dat and sqlite3-make-observations-table.sql from Blackboard. **Edit** the latter to reflect the location of YAEPS.observations-table-sqlite.dat

```
> sqlite3 MLD19.db
SQLite version 3.8.10.2 2015-05-20 18:17:19
Enter ".help" for usage hints.
sqlite> .read sqlite3-make-observations-table.sql
sqlite> .tables
Observations  Stars
```

sqlite from Python

or - what you really want to know

sqlite3 in python - an example

```
import sqlite3 as lite;
```

Load what is necessary

The database must be created first!

```
con = lite.connect(database)
```

Connect to database

```
with con:

    # Get a cursor.
    cur = con.cursor()

    # Execute commands
    cur.execute(command)
```

Use with to gracefully handle exceptions

cursors are used to navigate relational databases and are often needed in programatic access

Building the table in python:

```
# Next, we create a connection to the database.
con = lite.connect(database)


with con:

     table = ‘Stars’
    # Create the command to create the table. I use a
    # multiline string to ease readability here.
    command = """CREATE TABLE IF NOT EXISTS {0} (StarID INT,
            FieldID INT, Star varchar(10), ra DOUBLE,
            decl DOUBLE, g FLOAT, r FLOAT,
            UNIQUE(StarID), PRIMARY KEY(StarID),
            FOREIGN KEY(FieldID) REFERENCES Observations(ID))""".format(table)

    # Next, actually execute this command.
    con.execute(command)

    # Now that this is working, let us loop over the table entries
    # and insert these into the table.
    for row in cat:
        command = "INSERT INTO Stars VALUES({0},{1},'{2}',{3},{4},{5},
{6})".format(row[0], row[1], row[2], row[3], row[4], row[5], row[6])
        print command
        con.execute(command)
```

See the GitHub site for the script - now build one for the observations

Using python to query the database:

```
In [1]: import sqlite3 as lite;

In [2]: con = lite.connect('MLD19-python.db')

In [3]: rows = con.execute('SELECT ra, decl FROM Stars')

In [4]: for row in rows:
   ...:     print "Ra={0}  Dec={1}".format(row[0], row[1])
   ...:
Ra=198.8475  Dec=10.5034722
Ra=198.5654167  Dec=11.0231944
Ra=198.9370833  Dec=9.9168889
Ra=199.2516667  Dec=10.3486944
```

As should be clear: The execute statements executes SQL statements in the database and returns a list of results.