Foundation of Data-Driven Analysis for Policy Decisions Course (Day 1 - Linux Environment)

Hanjo Odendaal

LEAD DATA SCIENTIST (71POINT4)

ABOUT ME

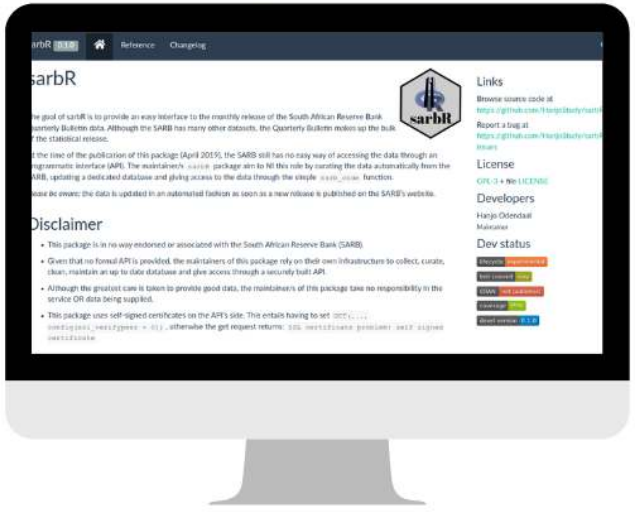
I lead the advanced data analytics and statistical modelling aspects of the work at 71point4. I am passionate about exploring different methodologies to collect and analyse new and alternative data sets.

I hold a PhD in Economics from the University of Stellenbosch: News, Sentiment and the Real Economy.

Hanjo Odendaal

LEAD DATA SCIENTIST (71POINT4)

Software Engineering



High Performance Cloud Computing

Production Machine Learning

Practical Machine Learning

H_2O.ai

Web Scraping

useR!

Agenda

1) About this Course
2) Understanding Data Science Teams
3) Software Requirements

About this Course

What this course aims to achieve

What the course aims to achieve:

> On completion of the course, participants will understand common 'data science' terminology and develop basic technical skills in three key areas: (1) System Administration, (2) Database Design and (3) Statistical Programming.

- Very few organizations need machine learning engineers, but all of them need a data team that communicates effectively and have the necessary skills to perform basic data tasks. Getting teams to understand the broader problem each department faces solves 80% of the frictions encountered when delivering insight from data.

What the course does **NOT** aim to achieve:

> It will NOT turn individuals with varying backgrounds, skills and motivations into fully-fledged Data Scientists. Also, note that the course does NOT cover topics related to cyber security!

- We wish to elevate people's knowledge and exposure to basic data science principles to help guide them on their data journey.

Key outcomes

More detailed outcomes will be stipulated at the start of each session, but there are certain core competencies that we want to see in individuals in order to be accredited with passing the course. You should:

- Be comfortable navigating using the command line in a *Linux* terminal.
- Be able to query a database and do *basic* aggregations.
- Document your analytical process using Rmarkdown and Rstudio.

We also encourage the following behaviour throughout the course:

- Learn from each other and share knowledge in groups.
- Ask questions during the course - the instructor has a lot of knowledge that you should tap.
- Arrive on time and complete all of the assignments.

Session Breakdown: Day 1 - Linux Environment

Session 1 (08:30 to 11:00) 🤦:

- Course introduction.
- Understand the different Data Science roles.
- Install software for course.

Session 2 (11:15 to 12:30) 🤦 & 💻:

- *I am because we are* - Welcome to Ubuntu.
  - Why do we use Linux operating systems for software development and analytics?
- Home is where `127.0.0.1` is - getting comfortable with a black screen terminal.
  - `whoami` - creating a user profile.

Session 3 (14:00 to 17:00) 💻:

- Navigating the matrix by getting familiar with the cornerstone functions.
  - `cd`, `mkdir`, `ls`, `find`, `less`, `mv`, `cp`, `rm`
- Editing files using `VIM`.
- Learning about *piping* in `cli`.
- Put your first script into production using `crontab`.

Session 4: (17:00 - 18:00) 👷:

- Attempt the exercises.

Session Breakdown: Day 2 - Databases

Session 1 (08:30 to 09:30) 👨‍💼:

- Going through exercises.

Session 2 (9:30 to 10:30) 💻:

- Install Rstudio.
- `netstat` command for network monitoring.

Session 3 (10:45 to 12:30) 👨‍💼 & 💻:

- Using `Rstudio` as a lab-book:
  - What is `Rstudio`?
  - Learning the basic markdown commands to document code.
  - *Knitting* your first lab-book.
  - Documenting your first bit of code.

Session 4 (14:00 to 17:00) 👨‍💼:

- A bunch of Excel files isn't a Database!
  - Why do we use databases?
  - What type of database should you be using?
  - What are the basics around database, table and view creation?
  - Loading data into a database table.
- Writing your first `SQL` query.

Session 5: (17:00 - 18:00) 👨‍🔧:

- Attempt the exercises.

Session Breakdown: Day 3 - Policy Analysis

Session 1 (08:30 to 09:30) 👨‍💼:

- Going through exercises.

Session 2 (09:45 to 12:30) 💻:

- Analysing the Rwandan housing market using *scraped* data:
  - Creating the database table structure.
  - Building a data-schema in draw.io to visualize relationships.
  - Building a data dictionary for the scraped data set
  - Uploading the information into the database.
  - Data Cleaning in SQL.

Session 3 (14:00 to 17:00) 💻:

- Answering key economic and policy questions using SQL:
  - Creating a derived field.
  - Which areas have the highest franc/sqm?
  - Has priced increased over time?
  - What premium is there for an extra bedroom in Kigali vs Outside of Kigali?
- Writing the analysis up in Rmarkdown.

Session 4: (17:00 - 18:00) 👨‍💻:

- Attempt the exercises.

Session Breakdown: Day 4 - Tidyverse Intro

Session 1 (08:30 to 09:30) 👨‍💻:

- Going through exercises.

Session 2 (09:45 to 11:30) 👨‍💻:

- Introduction to R and the tidyverse
  - How does R differ from python?
  - Learning data-structures.
  - How to read data into R.

Session 3 (11:45 to 12:30) 👨‍💻:

- Introduction to R and the tidyverse
  - Learn how similar it is to SQL.
  - Basic `dplyr`.

Session 4 (14:00 to 17:00) 💻:

- Getting to grips with the grammar of data analytics: `dplyr`.
  - Reproduce outputs from SQL analysis in R using `dplyr`
  - Plotting outputs using `ggplot`.

Session 5: (17:00 - 18:00) 👨‍🔧:

- Attempt the exercises.

Session Breakdown: Day 5 - Reproducible Research

Session 1 (08:30 to 09:30) 👨‍💼:

- Going through exercises.

Session 2 (09:45 to 12:00) 💻:

- Putting all of the analysis together.
  - Using Rmarkdown to write a report on the housing market.

Session 3 (12:15 - 12:45) 👨‍💼:

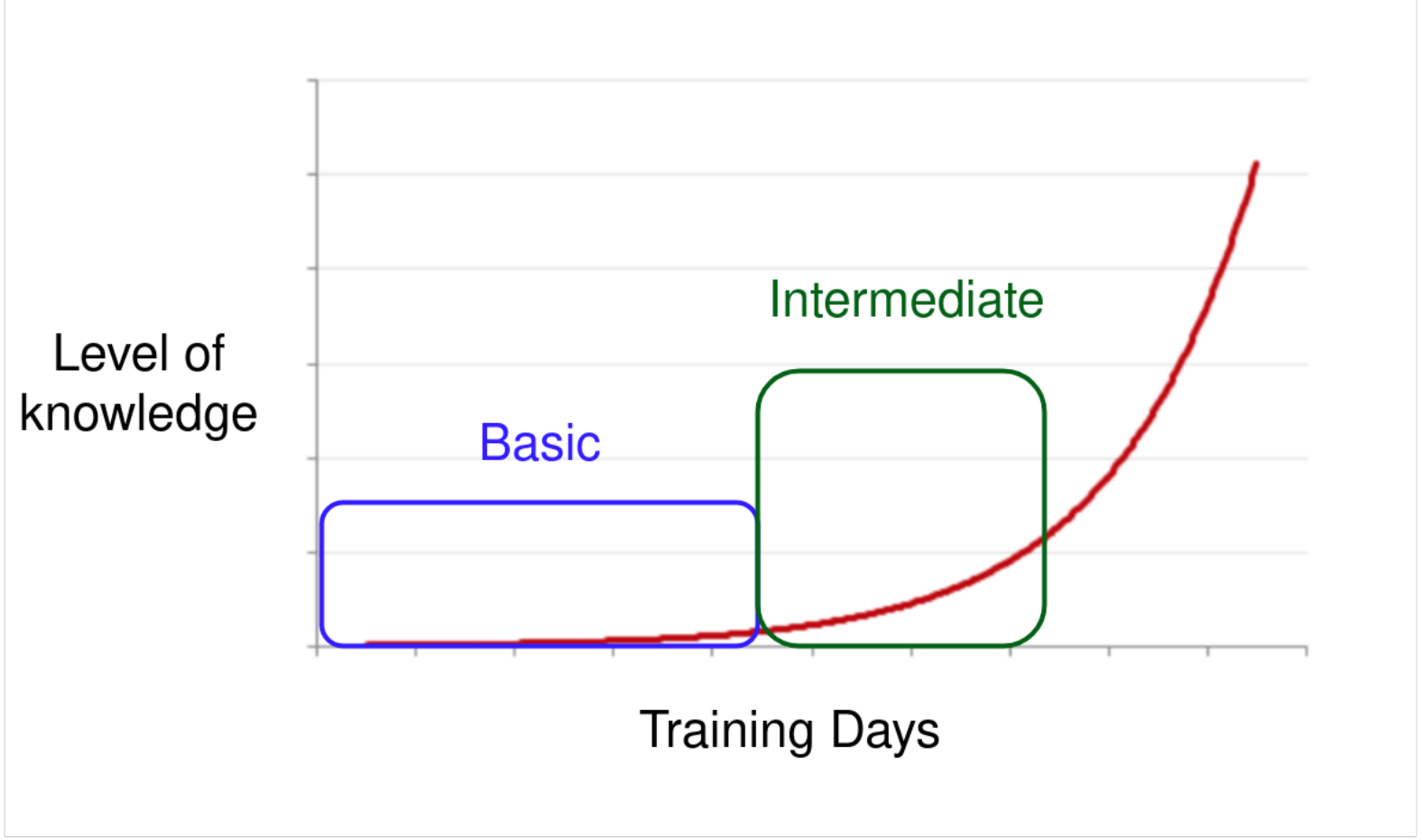
- Introduction of the different Advanced courses:
  - (lv.2) System hardening (cybersecurity), networking and administration.
  - (lv.2) Advanced Database for Big Data (Optimization and OLTP vs OLAP).
  - (lv.2) R and python for statistics and research.
  - (lv.3) Building and deploying production dashboards using R Shiny, APIs and Docker.

Asking for assistance

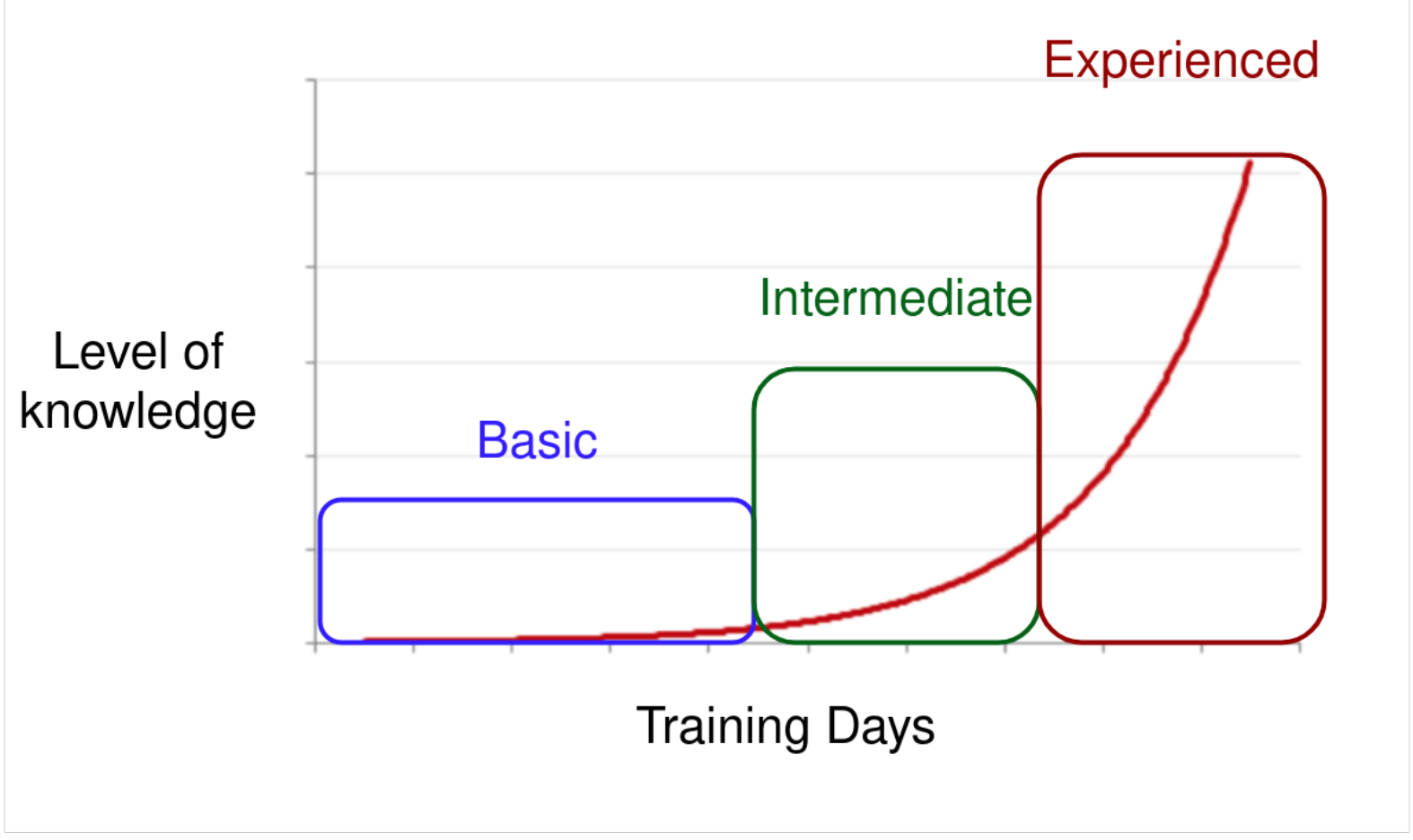
Self-assessment

Self-assessment

Self-assessment



Self-assessment



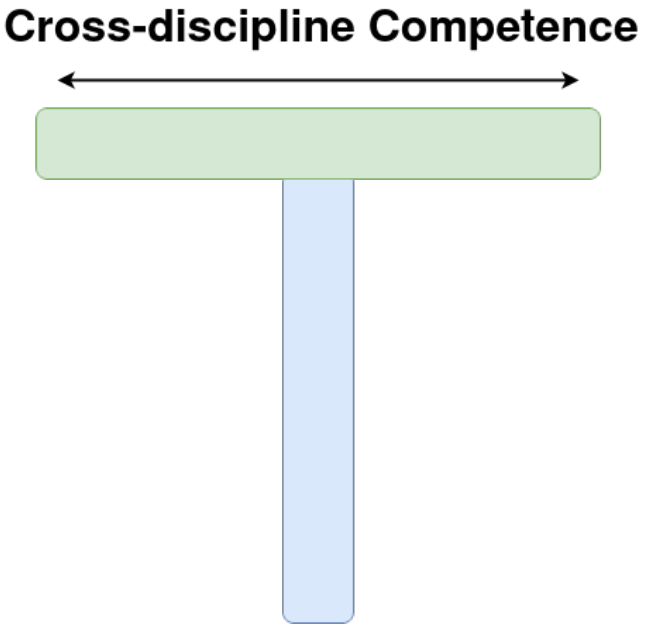
Understanding DS Teams

T-shaped Data Science Teams

> Over the last few years data science has evolved into a multidisciplinary field with specific specialist roles becoming more important.

With this shift, companies are more and more looking to hire "T"-shaped individuals to join their analytics team.

- Cross-discipline looks to ensure that the team speaks the same language when attempting something new or solving a problem.

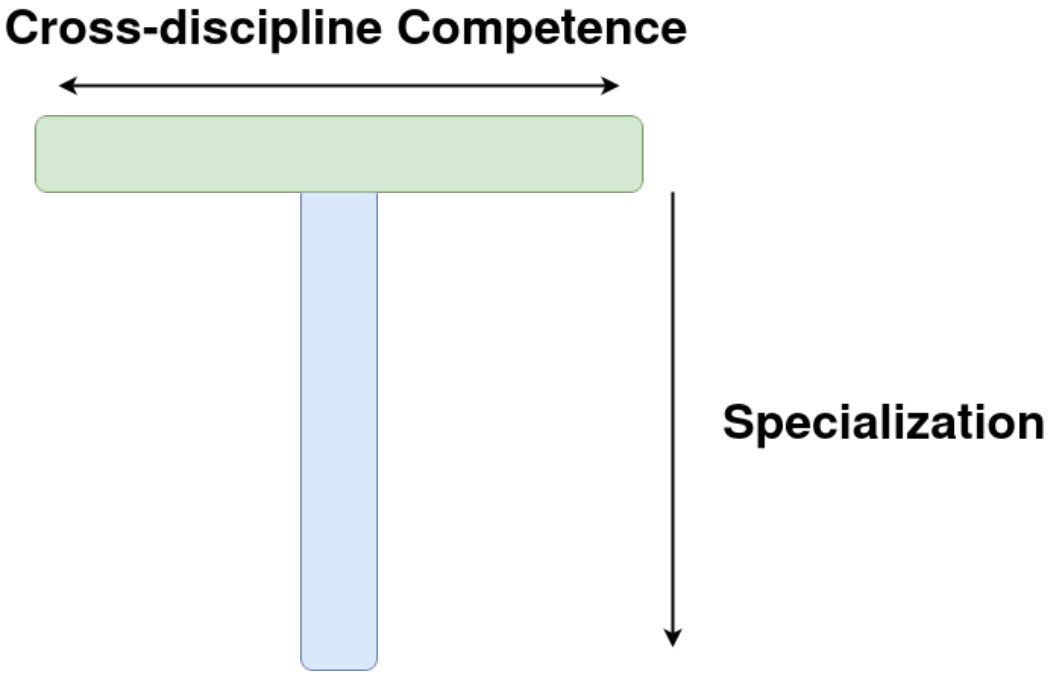


T-shaped Data Science Teams

> Over the last few years data science has evolved into a multidisciplinary field with specific specialist roles becoming more important.

With this shift, companies are more and more looking to hire "T"-shaped individuals to join their analytics team.

- Cross-discipline looks to ensure that the team speaks the same language when attempting something new or solving a problem.
- Specialization helps to resolve difficult problems that cause bottlenecks in the pipeline as fast as possible.



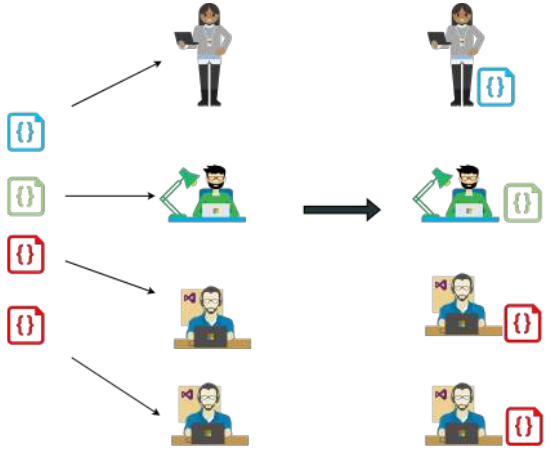
Analytical process

As a data science team, task will most likely be divided up into two main components:

- Immediate need of client/manager.
- The things we would like to be doing.

Deciding how to divide up the work is known as *demand-leveling* and the traditional approach is to balance resource and *demand* based on availability of the team.

- As we can see by the graph, a new task can only be assigned to a team member, once that task has been completed.



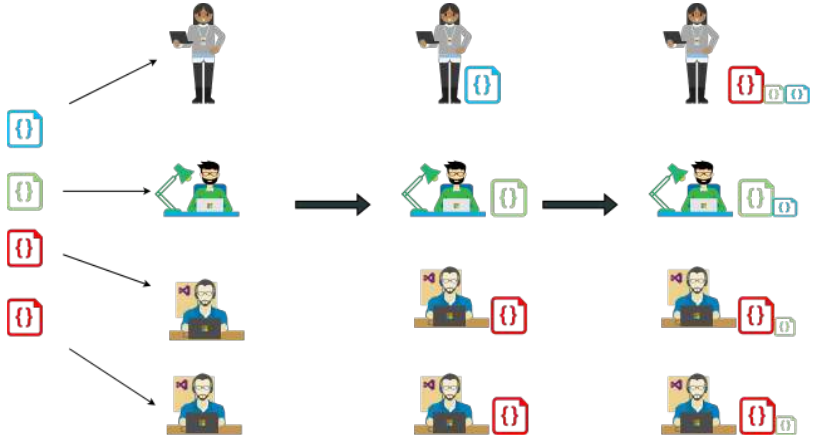
Analytical process

As a data science team, tasks often fall into one of the following two categories:

- Tasks that serve the immediate needs of clients/managers.
- The things we would like to be doing.

Deciding how to divide up the work is known as *demand-leveling* and the traditional approach is to balance *resources* and *demand* based on availability of the team.

- In I-shaped teams, specialization results in a silo-ing of tasks required to solve problems
  - Can lead to task backlogs.
- T-shaped people effectively increase the number of available resources.
  - Affords a more adaptable response underpinned by the prioritization of tasks.



Advantages of T-shaped Teams

> Use experts to solve bottlenecks and non-experts to provide additional support.

By having a data science team of individuals that are T-shaped, experts can offload menial tasks to non-experts that have the required basic knowledge.

- This frees up the expert to solve the bottleneck much quicker.
- Even partial knowledge on how to solve a bottleneck is more valuable than having an expert work on a **non**-bottleneck problem.
- The combined effort of all individuals working on tasks *together* accomplishes more than a silo-ed approach to task management.

So, besides efficiency what other advantages are there in T-Shaped teams?

Advantages of T-shaped Teams

Communication among team members is more efficient.

- In the process up picking up a skill, you will also start to build domain-specific language. This ensures that the team understands different perspectives.

Team members stay interested in what they are doing.

- What's boring for one person on the team might be an exciting challenge for another member.
- A multidisciplinary approach enables learning and a growth mentality among the team.
  - Essential to avoid the *paradox of expertise*.

You're more attractive to employers.

- We are building careers in a very dynamic sector where new roles and responsibilities emerge all the time. Cross-competencies make for an adaptable resource, a chracteristic that is very valuable to employers.

Moving to a T-shaped DS Team

> Unfortunately, there isn't a universal cheatcheat...

Assess what skills and knowledge that each individual already has.

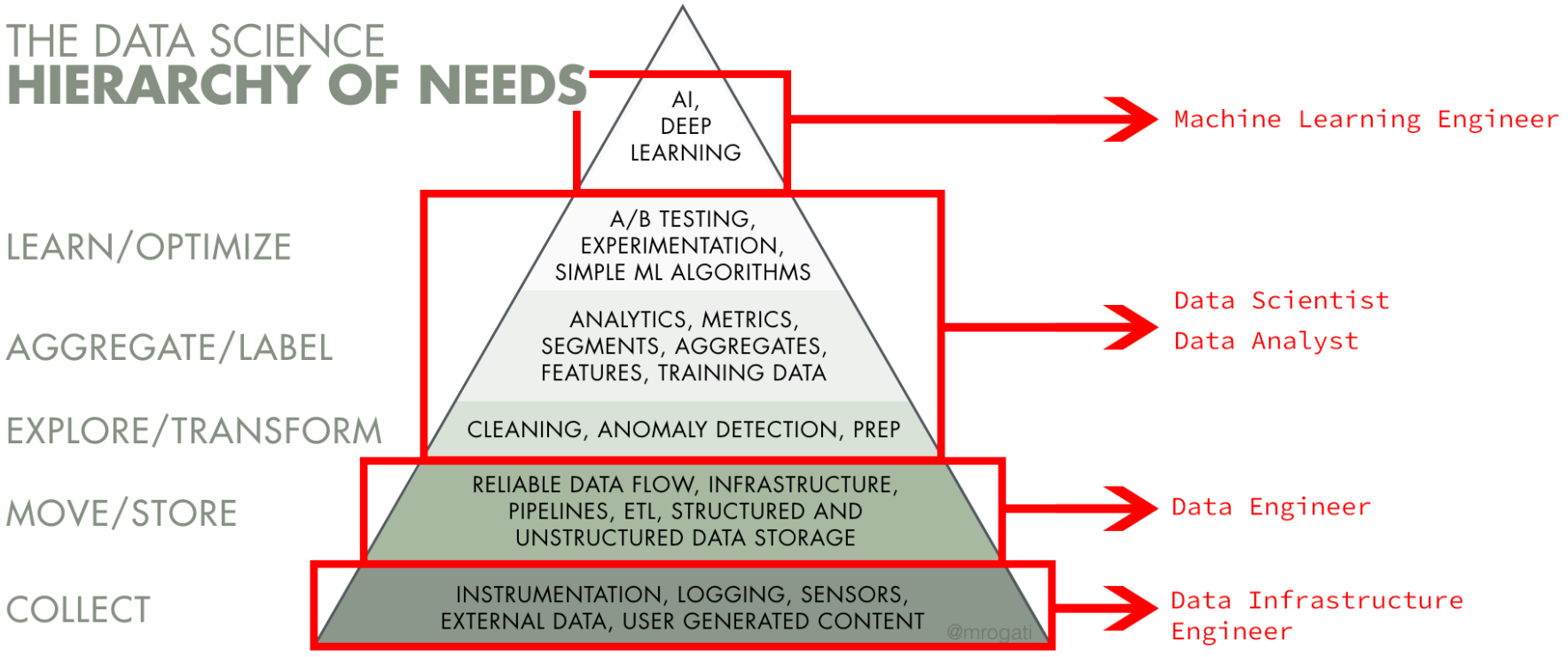
- Rate their ability within these different areas as well as the skills that the team member wants to improve on.

Ensure that work is broken down into incremental outcomes across the work streams.

- This approach ensures that tasks usually require multiple skills.
- Example: creation of a new indicator. This would entail writing `SQL` in a programmatic language (Data scientist), compiling the software bundle (Developer) and putting the new version into production (DevOps).
- This approach also encourages pair-coding which greatly increases knowledge sharing.

What are these expert roles?

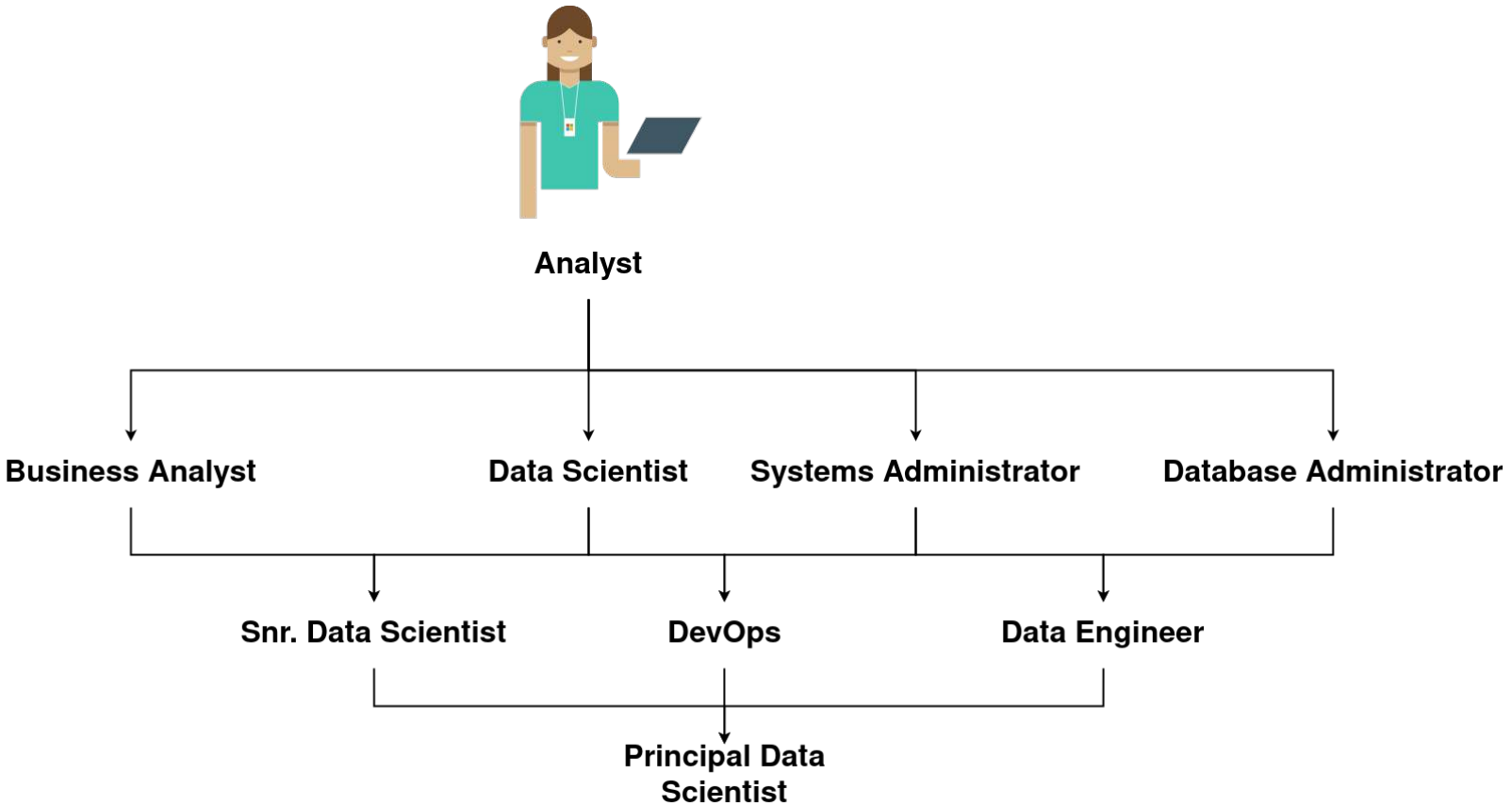
Data science hieracrhy of needs.



*Data science hieracrhy of needs. Created by Monica Rogati

What are these expert roles?

The journey might look a bit different for each individual, but the *general* career path looks like this:



Self-assessment

Analyst (Business Analyst)

> The typical data analyst role is consulting-centric.

Job Description

A data analyst typically works as part of an interdisciplinary team to determine and execute the organization's goals. The analyst mostly gathers data to identify trends that help business leaders make strategic decisions.

Analysts usually have a strong economic or business background that is supported by his/her knowledge of statistics and mathematics. As opposed to Data Scientist, analysts are more generalist and tend to be more flexible in the job market.

Knowledge of

- Excel
- SQL
- Power BI

Responsibilities

- Pull data from databases.
- Analyse and forecast trends.
- Create dashboards using Power BI or Tableau.
- Descriptive, diagnostic, predictive or prescriptive analytics.

Data Scientist

> A data scientist is someone who is better at statistics than any software engineer, and better at software engineering than any statistician.

Job Description

Although it is important for Data Scientists to have a good understanding of business processes, the majority of their work will involve solving complex problems by developing new tools, methods or procedures. While the analyst is closely involved in answering an organization's business questions on a day to day basis, the DS focuses on a macro level to develop ways to meet those business needs. It is important to develop analyses in a structured way so that they can be automated and scaled if the business requires them on a regular basis.

This is a much more specialized role and organisations that fully utilize the data scientist skillset can be hard to find.

Knowledge of

- R
- Python
- Java
- Scala

Responsibilities

- Data cleaning and wrangling (80%)
- Building APIs and ETL pipelines
- Statistical analysis using ML
- Automate processes

Database Administrator

> Contingency is the name of the game.

Job Description

A database administrator, commonly abbreviated as DBA, maintains the integrity and functioning of a database. This position entails running regular diagnostic tests to ensure data is not corrupt and combing for bugs or glitches within the system. Safely storing and backing-up data in case of system failure or memory loss and creating plans for addressing large-scale errors are also important responsibilities of a DBA.

It is important that DBAs works closely with SysAdmins to ensure high availability of servers supporting clusters.

Knowledge of

- Linux
- SQL
- Java
- Python

Responsibilities

- Manage backups
- Capacity planning
- Disaster recovery processes and procedures
- Security
- Index maintenance

Data Engineer

> Overseeing the technical part of data.

Job Description

Data Engineers are usually senior individuals in the organisation with extensive knowledge of data models, databases, IT infrastructure and software engineering.

Your data engineers are responsible for building optimized data flows that can be relied on in every day decision making and operations. To accomplish this, data engineers need experience in building database architecture by allocating data storage, establishing rules for data flow and most importantly, choosing the correct technology stack to run the data pipelines.

Knowledge of

- Linux
- Distributed database systems
- Python/R
- Java
- C++
- Scala
- Airflow

SysAdmin

> When its good, its good, but when its bad, hell hath no fury like a SysAdmin scorned.

Job Description

System administrators (SysAdmin) are benevolent creatures with endless power who make sure your computer and network remain in good working order no matter what the silly data engineers/scientist/ devops do.

By far the most important role if a organisation is to succeed, as the SysAdmin is responsible for keeping the system running in a secure manner no matter what the workload is. SysAdmins are also responsible for configuration management tools so that a system can be restored procedurely if it goes down.

Knowledge of

- Linux CLI tools (awk, sed, jq)
- Perl/Python
- Ansible/ Jenkins etc

Responsibilities

- Be ready for 2am calls. ;-)

DevOps

> DevOps transforms the delivery capability of development and software teams.

Job Description

Development & Operations (DevOps) is a series of practices and processes that are intended to speed up and automate the developing, testing, and releasing of software to allow for the continuous delivery of software and software updates. DevOps are the team that is responsible for putting models, applications, dashboards and APIs into production and orchestrating how each of the pieces of software interact with one another and the public.

Knowledge of

- Linux
- Kubernetes
- Docker
- Python
- CI/CD tools
- Ansible/ Jenkins etc

Installing Solar-Putty

What is Solar-Putty?

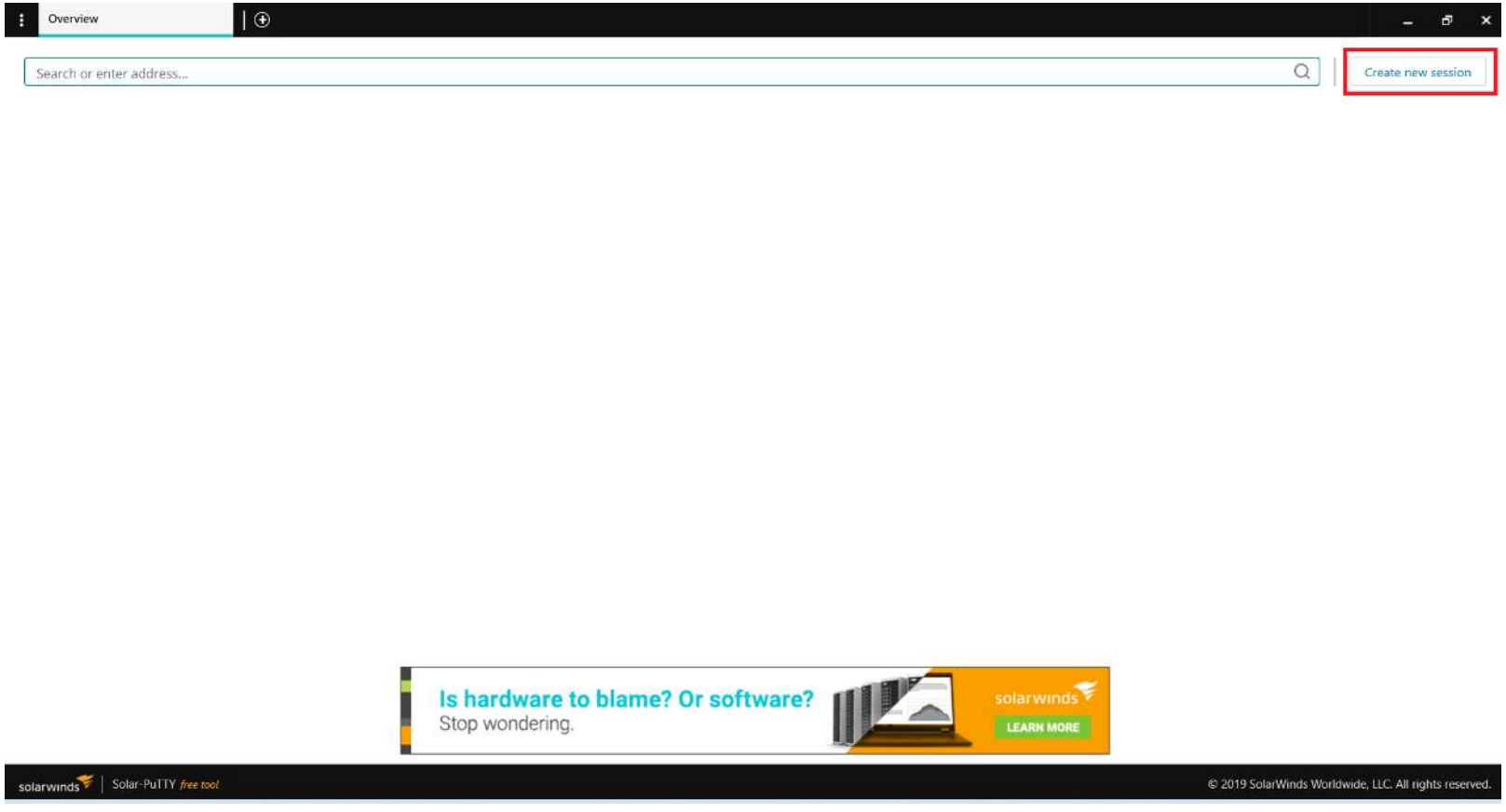
A tool to help us manage remote sessions, which is us logging onto remote servers

Managing remote sessions have never been so easy and comfortable!

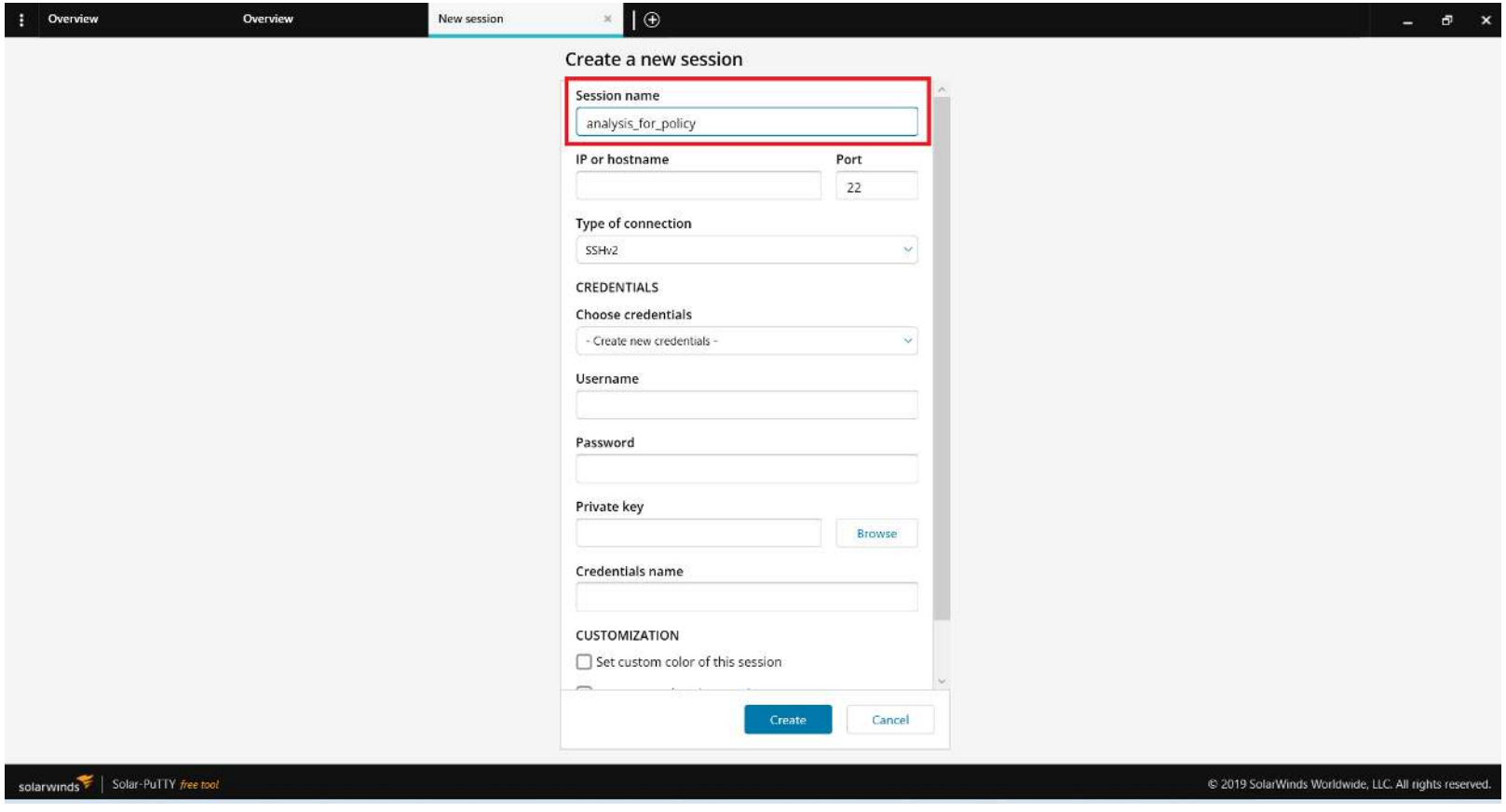
Experience Solar-PuTTY, the SSH client you always wanted

	Solar-PuTTY 100% Free	PuTTY
Support of SCP, SSH, Telnet, SFTP	✔	✔
Saving credentials (including private key) for auto-login	✔	–
Support of multiple sessions in tabbed interface	✔	–
Quick access to the most used sessions	✔	–
Auto-reconnecting capability	✔	–
Graphical SFTP file transfer	✔	–
Support of post-connections scripts	✔	–
Integration of Windows Search	✔	–
	DOWNLOAD FREE TOOL	

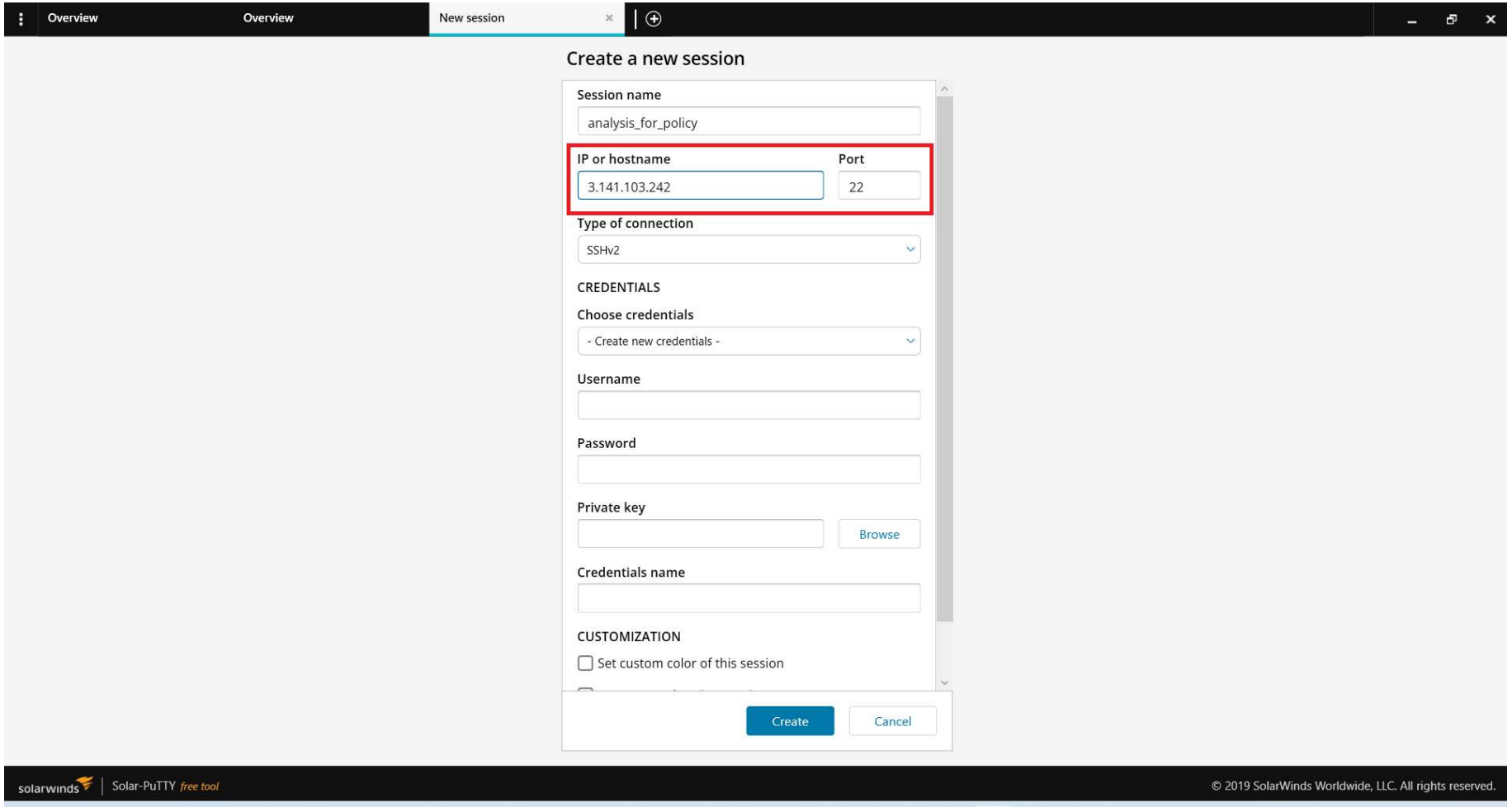
Steps to installation



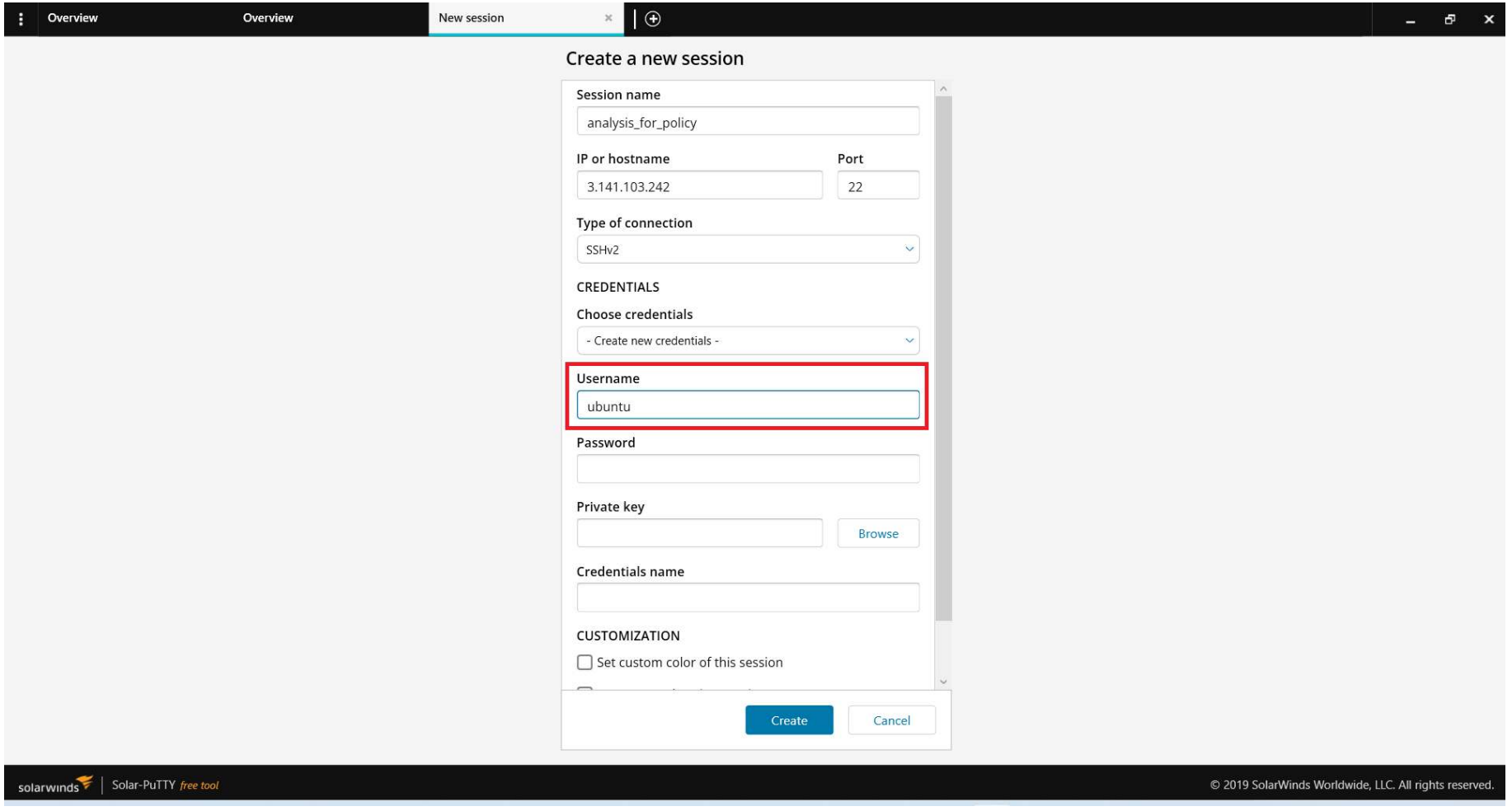
Steps to installation



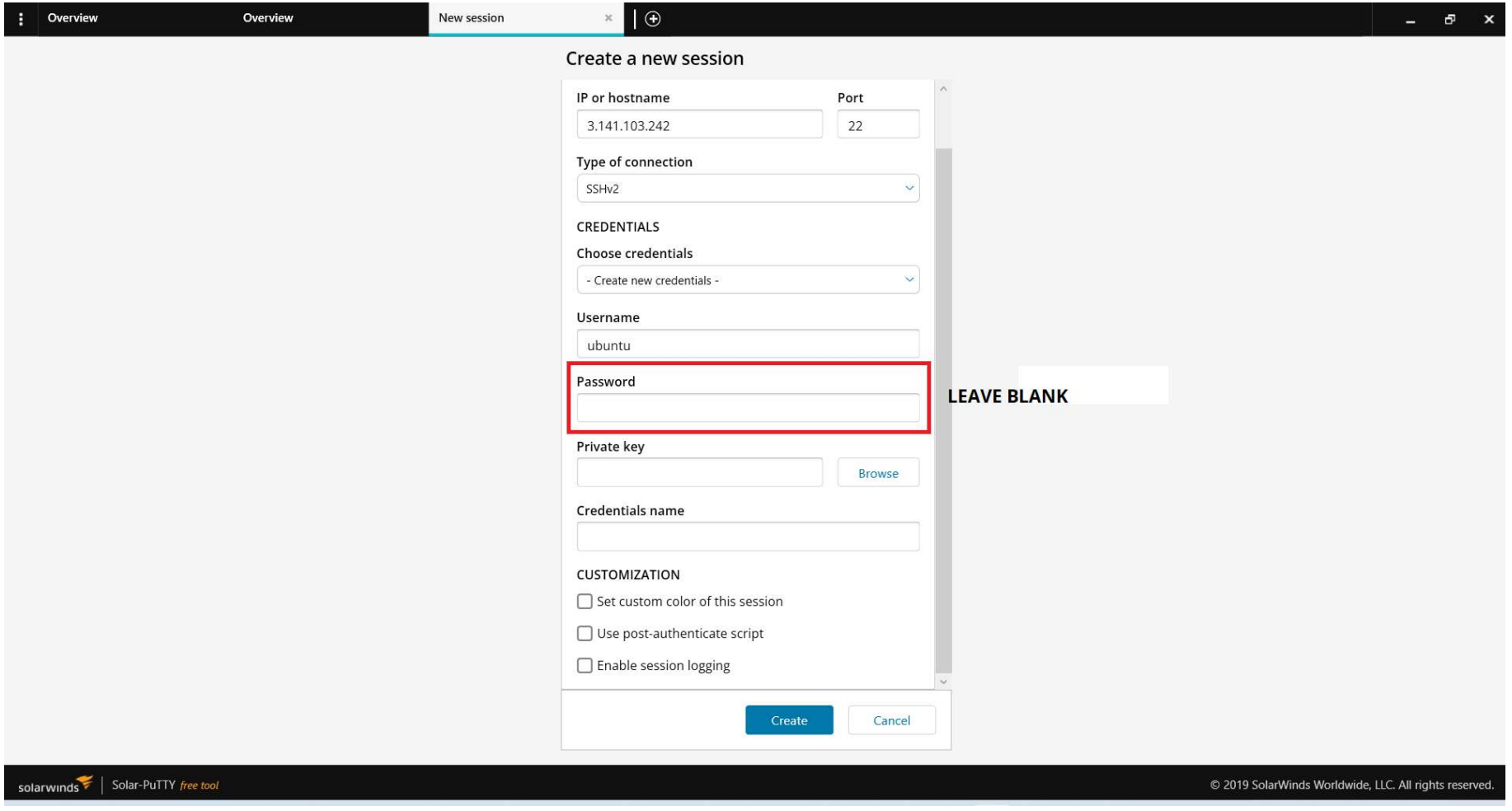
Steps to installation



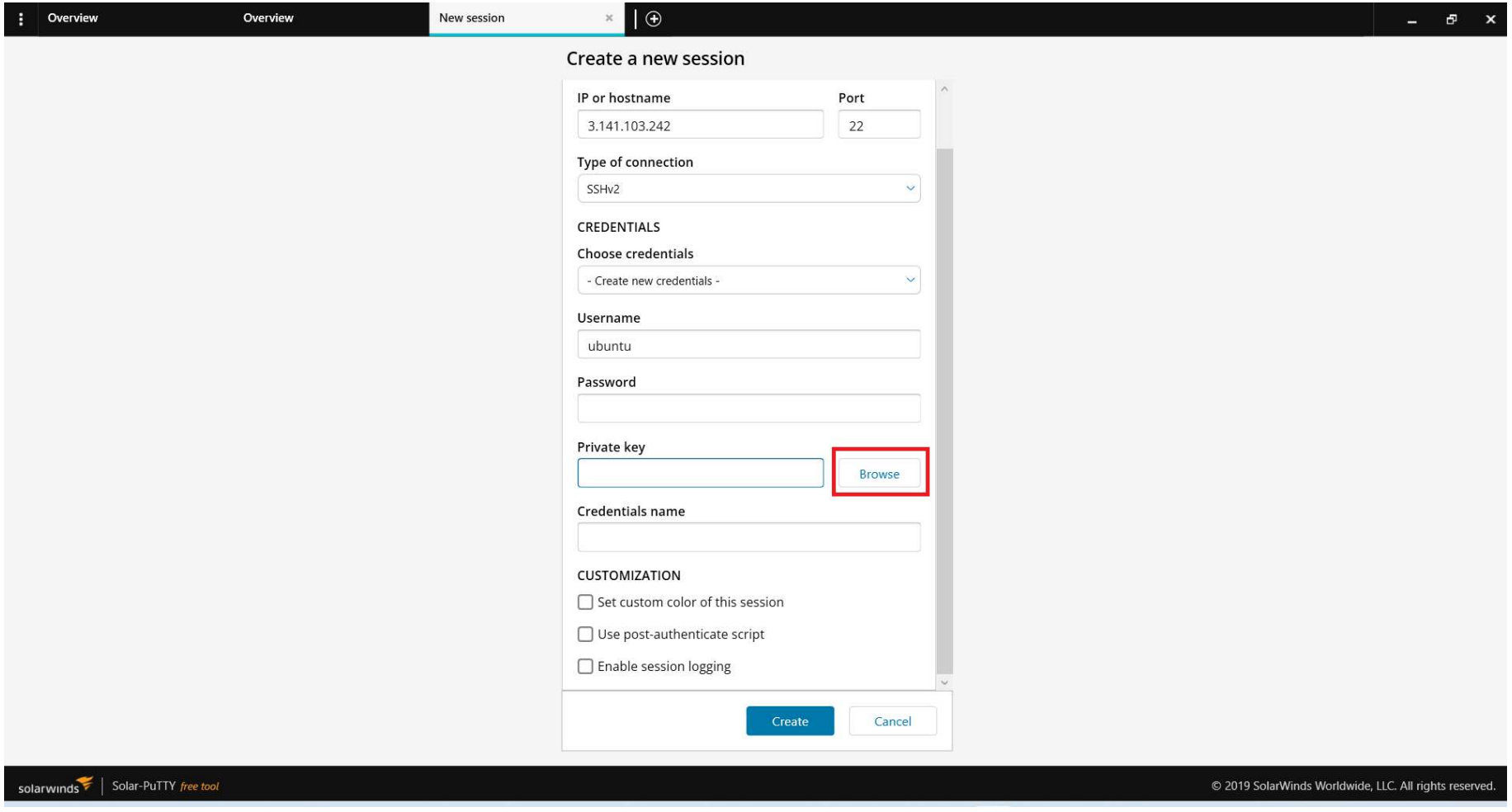
Steps to installation



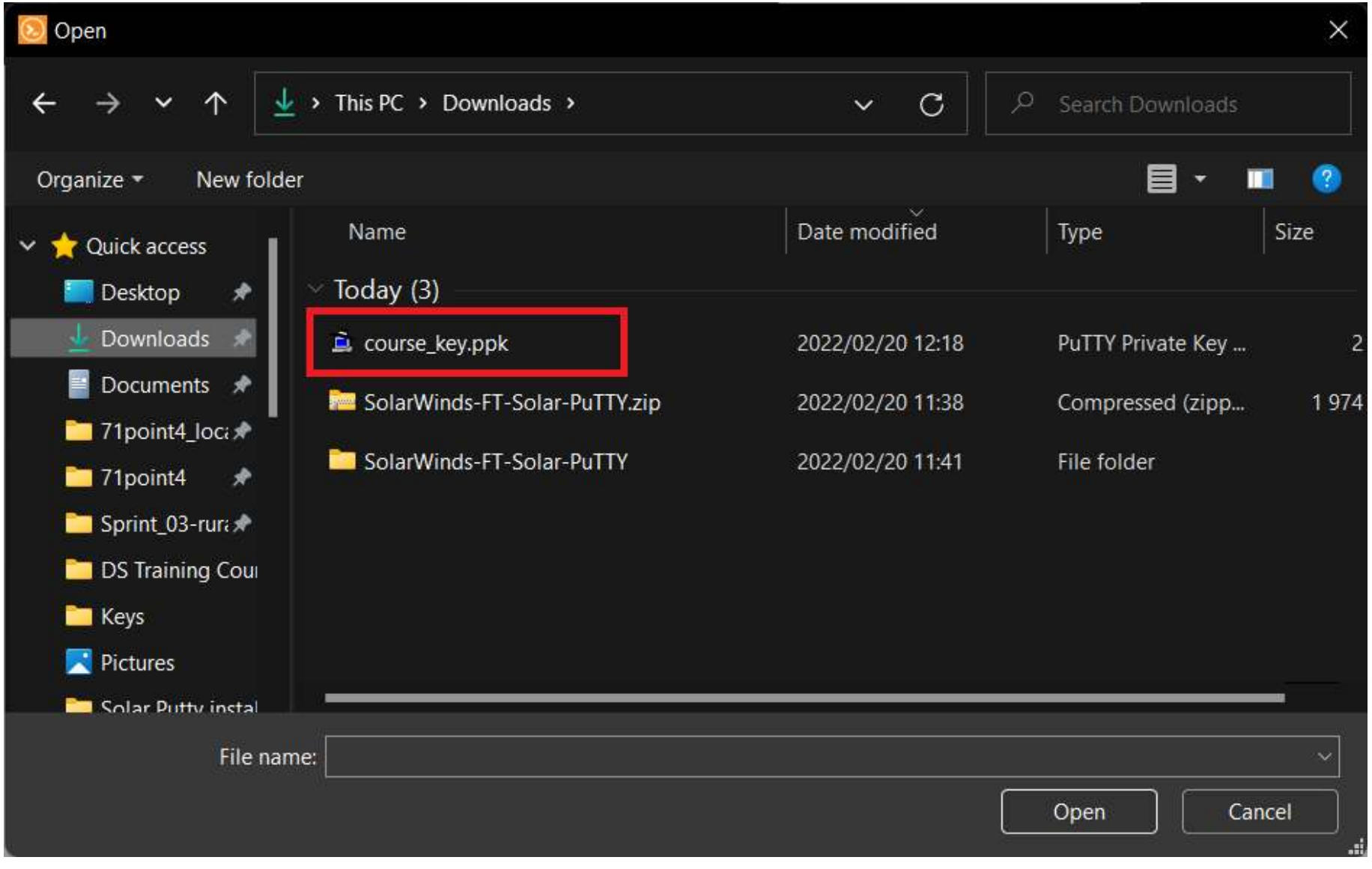
Steps to installation



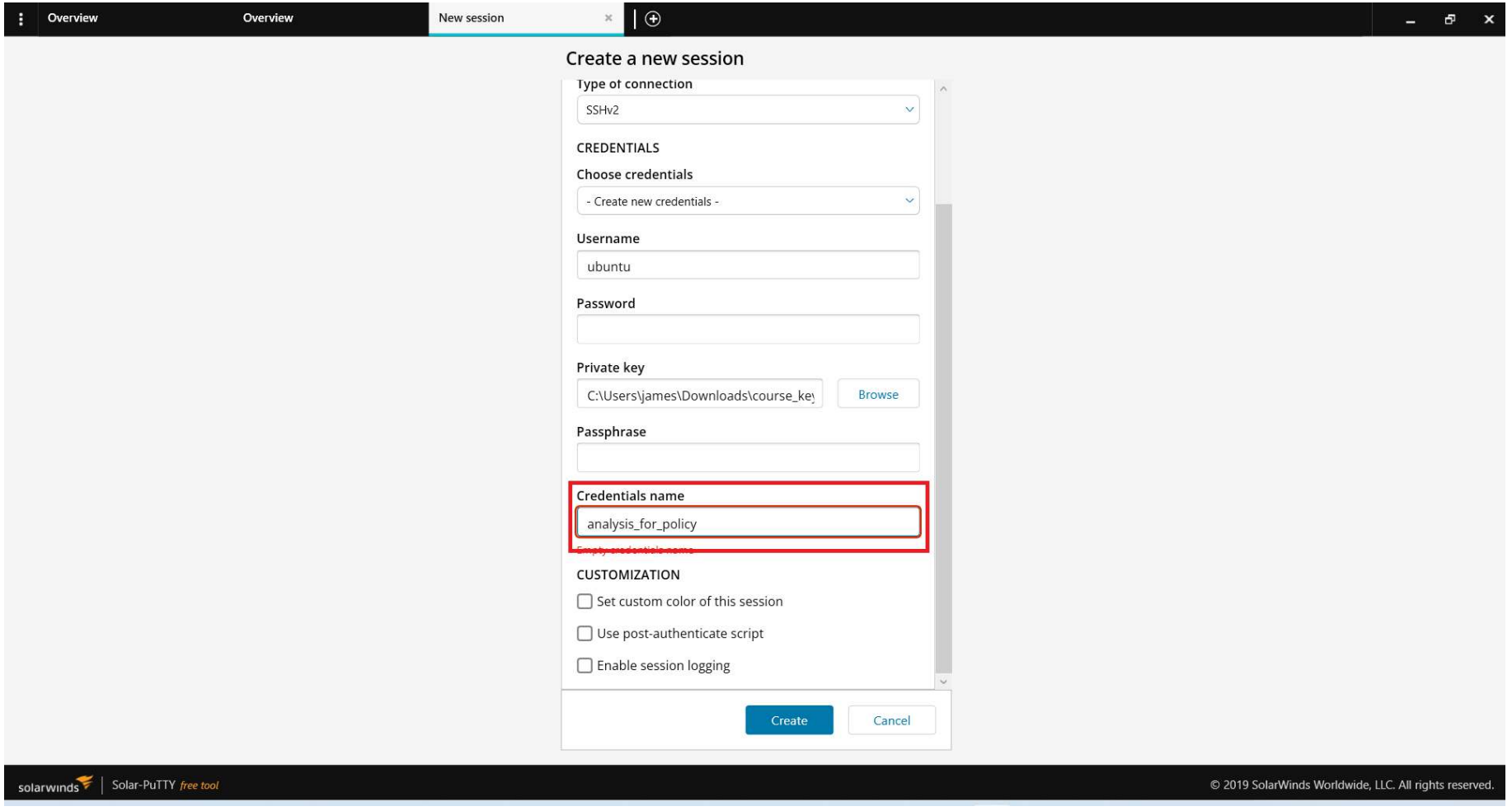
Steps to installation



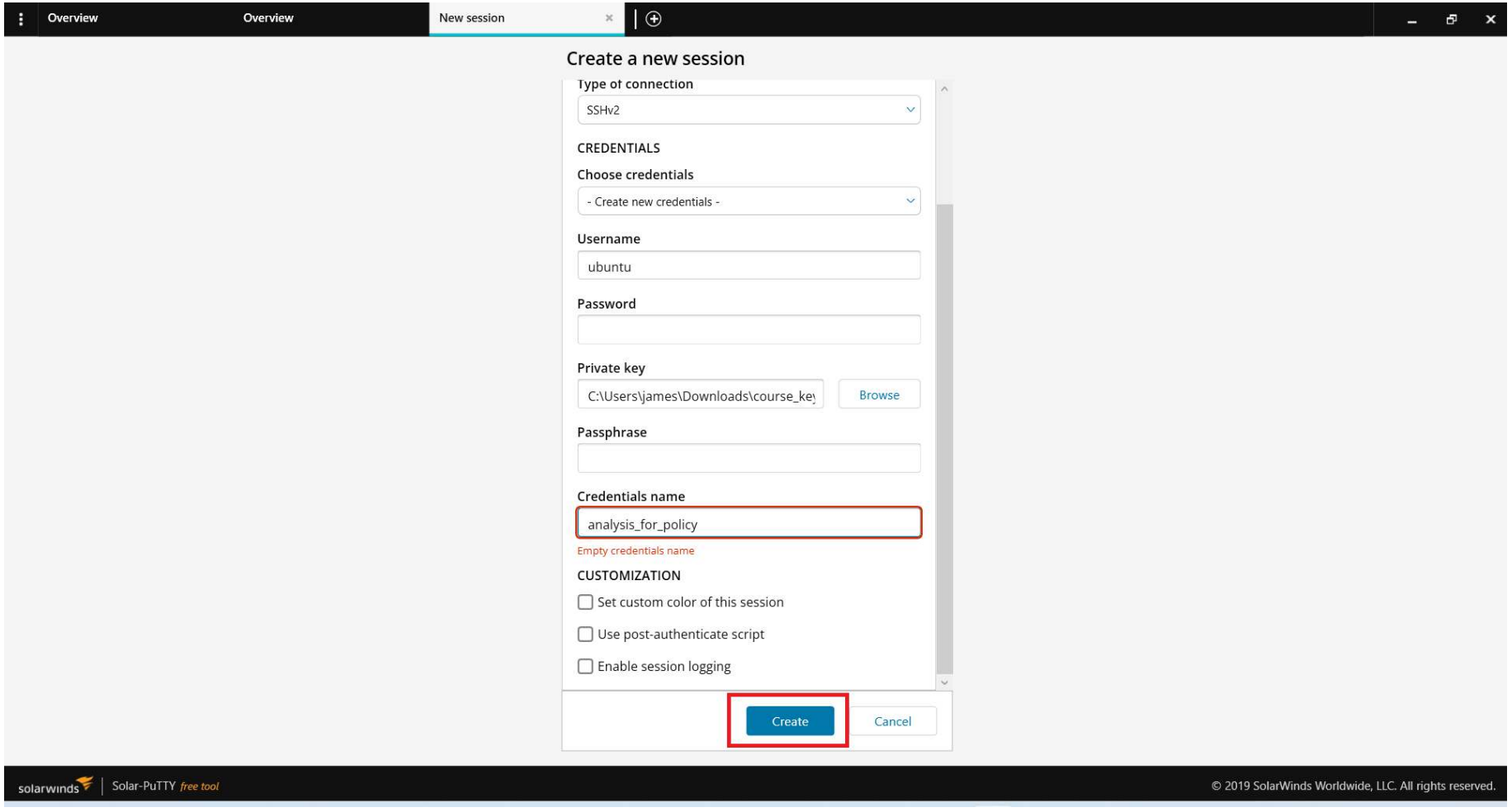
Steps to installation



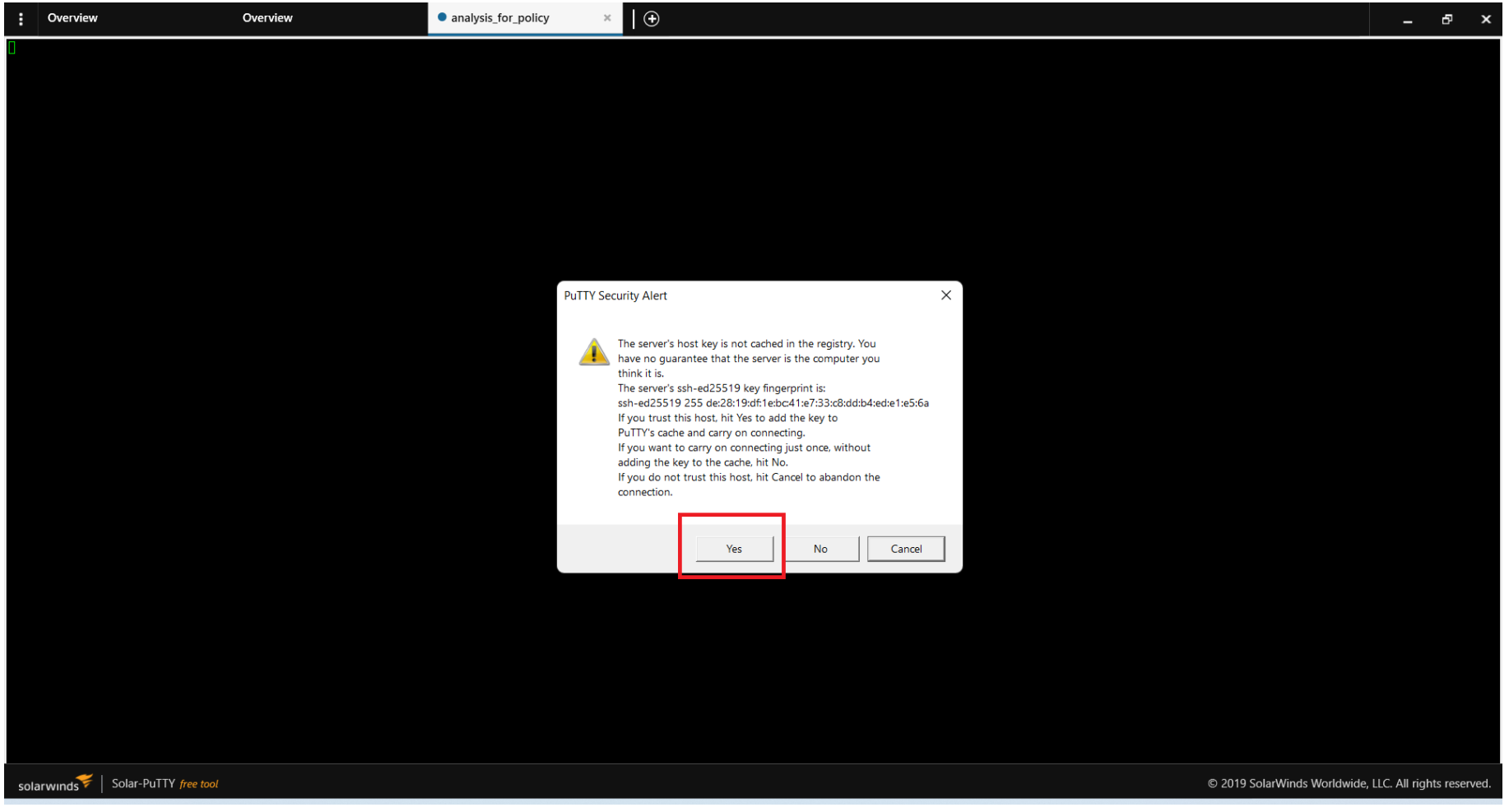
Steps to installation



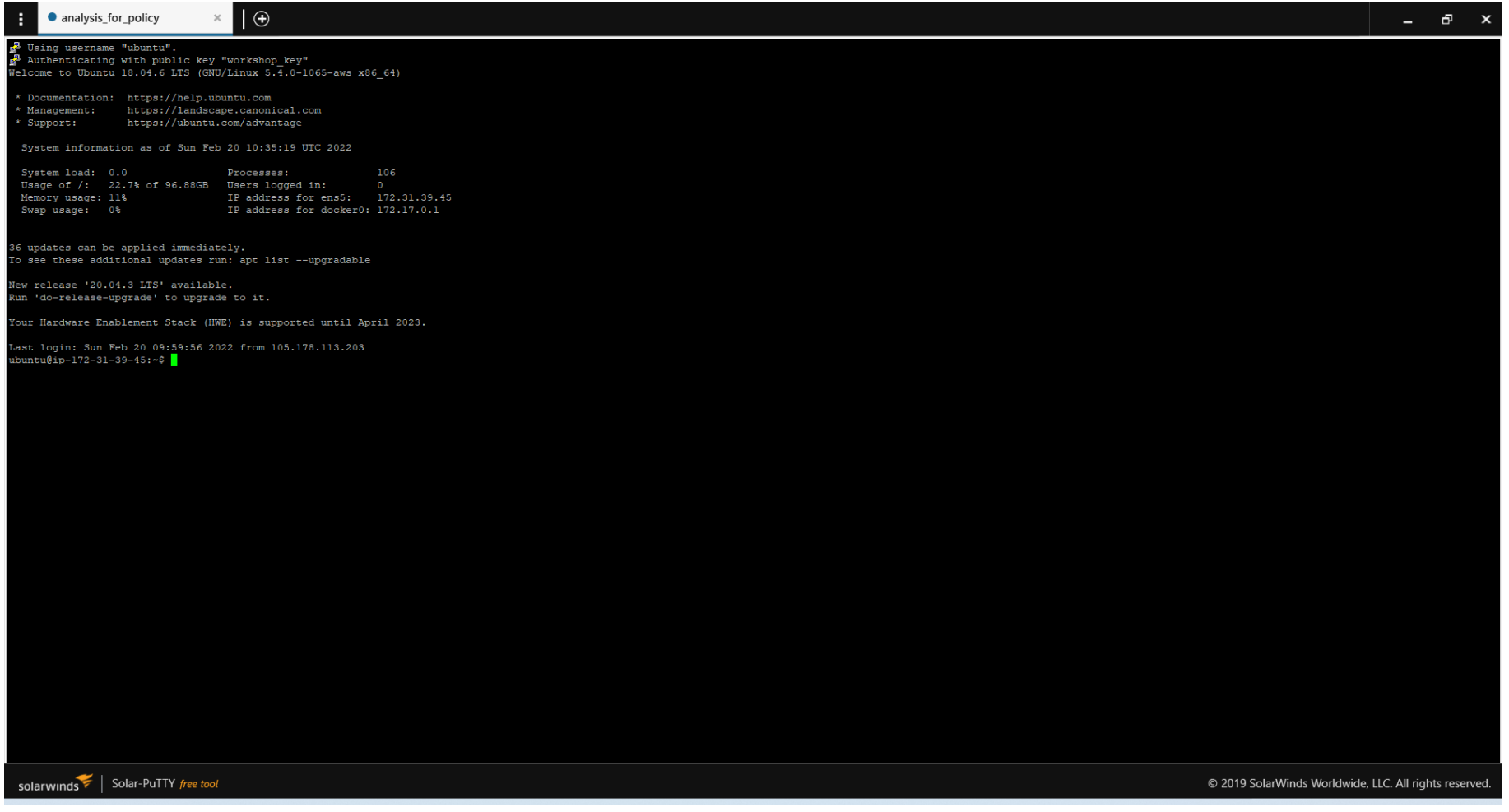
Steps to installation



Steps to installation



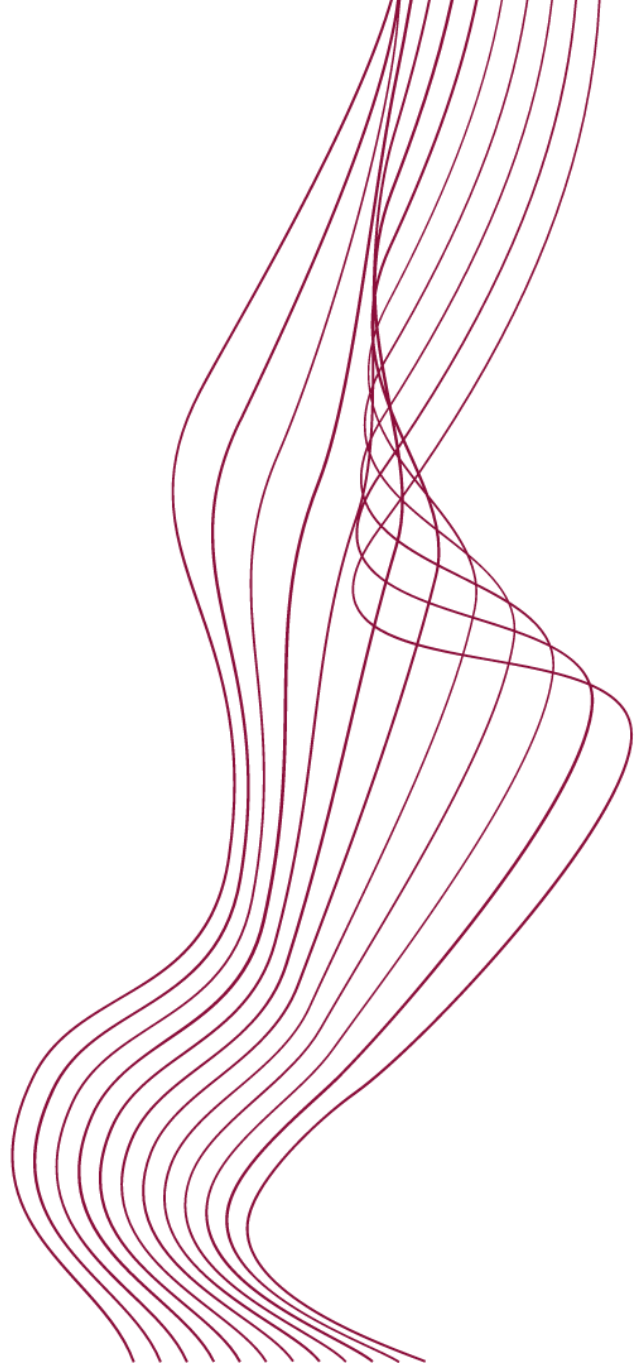
Steps to installation



Welcome to Ubuntu (01 - Session 2)

Agenda

1) Why Linux?
2) Login into Server
3) Creating a profile



Why Linux  ?

Choose your warrior!

http://askyadnesh.blogspot.com/2017/01/

Logging into Server

What is shell?

Whenever we talk about *black screen*, *command line* or *shell* we are essentially talking about the interface that takes input from the keyboard and sends it to the operating system (OS).

Almost all Linux distributions supply a shell program from the GNU Project called `bash` that looks like this:

```
hanjo@optimus:~$ penguin
```

This interface is called *shell prompt* and usually contains `username@machinename:directory`. If the last character of the prompt is a hash mark (`#`) rather than a dollar sign (`$`), the terminal session has superuser privileges (a little bit more on this later).

- Pressing the up ☝ arrow on your keyboard goes into your command history.
  - Be aware that history stores about 1,000 commands.

Creating a user

Different type of users

Superuser (root)

> With great power comes great responsibility!

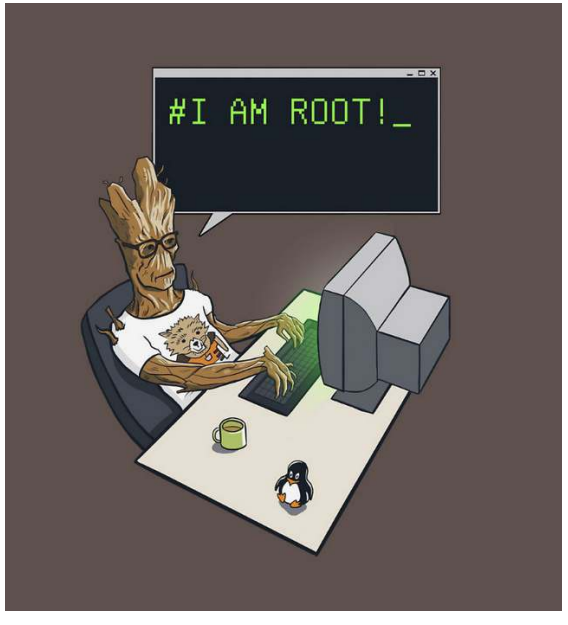
Different type of users

Superuser (root)

> With great power comes great responsibility!

On a Linux system Superuser refers to the root user, who has unlimited access to the file system with privileges to run all Linux commands.

- This responsibility is mostly given to experienced SysAdmins. The reason being there is no "take-backsies" in linux. Once a command has been executed under `sudo` (superuser do) , there is almost never a way to reverse the execution (ex. deleting a file).
- The Superuser/Root is also responsible for setting up security and thus, limiting the power to a single (or very few individuals is preferred).



Different type of users

Superuser (root)

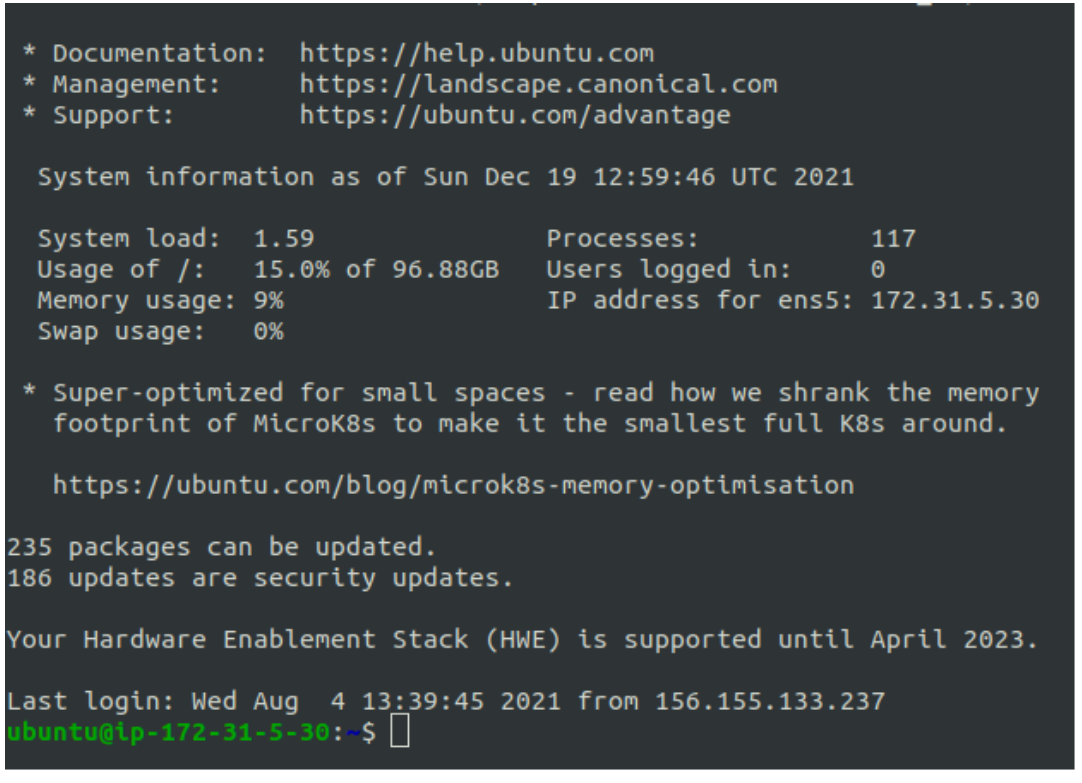
> With great power comes great responsibility!

You will need `sudo` access to execute some of the commands during this course.

- One of the key responsibilities of SysAdmin is to install the necessary software. I will show you how one can do this in a safe manner.
- Once we have created a dedicated user, we will give that user `sudo` privileges.
  - Requires you to type in a password before executing a command that changes system configuration.

Creating your own user account

- When you first log into the machine via ssh, you should see something similar to the picture below:



Basic shell commands

Try these basic commands:

```
date
```

```
## Thu 30 May 2024 15:17:52 SAST
```

```
free -h
```

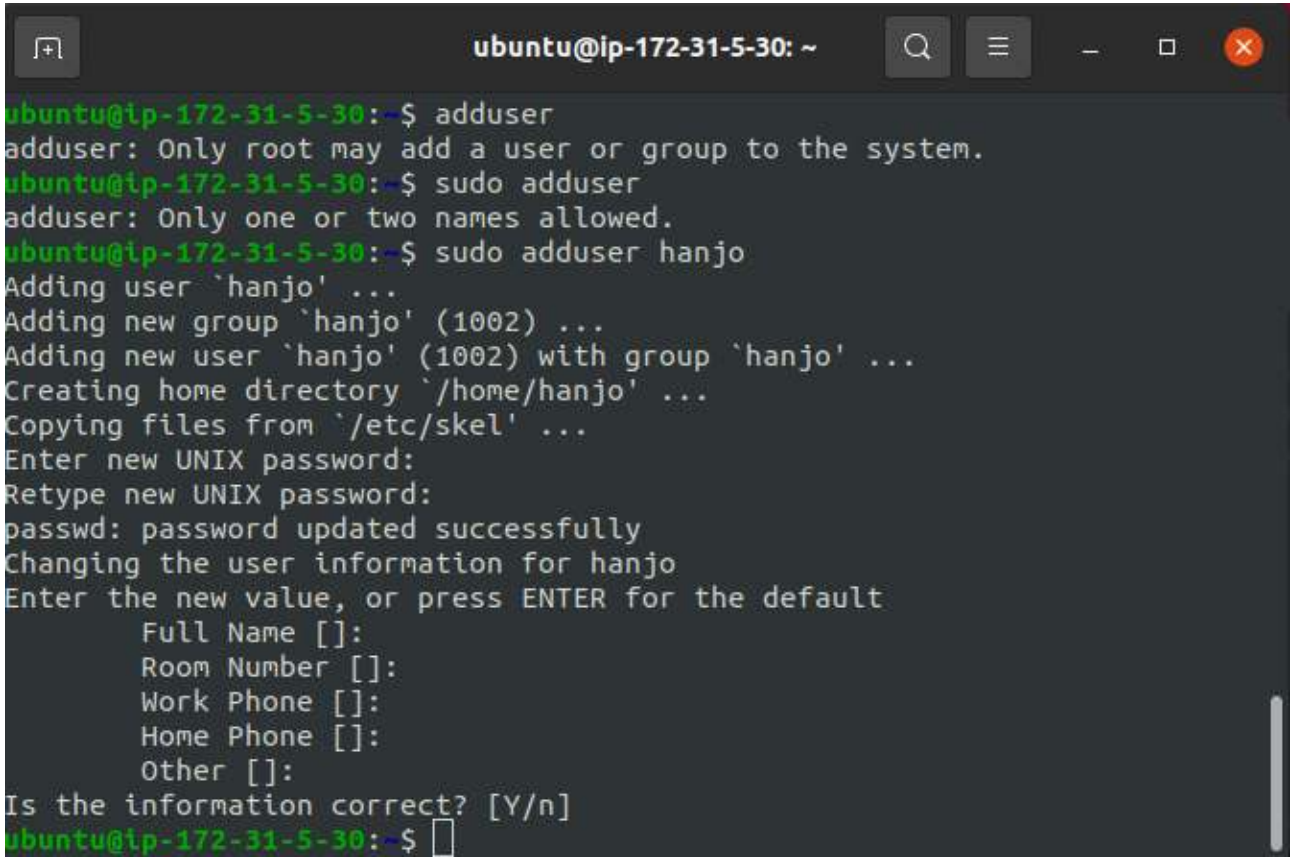
```
##                total        used        free      shared  buff/cache   available
## Mem:            62Gi       8.2Gi        40Gi       3.5Gi        13Gi        50Gi
## Swap:           19Gi          0B        19Gi
```

```
cal
```

```
##       May 2024
## Su Mo Tu We Th Fr Sa
##           1  2  3  4
##  5  6  7  8  9 10 11
## 12 13 14 15 16 17 18
## 19 20 21 22 23 24 25
## 26 27 28 29 30 31
##
```

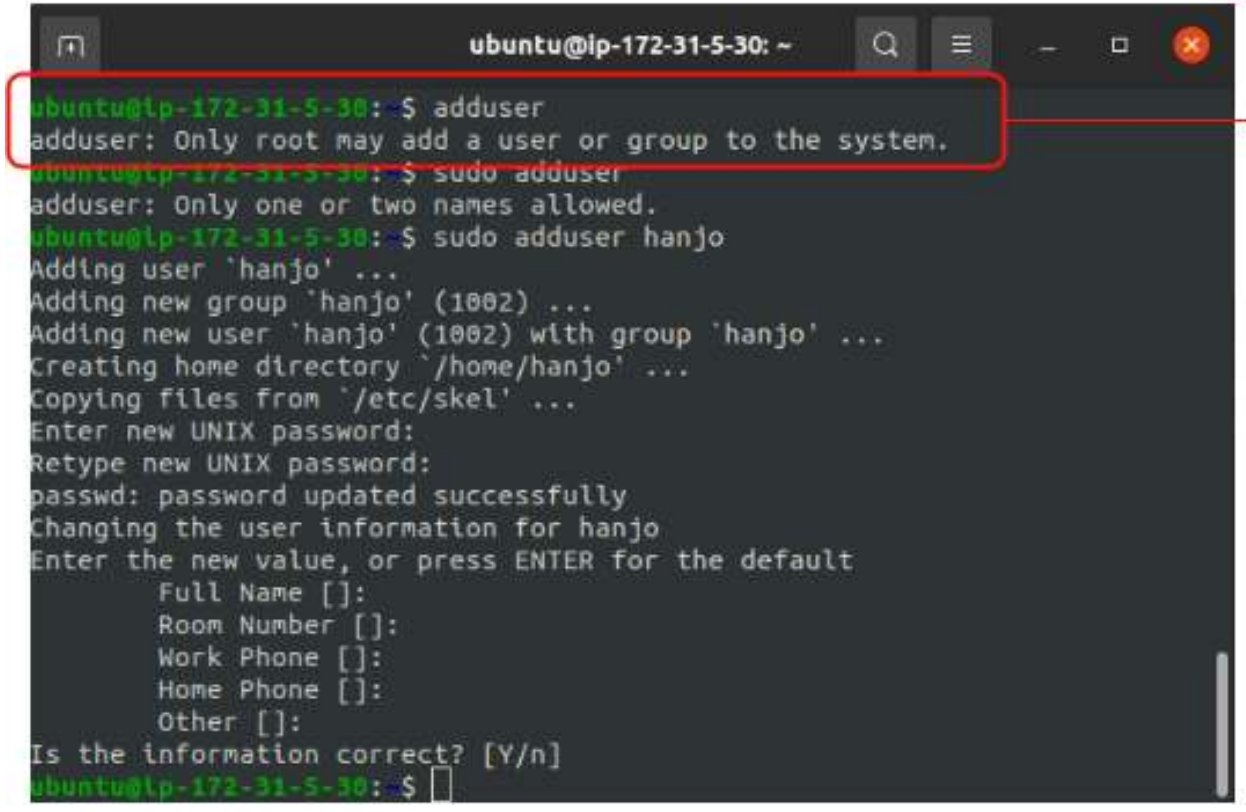
Creating your own user account

- To create a user we will use the `adduser` command.



Creating your own user account

To create a user we will use the `adduser` command.

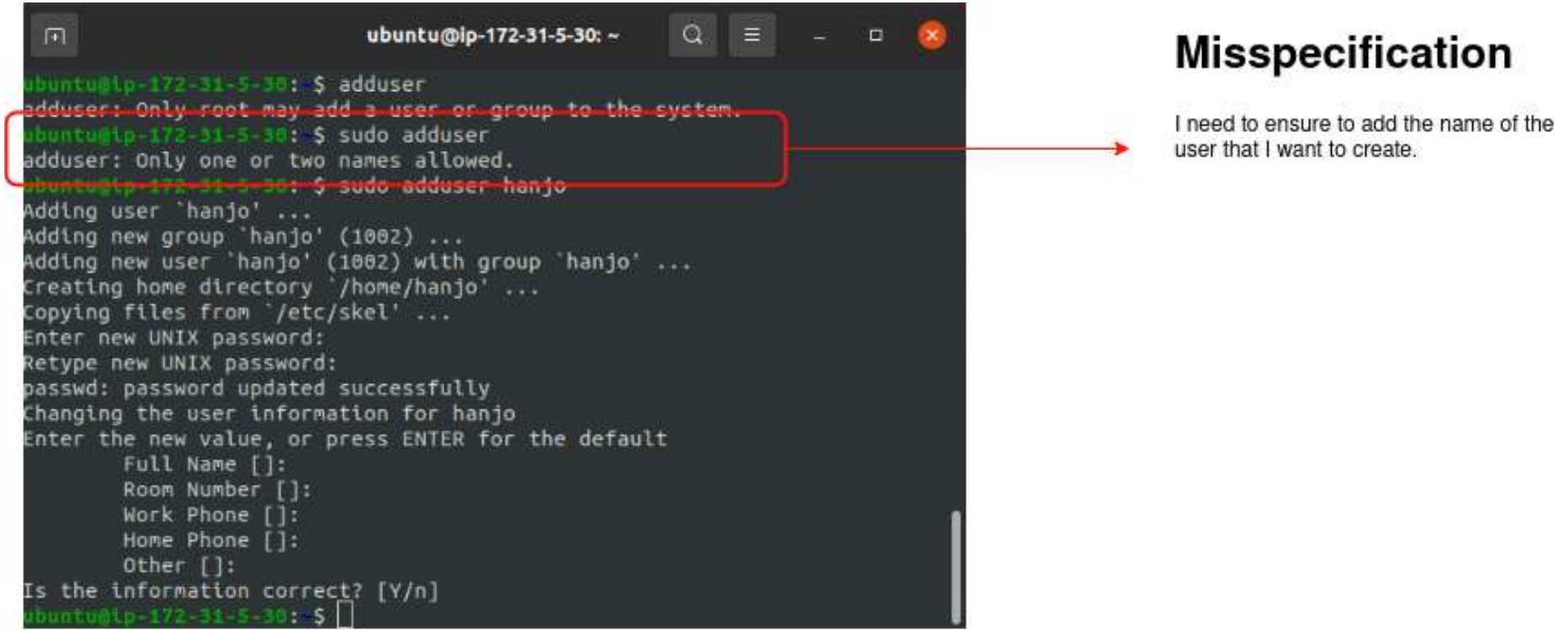


Restriction

Only the superuser may execute this command.

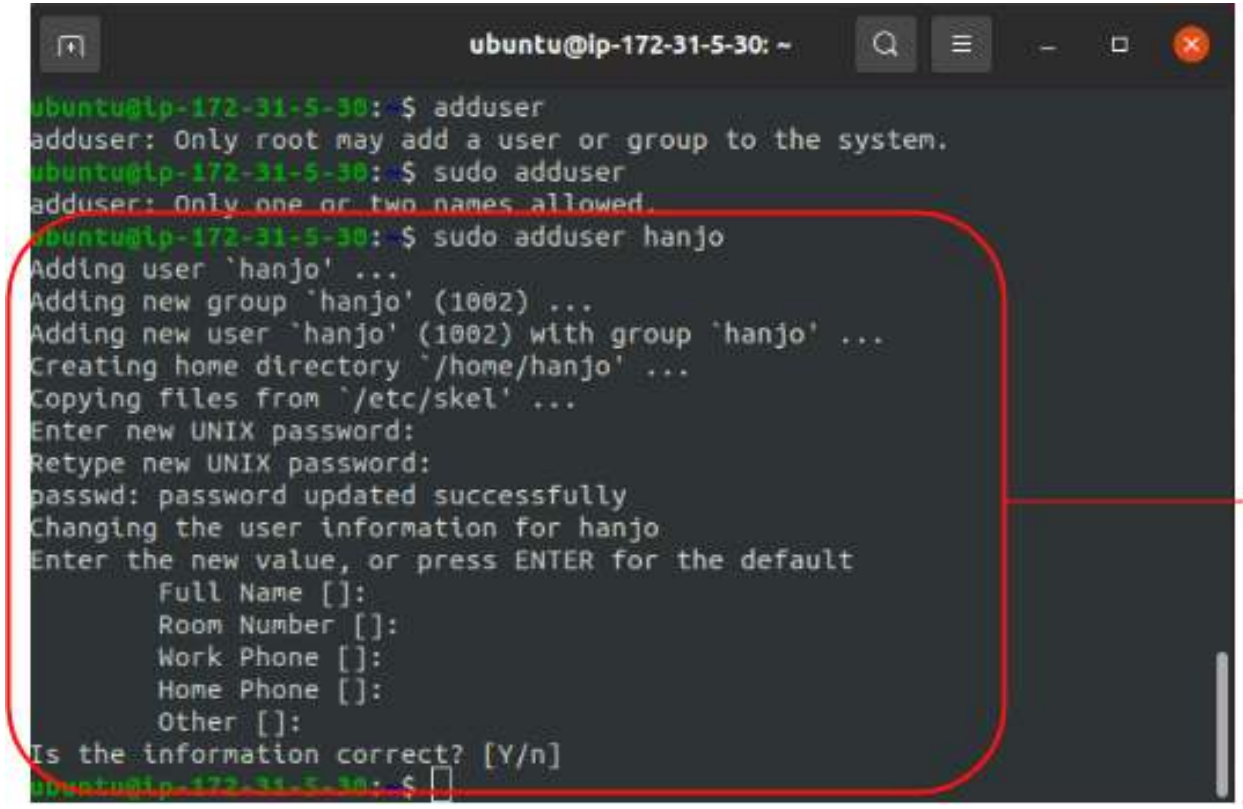
Creating your own user account

To create a user we will use the `adduser` command.



Creating your own user account

To create a user we will use the `adduser` command.



Execution of command

For user Hanjo, there exists now a "home" directory with the files and folders all belonging to user Hanjo

Creating your own user account

> Welcome to your new home, or `127.0.0.1` as I would like to call it.

Login as user Hanjo using `Putty` or `Terminal` and execute the following command:

```
hanjo@optimus:~$ ls -lart

## total 20
## drwxr-xr-x 6 root  root  4096 Dec 19 13:05 ..
## -rw-r--r-- 1 hanjo hanjo  807 Dec 19 13:05 .profile
## -rw-r--r-- 1 hanjo hanjo 3771 Dec 19 13:05 .bashrc
## -rw-r--r-- 1 hanjo hanjo  220 Dec 19 13:05 .bash_logout
## drwxr-xr-x 2 hanjo hanjo 4096 Dec 19 13:05 .
```

- Can anyone tell me what they think the `-rw-r--r--` stands for?

Although we will not go deep into security in this course, it is good to understand some basics.

Permissions

> Owners, Group Members, and Everybody Else

One of the fundamentals that were built into Linux systems from the start is the concept of it being a *multiuser* system. This means that multiple users can log into the system at the same time without interfering (mostly) with each others processes and files.

In the Linux security model, a user may *own* files and directories.

- When a user owns a file or directory, the user has control over its access.
- Users can, in turn, belong to a group consisting of one or more users who are given access to files and directories by their owners.
- An owner may also grant some set of access rights to everybody, which in Linux terms is referred to as the world.

Permissions

> Owners, Group Members, and Everybody Else

How does this look for the user I just created?

```
hanjo@optimus:~$ id
## uid=1002(hanjo) gid=1002(hanjo) groups=1002(hanjo)
```

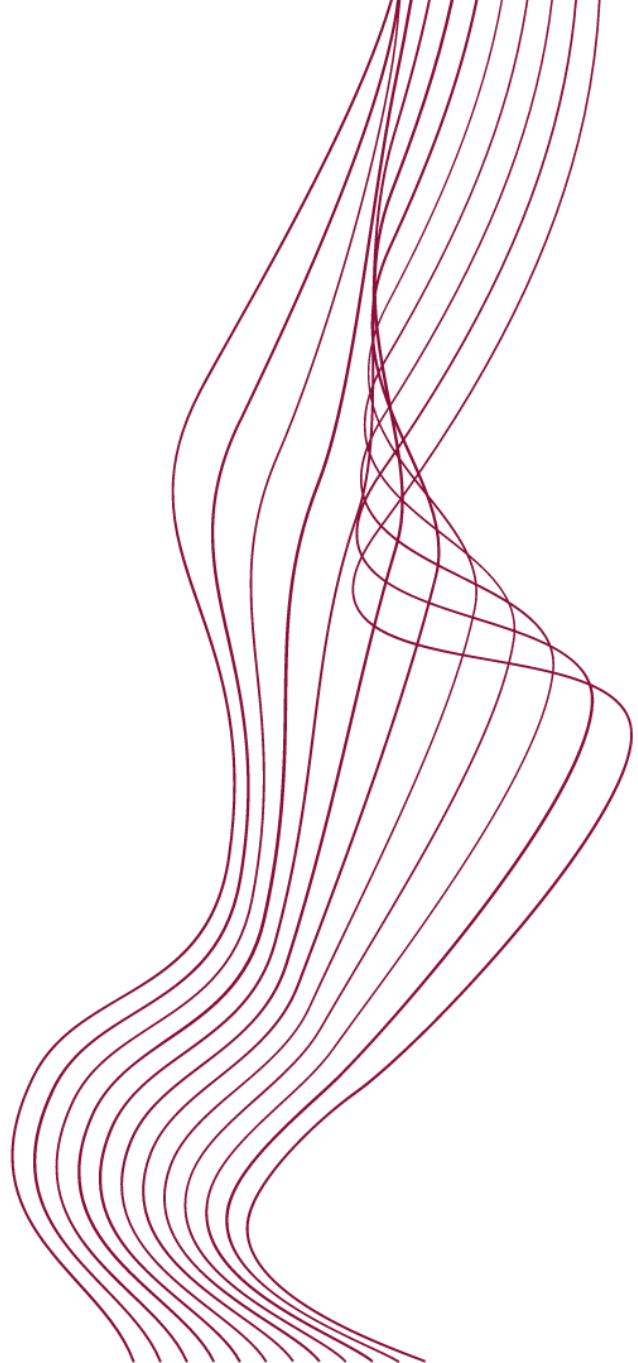
And for Superuser `ubuntu`?

```
## uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(au
```

Welcome to Ubuntu
(01 - Session 3)

Agenda

1) Basic commands
2) Some VIM
3) Crontab

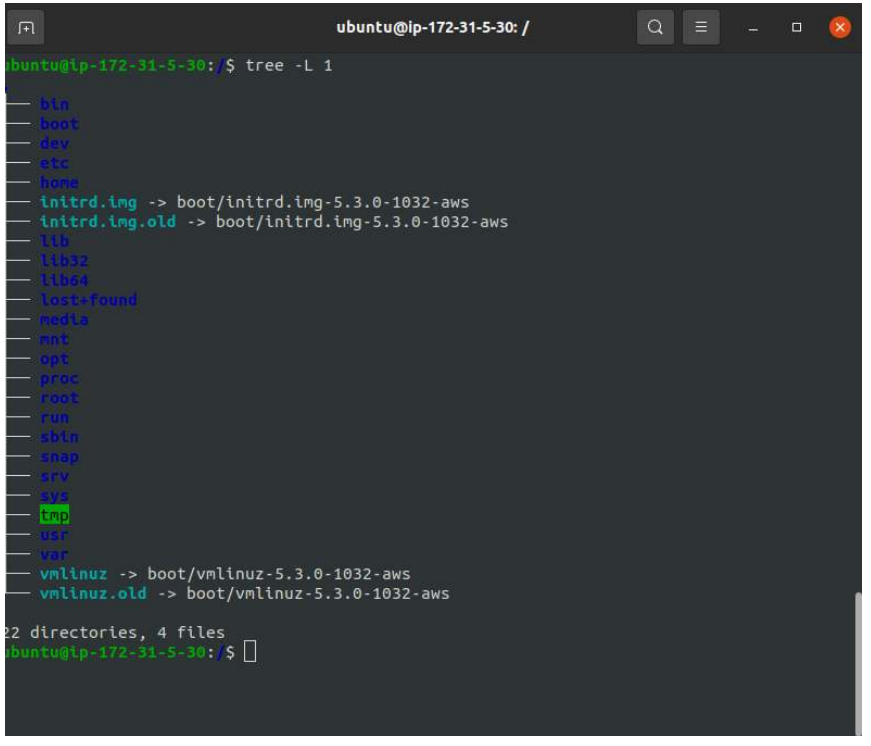


Basic Commands 🤓

Understanding folder structures

Just like any Windows system that you will be used to, all *Unix*-like systems use a folder structure that follows a tree structure. The top-most level is called the *root* directory:

```
hanjo@optimus:~$ tree -L 1
```



Listing directories

To find out where in the `tree` you are, we can use a simple command called: `pwd`

```
hanjo@optimus:~$ pwd
## /home/hanjo
```

Upon logging into a system, the terminal will always set your working directory to `home` also known as `~`.

- If you log in as a regular user, your `home` directory is the only place where you will be able to write and create files.

So, now that we are in the system, what directories are in my `home` folder [*]?

```
hanjo@optimus:~$ ls
## Data  Desktop  Documents  Pictures
```

To list the files and directories in the current working directory, we use the `ls` command. This command is very versatile as you will see in a minute.

[*] *Yours might look a bit different depending whether you are running Linux on a server or a desktop.*

Changing the current working directory

Obviously looking at files in your `home` directory doesn't take you very far. We need to be able to navigate the file system in a quick and efficient manner.

The `cd` command in Linux is a powerful way to navigate the tree folder structure that is the file system.

```
hanjo@optimus:~$ cd Data
hanjo@optimus:~/Data$
```

The two main methods for traversing the tree is: (1) Absolute Paths and (2) Relative Paths:

- **Absolute Paths** begins with the root redirectory `/` and expands to the folder you are interested in: `/home/hanjo/Data`
- **Relative Paths** starts at the working directory and starts navigation from there. These paths have a special notation, a single dot (`.`) and a dot dot (`..`). The `.` notation refers to the working directory, and the `..` notation refers to the working directory's parent directory.

Changing the current working directory

Lets see an example of the **absolute** and **relative** path in action. Start by navigating the `/usr/bin` directory and listing all the files.

```
hanjo@optimus:~$ cd /usr/bin
hanjo@optimus:/usr/bin$ pwd
#/usr/bin
hanjo@optimus:/usr/bin$ ls
## 2to3-2.7 funzip mpiCC splitfon ...
```

Now lets move to the `/usr` directory from our working directory `/usr/bin`. There are two ways to do this, either **absolute** (`cd /usr`) or **relative**. Let us practice using the **relative** method.

```
hanjo@optimus:~$ cd /usr/bin
hanjo@optimus:/usr/bin$ cd ..
hanjo@optimus:/usr$ pwd
# /usr
hanjo@optimus:/usr$ ls
# bin/  games/  include/  lib/  lib32/  local/  sbin/  share/  src/
```

Changing the current working directory

There are also some nice shortcuts to be aware of:

- Change the working directory to your home directory: `cd ~`
- Change the working directory to the previous working dir: `cd -`
- Change the working directory to a specific user: `cd ~ubuntu`

Notes about filenames in Linux

Filenames in Linux are quite special and if you have worked closely with someone who works in Linux, you would have noticed some things. First and foremost:

- NEVER use a space in filenames use an underscore (`_`) instead - thank me later ;-)
  - ex. `this file name sucks.txt` where `this_is_much_better.txt`
- Filenames that start with a `.` are hidden files. The `ls` command will not list these unless you use a *parameter* `ls -a`. These files usually relate to configuration settings.
  - ex. `.bashrc`.
- CASE MATTERS, so dont ever use Capitals for folders or filenames - it gets confusing.
  - ex. `This/path/IS/different/`. from `/this/path/is/different/`
- Linux does not have any concept of "file extensions". So remember to name your files in an appropriate manner if you would like them to be readable by the correct application.
  - ex. `mypdffile` and `mypdffile.pdf` is the same

See this presentation by Dr. Anna Krystalli for further tips on file naming.

Getting to know 'ls'

The `ls` command is probably one of the most used commands that any Linux user will encounter from day to day. As you will come to see, it is also one of the most powerful commands.

Let's start by listing the contents of `/usr` while our working directory is `~`:

```
hanjo@optimus:~$ ls /usr
# bin/  games/  include/  lib/  lib32/  local/  sbin/  share/  src/
```

You can also ask for multiple directories in a single line:

```
hanjo@optimus:~$ ls /usr ~
## /home/hanjo:
## Data  Desktop  Documents  Pictures
##
## /usr:
## bin  games  include  lib  lib32  local  sbin  share  src
```

Options and Arguments

By now you should have noticed once or twice that I have added an *options* parameter to my commands: `command -options arguments`. Type `man ls` to see all options for the `ls` command.

```
hanjo@optimus:~$ ls -l
## total 16
## drwxrwxr-x 2 hanjo hanjo 4096 Dec 20 09:23 Data
## drwxrwxr-x 2 hanjo hanjo 4096 Dec 20 09:23 Documents
```

My favourite command is `ls -lart` which stands for "list ALL the contents in REVERSE order SORT BY TIME".

```
hanjo@optimus0:~$ ls -lart
## total 40
## drwxr-xr-x 6 root  root  4096 Dec 19 13:05 ..
## -rw-r--r-- 1 hanjo hanjo  807 Dec 19 13:05 .profile
## -rw-r--r-- 1 hanjo hanjo 3771 Dec 19 13:05 .bashrc
## -rw-r--r-- 1 hanjo hanjo  220 Dec 19 13:05 .bash_logout
## -rw------- 1 hanjo hanjo   26 Dec 19 13:35 .bash_history
## drwxrwxr-x 2 hanjo hanjo 4096 Dec 20 09:23 Documents
## drwxrwxr-x 2 hanjo hanjo 4096 Dec 20 09:23 Data
## drwxr-xr-x 6 hanjo hanjo 4096 Dec 20 09:23 .
```

Creating files and folders

Apart from knowing how to navigate folders, we must also know how to create files and folders.

The basic commands for this is:

- Create folder

```
mkdir scripts
mkdir scripts data analysis
```

- Create file

```
touch analysis.R
```

90% of the time you will be using the basic versions of these commands. But they can also do some pretty interesting things.

Tricks and Tips for mkdir

- Create folders within folders that do not already exist (recursively create).

```
mkdir project/analysis/scripts
# mkdir: cannot create directory ‘project/analysis/scripts’: No such file or directory

# Correct usage
mkdir -p project/analysis/scripts
```

- What if I wanted to create a `data`, `scripts` and `output` folder in a single line?

```
# Note, there is NO spaces in the array
mkdir -p project/analysis/scripts/{data,scripts,output}
cd project/analysis/scripts/ && ll
```

- Current date in directory name

```
mkdir `date '+%Y%m%d'`
```

Viewing contents of files

To view the contents of a file, we use a program called `less`.

> The `less` program was designed as an improved replacement of an earlier Unix program called `more`. The name `less` is a play on the phrase "*less is more*" — a motto of modernist architects and designers.

`more` (developed in 1978) was replaced by `less` in 1983, first and foremost because `more` could only scroll forwards through a text file. `less` was written by Mark Nudelman and is currently being maintained by him to this day!

- Backwards movement
- Searching and highlighting
- Multiple files
  - Less allows you to switch between any number of different files, remembering your position in each file. You can also do a single search which spans all the files you are working with.
- Advanced features
  - You can change key bindings, set different tab stops, set up filters to view compressed data or other file types, customize the prompt, display line numbers, use "tag" files, and more.

http://www.greenwoodsoftware.com/less/faq.html#mail

Viewing contents of files

Lets start by looking at the users on the system:

```
hanjo@optimus0:~$ less /etc/passwd
```

Navigation:

- **G** - Move to the end of the text file
- **g** - Move to the beginning of the text file
- **10g** - Move to the nth line
- **q** - Exit

Forward Search:

- **/characters** - Search forward
- **n** - Search forward
- **N** - Search backwards

Useful options for Less

Squeeze Blank Lines:

- The `-s` (squeeze blank lines) option removes a series of blank lines and replaces them with a single blank line.

Viewing Multiple Files:

- `less file1.txt file2.txt`
- To view the next file, press `:` and then hit `n`.

Mark places:

- Press `m` and then a letter, example: `a`. To return to that mark press apostrophe `'` and `a`.

Switch to editor:

- Pressing `v` while in `less` pushes you to default editor.
- `sudo update-alternatives --config editor`

Practice using Less

First lets get some files to work with:

```
hanjo@optimus0:~$ cd ~
hanjo@optimus0:~$ wget -O war_and_peace.txt https://www.gutenberg.org/files/2600/2600-0.txt
hanjo@optimus0:~$ wget -O anna_karenina.txt https://www.gutenberg.org/files/1399/1399-0.txt
```

> The only way to write good code is to write tons of bad code first. Feeling shame about bad code stops you from getting to good code.
>
> — *Hadley Wickham*

05:00

Practice using Less

Please demonstrate to the class how you would do the following (open up both files at same time):

- Using War and Peace, search for the following animals: horse, eagle, bear.
- Start Anna Karenina but remove excessive white space.
- Provide a "mark" when Count Alexey Kirillovich Vronsky is first mentioned in Anna Karenina.
    - On which line number of the text is this?

71point4
cenfri
mastercard foundation

Learning basics of VIM 

Why learn VIM?

There is an old joke about a visitor to New York City asking a passerby for directions to the city's famous classical music venue.

> Visitor: Excuse me, how do I get to Carnegie Hall?
> Passerby: Practice, practice, practice!

Learning the Linux command line, like becoming an accomplished pianist, is not something that we pick up in an afternoon. It takes years of practice. In this chapter, we will introduce the `vi` (pronounced "vee eye") text editor, one of the core programs in the Unix tradition. `vi` is somewhat notorious for its difficult user interface, but when we see a master sit down at the keyboard and begin to "play," we will indeed be witness to some great art. We won't become masters in this chapter, but when we are done, we will know how to play the equivalent of "Chopsticks" in `vi`.

Linux Command Line, 2nd Edition - Jr. William E. Shotts

Why learn VIM?

So why learn VIM when you have something like `VSCode` or `Rstudio` as an IDE?

- VIM is ubiquitous on all Unix systems, which mean you will always have access to an editor, even if your front-end crashes.
- VIM is powerful and fast. Sometimes you just need to change a simple config file and VIM works best for these times. Also, you may not have GUI access when doing routine SysAdmin tasks as root.
- Once you have mastered VIM[1], then there are few way to be more efficient in typing up code or changing files since you never need a mouse.
- We don't want other Linux and Unix users to think we are cowards.[2]

[1] *No one on earth can say that they have mastered VIM.*

[2] *Its a joke from Linux Command Line, 2nd Edition, but still true* 😜

First things first, how to exit

> Stackoverflow, helping people exit vim since 2011.

To exit we enter the *editor* with `:`, then type `q` and a `!` (the exclamation, `!`, means to force close):

```
:q!
```

Basics of editing a file

⚠️ Follow my commands before typing. Do not type anything yet!

> Remember, if something bad happens just press ESC a couple of times and then exit VIM with `:q!`

```
hanjo@optimus0:~$ vim owner_information.txt
```

- In VIM, every keystroke is a specific command, this type of editor is known as a *modal editor*.
  - VIM starts by going into *command mode*, which means it expects commands, NOT input text.

To type something we must go to *Insert Mode*. To do this, type `i`. You should see the following at the bottom:

```
--INSERT--
```

Now, type the following:

```
[owner] Hanjo Odendaal
```

Save and exit by pressing `ESC` and `:wq`

Basics of editing a file

What happens if you happen to have made a mistake? To delete a file, use the `rm` command.

⚠️ In Linux, when you delete the file is gone forever. So be careful!

```
hanjo@optimus0:~$ rm owner_information.txt
```

Moving around VIM like a Pro

VIM has some very nice commands which enables you to quickly move through a piece of text without needing a mouse.

> If you can conquer these, not only will you be more efficient, but your colleagues will also be impressed 😮

Make sure you are in command come when you use these commands (Press `ESC` a couple of times):

- **0 (zero)** - Beginning of line.
- **$** - End of line.
- **w** - Next word (**W** ignores punctuation).
- **b** - Back a word (**B** ignores punctuation).
- **nG** - Move to specific line (**10G** will take you to the 10th line. Just **G** goes to last line in file.).
- **o** - Opening a line below (Think **i**, but opens next line. **O** opens above current line).
- **x** - Deletes a character.
- **dd** - Deletes a line.
- **yy** - Yanks (copies) a line.
- **p** - Pastes a line.

Practice using VIM

Recreate the file called `owner_information.txt`:

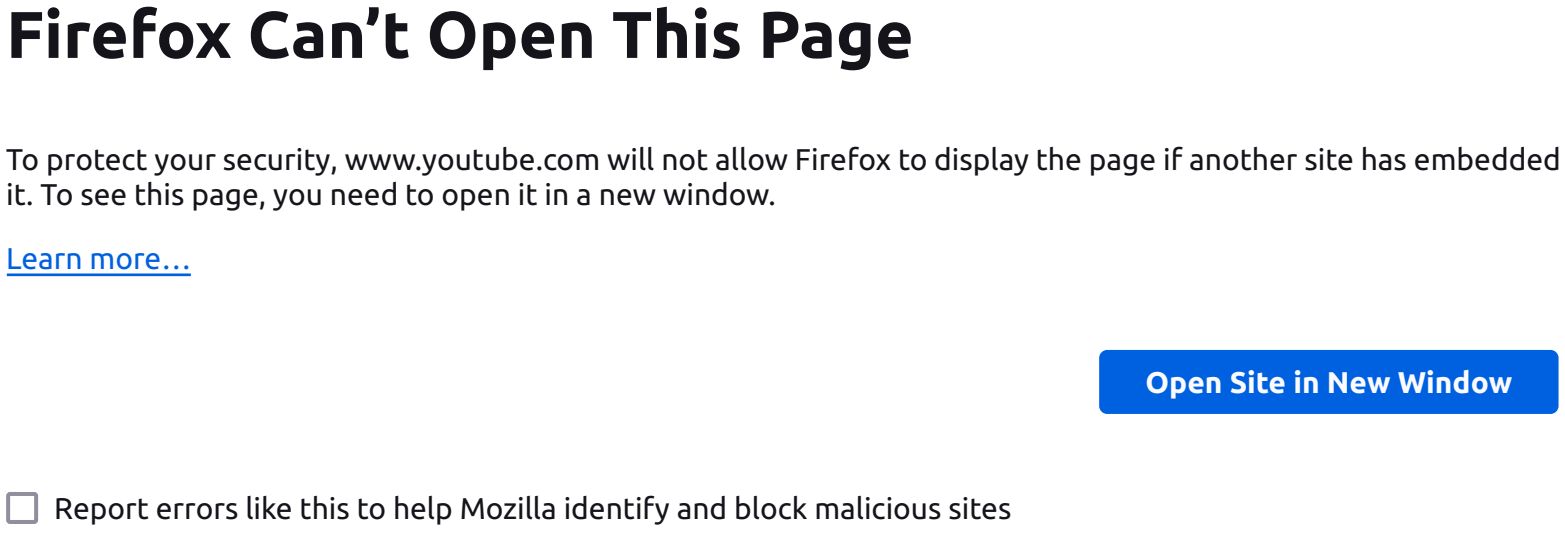
```
hanjo@optimus0:~$ vim owner_information.txt
```

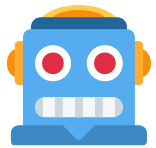
Next, add the following information into the file using ONLY your newly learned commands:

```
[OWNER]
Hanjo Odendaal

[OCCUPATION]
Data Scientist

[FAVOURITE COMMAND]
ls -lart
```

05:00

Side-quest


Firefox Can’t Open This Page
To protect your security, www.youtube.com will not allow Firefox to display the page if another site has embedded it. To see this page, you need to open it in a new window.
Learn more…
Open Site in New Window
Report errors like this to help Mozilla identify and block malicious sites

Redirection in Linux 

Redirection & Piping

This is maybe one of the coolest features of command line that you will learn: *Redirecting* or *piping* your results into another command. The *Input/Output* allows us to chain together commands and build pipelines of instructions.

- I/O redirection (`>`) allows us to change where output goes and where input comes from.
  - A good example of this would be the `ls` command we learned earlier.

```
hanjo@optimus0:~$ ls -l
hanjo@optimus0:~$ ls -l > all_files.txt
hanjo@optimus0:~$ less all_files.txt
```

We can also append a file using (`>>`):

```
hanjo@optimus0:~$ ls -l >> all_files.txt
hanjo@optimus0:~$ ls -l >> all_files.txt
hanjo@optimus0:~$ ls -l >> all_files.txt
hanjo@optimus0:~$ less all_files.txt
```

Redirection & Piping

In the previous examples we redirected only the `stdout` of the command. But, we sometimes also want to redirect the errors or Standard Error (`stderr`).

To do this we add an additional command to the end of the line (`2>&1`):

```
hanjo@optimus0:~$ ls -l > all_files.txt 2>&1
hanjo@optimus0:~$ less all_files.txt
```

We redirect file descriptor 2 (standard error) to file descriptor 1 (standard output) using the notation `2>&1`.

Once we know the concept of standard output and input, we can start stringing commands together. These are called *pipelines* and it looks this, `command1` *pipes into* `command2`:

```
command1 | command2
```

Here we can see that `command2` takes `command1`'s output as its input. As you get more comfortable with the terminal, these become core concepts you will use every day.

Putting your first script into production

> Redirect and piping are core functions of Linux systems. Now its time for you to put those functions to good use and `productionize` your first *script*.

What is a *script*? A script is a series of commands that execute (mostly) from top to bottom if we ask Linux to run the file, also known as executing the *script*.

Lets imagine we want to create a file that runs every hour that checks what files are in a directory. Once I get the list I want to sort the list and only get uniques. How would I do this step by step?

```
hanjo@optimus0:~$ ls /bin /usr/bin
```

```
hanjo@optimus0:~$ ls /bin /usr/bin | sort
```

```
hanjo@optimus0:~$ ls /bin /usr/bin | sort | uniq
```

```
hanjo@optimus0:~$ ls /bin /usr/bin | sort | uniq >> ~/all_binaries.txt
```

Putting your first script into production

How do we know turn this into a script? Well, lets use `VIM` to open a file and write our commands:

```
hanjo@optimus0:~$ vim sorting.sh
```

In VIM editor, add your commands that you want to run:

```
#!/bin/bash

DATEVAR=$(date +%Y%m%d)

logger -s "Todays date is: $DATEVAR"
logger -s "<---------------------- START ----------------------->"
ls /bin /usr/bin | sort | uniq | head
```

Putting your first script into production

Understanding each of the commands:

```
#!/bin/bash

DATEVAR=$(date +%Y%m%d)

logger -s "Todays date is: $DATEVAR"
logger -s "<---------------------- START ----------------------->"
ls /bin /usr/bin | sort | uniq | head
```

`#!` or *shebang* allows the operating system to know what type of script this is. In our case, its the `bash` shell.

This enables us to run the script from the command line using a simple syntax if we tell Linux this is an executable script!

```
hanjo@optimus0:~$ chmod +x sorting.sh
```

```
hanjo@optimus0:~$ ./sorting.sh
```

Putting your first script into production

Understanding each of the commands:

```
#!/bin/bash

DATEVAR=$(date +%Y%m%d)

logger -s "Todays date is: $DATEVAR"
logger -s "<---------------------- START ----------------------->"
ls /bin /usr/bin | sort | uniq | head
```

The dollar `$` sign denotes a variable in bash. Here I am reformatting the `date` command and assigning that output to variable `DATEVAR`. This means I can now use that variable in multiple places in my script instead of typing `$(date +%Y%m%d)` everywhere.

Putting your first script into production

Understanding each of the commands:

```
#!/bin/bash

DATEVAR=$(date +%Y%m%d)

logger -s "Todays date is: $DATEVAR"
logger -s "<---------------------- START ----------------------->"
ls /bin /usr/bin | sort | uniq | head
```

Logging is one of the most important parts of scripting. How else are you suppose to know what your script is doing? There are a couple of ways to *log*. You can use `echo` or the `logger` command.

Putting your first script into production

Understanding each of the commands:

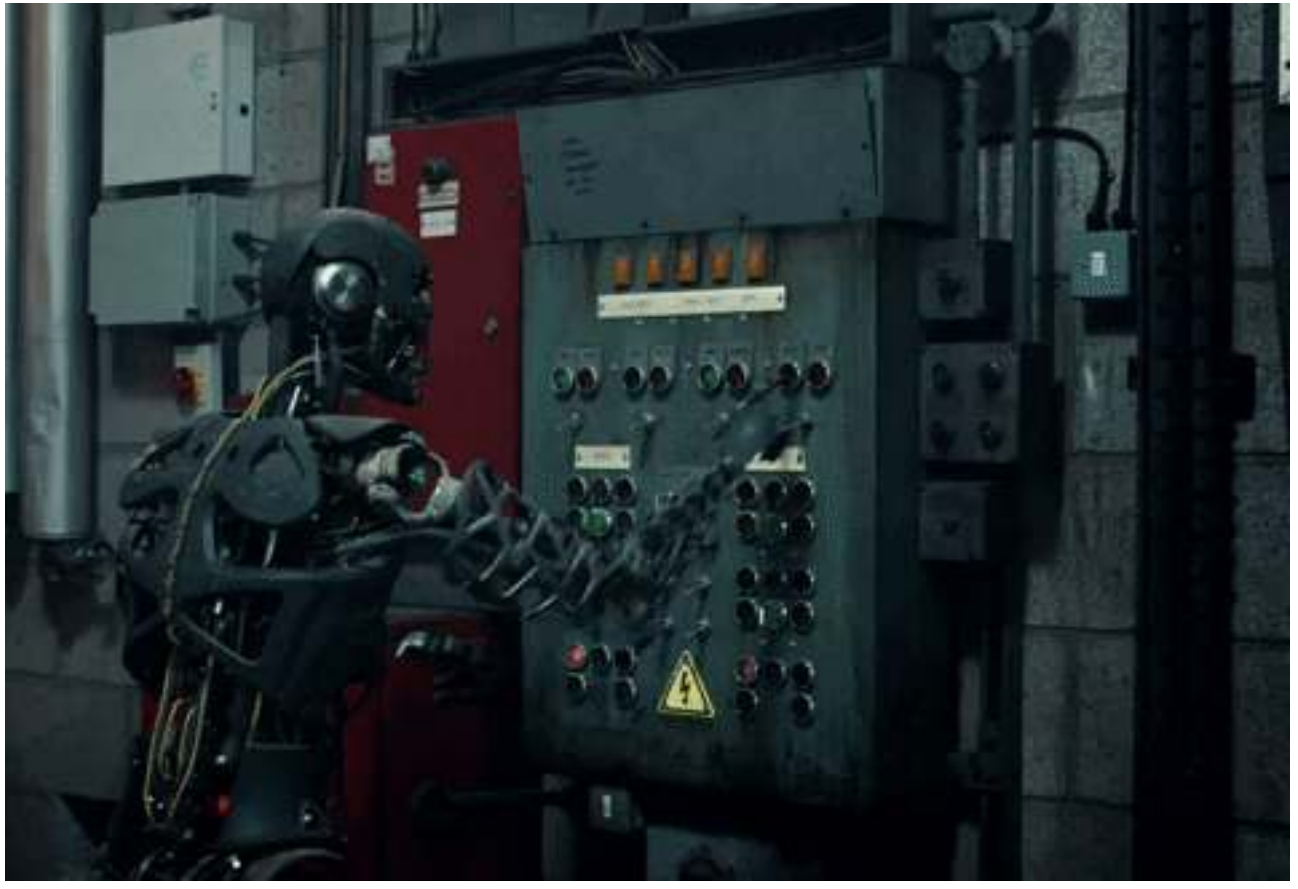
```
#!/bin/bash

DATEVAR=$(date +%Y%m%d)

logger -s "Todays date is: $DATEVAR"
logger -s "<---------------------- START ----------------------->"
ls /bin /usr/bin | sort | uniq | head
```

Lastly, we run the command that we want to execute. Here I am:

- Getting the files in 2 directories
- Sorting the files
- Getting unique files
- Only taking top 5

Putting your first script into production

> The time has come!



Cronjobs and crontabs

What is cron?

> The cron command-line utility, also known as cron job is a job scheduler on Unix-like operating systems.

Lets open crontab:

```
hanjo@optimus0:~$ crontab -e
```

```
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Cronjobs and crontabs

Lets say that the script must run every minute and output to a file in a folder called `logs`:

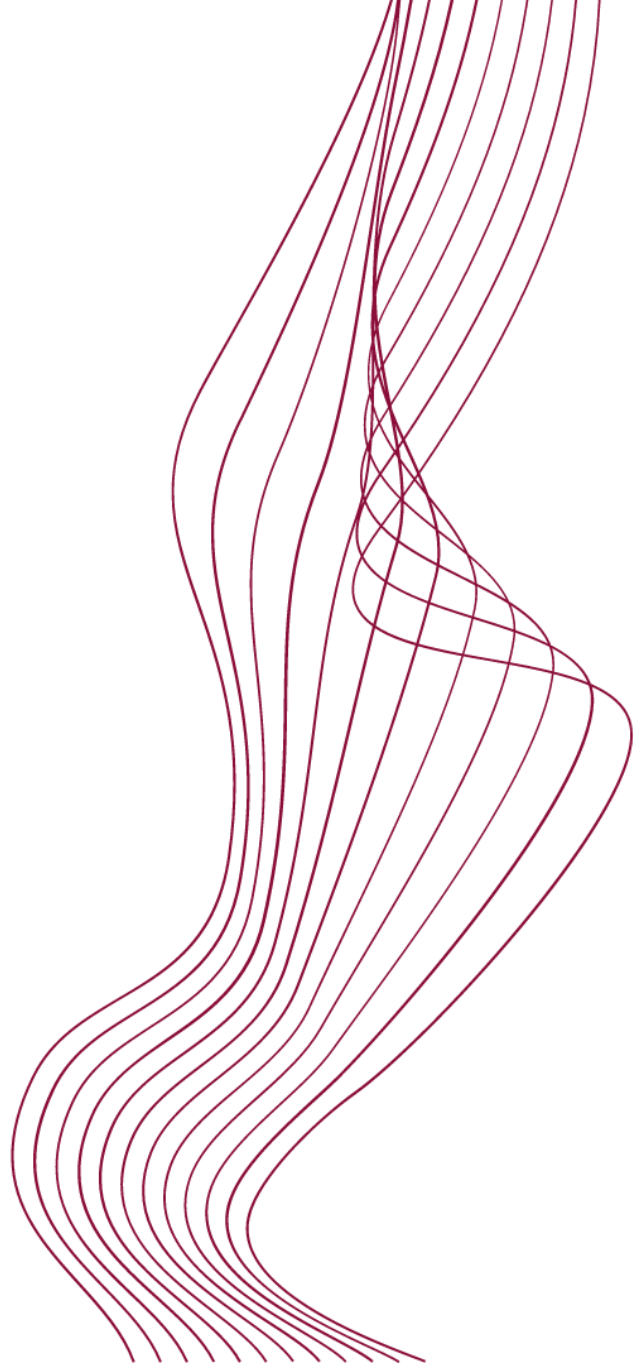
```
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

  * * * * ./sorting.sh >> ~/logs/sort_logs.log 2>&1
```

The end

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 2 - Documenting & Databases)

Agenda

1) Homework
2) Documenting with Rstudio
3) Introduction to databases



Homework 

Documenting with Rstudio 🧪

Why do we document our code?

When working in a lab, it is important to always take notes on the steps taken in the experiment - why?

- Ensure robustness of results.
- Reliability of reproducibility.
- Ensures that decision can be made using the notes.
- Future you will hate you if you didn't write good documentation and need to redo the experiment or analysis.

But we do not just write down irrelevant comments, we need to make sure our documentation FAIR:

- findable
- accessible
- interoperable
- reusable

> i.e. they must adequately describe procedure, archive changes, and make the results accessible in an easy manner.

⚠️ As programmers, we need to ensure that we document both the code that produced the results as well as the procedures used to conduct the analysis (data cleaning, sampling, source of information etc.).

Reproducible research as a philosophy

A data analysis is reproducible if all the information (data, files, etc.) required to reproduce the analysis is available to someone else (or future you). These include (but is not limited to):

- Data repository.
- All code files for cleaning raw data.
- All code files and software (specific versions, packages) used in the analysis.

Some advantages of making your research reproducible are [1]:

- You can (easily) figure out what you did six months from now.
  - If your documentation was well done.
- You can (easily) make adjustments to code or data, even early in the process, and re-run all analysis.
- When you're ready to publish, you can (easily) do a last double-check of your full analysis, from cleaning the raw data through generating figures and tables for the paper.
- You can pass along or share a project with others.
  - Especially true once you learn `git`
- You can give useful code examples to people who want to extend your research.

[1] *Gandrud, C., 2013. Reproducible research with R and Rstudio. CRC Press.*

The Reinhart-Rogoff error – or how not to Excel at economics

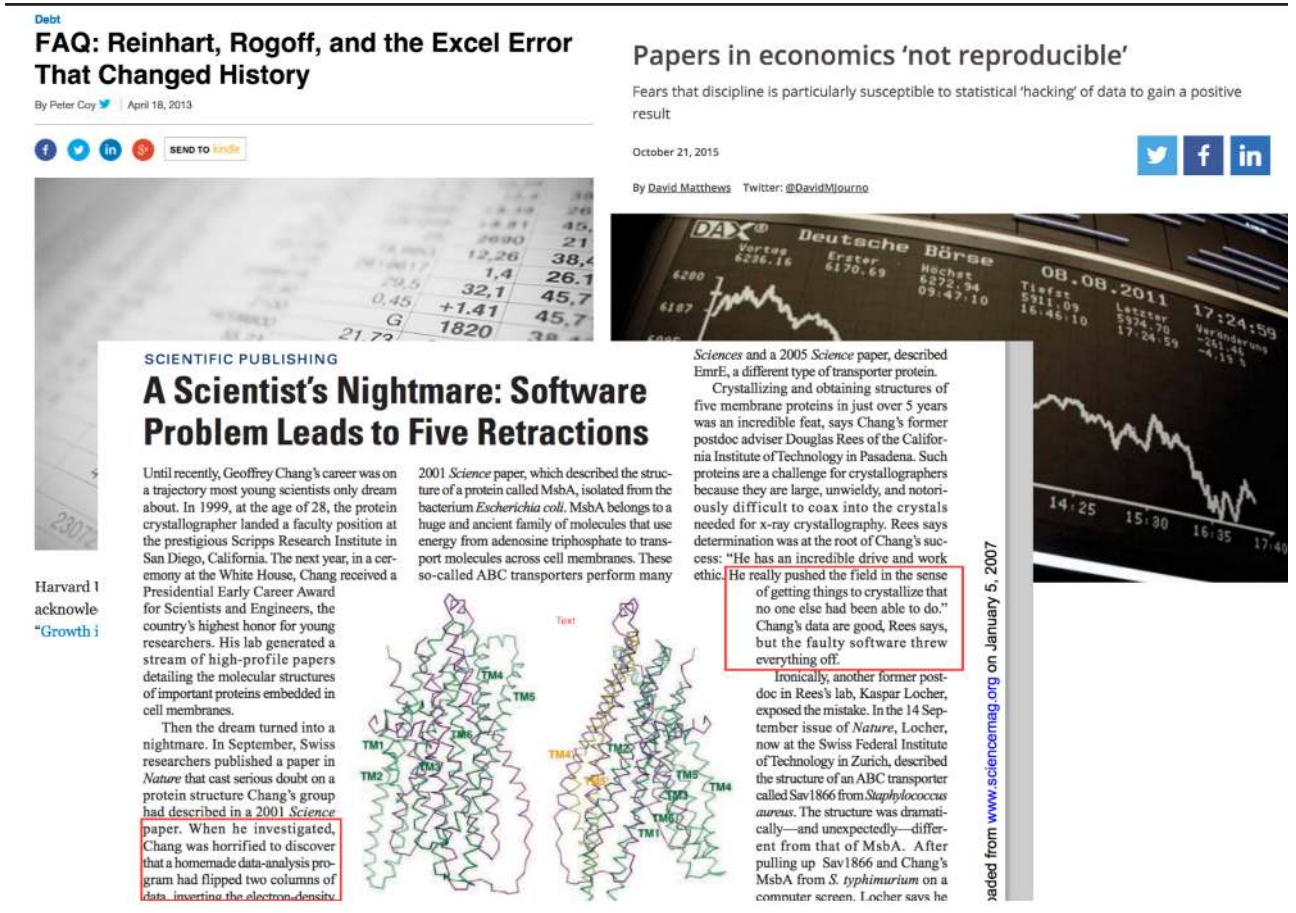
Growth in a Time of Debt

Carmen M. Reinhart & Kenneth S. Rogoff

WORKING PAPER 15639 **DOI** 10.3386/w15639 **ISSUE DATE** January 2010 **REVISION DATE** December 2011

We study economic growth and inflation at different levels of government and external debt. Our analysis is based on new data on forty-four countries spanning about two hundred years. The dataset incorporates over 3,700 annual observations covering a wide range of political systems, institutions, exchange rate arrangements, and historic circumstances. Our main findings are: First, the relationship between government debt and real GDP growth is weak for debt/GDP ratios below a threshold of 90 percent of GDP. Above 90 percent, median growth rates fall by one percent, and average growth falls considerably more. We find that the threshold for public debt is similar in advanced and emerging economies. Second, emerging markets face lower thresholds for external debt (public and private)--which is usually denominated in a foreign currency. When external debt reaches 60 percent of GDP, annual growth declines by about two percent; for higher levels, growth rates are roughly cut in half. Third, there is no apparent contemporaneous link between inflation and public debt levels for the advanced countries as a group (some countries, such as the United States, have experienced higher inflation when debt/GDP is high.) The story is entirely different for emerging markets, where inflation rises sharply as debt increases.

The Reinhart-Rogoff error – not unique



Debt

FAQ: Reinhart, Rogoff, and the Excel Error That Changed History

By Peter Coy | April 18, 2013

SEND TO kindle

Papers in economics 'not reproducible'

Fears that discipline is particularly susceptible to statistical 'hacking' of data to gain a positive result

October 21, 2015

By David Matthews Twitter: @DavidMJourno

SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a Presidential Early Career Award for Scientists and Engineers, the country's highest honor for young researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

Then the dream turned into a nightmare. In September, Swiss researchers published a paper in *Nature* that cast serious doubt on a protein structure Chang's group had described in a 2001 *Science* paper. When he investigated, Chang was horrified to discover that a homemade data-analysis program had flipped two columns of data, inverting the electron-density

2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many

Sciences and a 2005 *Science* paper, described EmrE, a different type of transporter protein.

Crystallizing and obtaining structures of five membrane proteins in just over 5 years was an incredible feat, says Chang's former postdoc adviser Douglas Rees of the California Institute of Technology in Pasadena. Such proteins are a challenge for crystallographers because they are large, unwieldy, and notoriously difficult to coax into the crystals needed for x-ray crystallography. Rees says determination was at the root of Chang's success: "He has an incredible drive and work ethic. He really pushed the field in the sense of getting things to crystallize that no one else had been able to do." Chang's data are good, Rees says, but the faulty software threw everything off.

Ironically, another former postdoc in Rees's lab, Kaspar Locher, exposed the mistake. In the 14 September issue of *Nature*, Locher, now at the Swiss Federal Institute of Technology in Zurich, described the structure of an ABC transporter called Sav1866 from *Staphylococcus aureus*. The structure was dramatically—and unexpectedly—different from that of MsbA. After pulling up Sav1866 and Chang's MsbA from *S. typhimurium* on a computer screen, Locher says he

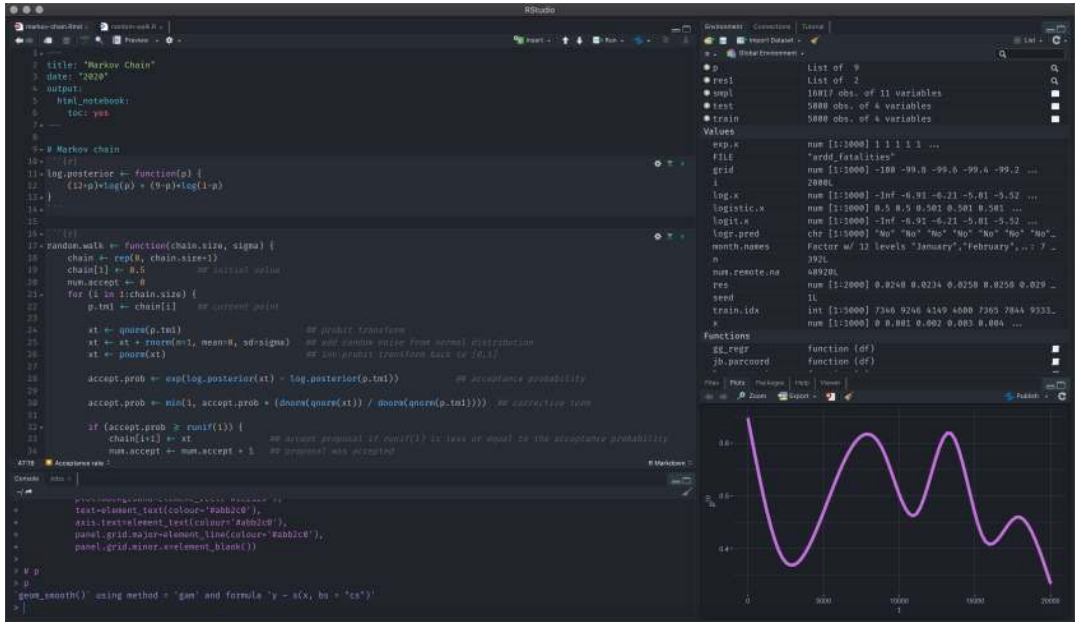
Installing your first piece of software

> Experience the ease of software installation in Linux!

Go to rstudio server website.

```
hanjo@optimus:~$ cd Downloads
hanjo@optimus:~$ wget {installation file p
hanjo@optimus:~$ dpkg -i {installation fil
```

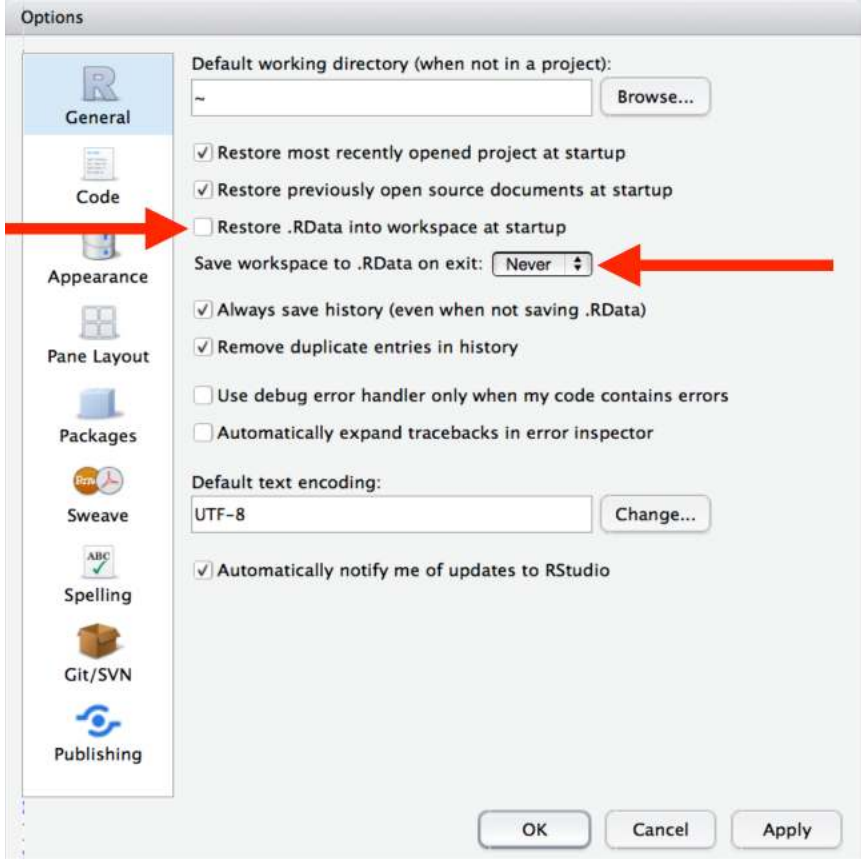
Verify your install, go to the ip of your machine in the web-browser



Setting up Rstudio for analysis

> To ensure reproducibility, we want to ensure that our scripts are always able to run without needing some hidden data.

- Make sure that the Rstudio never restores `.RData` at startup.
- This ensures that no hidden *objects* are still in your *environment* when you start Rstudio.
  - We will talk a little bit more about these concepts later in the course.



Using Projects

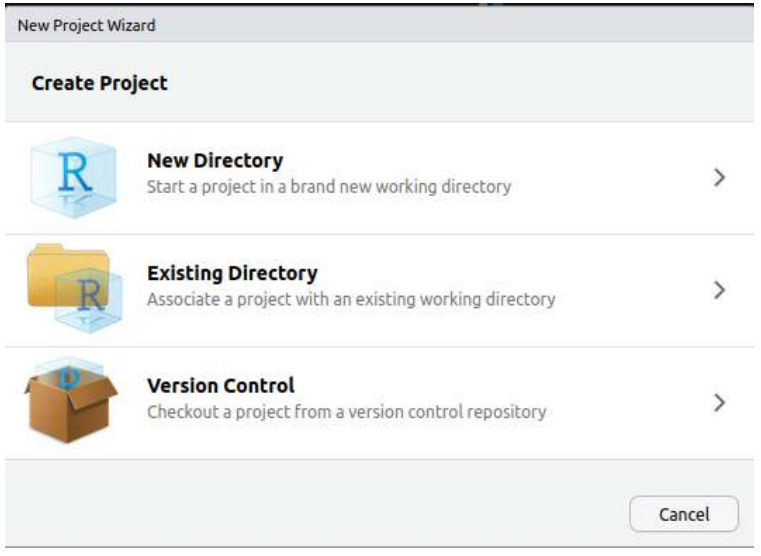
Ever had the following expression when people ask you 8 months later "*Where is that bit of analysis you did for me*": 🤔.

We want to avoid feeling like that by keeping all our *notes*, *scripts*, *data* and *output* in one single place. This is where Rstudio makes it easy by creating a project.

Start by creating a folder in your `home` directory called `projects` and starting a project called `markdown`:

```
hanjo@optimus:~$ mkdir -p projects/markdown
```

- Next click on the menu:
  `File > New Project`



Using Projects

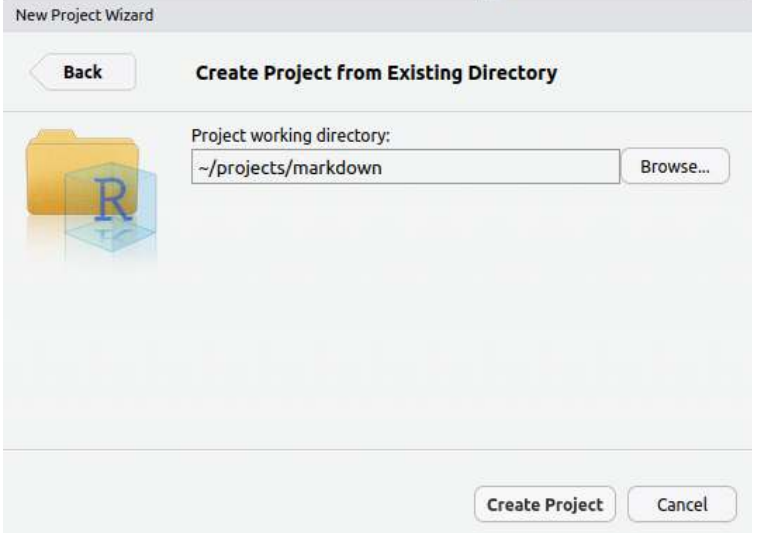
Ever had the following expression when people ask you 8 months later "*Where is that bit of analysis you did for me*": 🤔.

We want to avoid feeling like that by keeping all our *notes*, *scripts*, *data* and *output* in one single place. This is where Rstudio makes it easy by creating a project.

Start by creating a folder in your `home` directory called `projects` and starting a project called `markdown`:

```
hanjo@optimus:~$ mkdir -p projects/markdown
```

- Next click on the menu: `File > New Project`
- Then select the path `projects/markdown` as your project folder and click `Create Project`



Using Projects

Beyond having a dedicated work environment for you project, projects also have other advantages.

The biggest one of them all is *relative paths*. Ever get a document from someone and they have a link in their document, but it says something like `/Documents/Chris/my_work/data/data.csv` and now the link no longer works on your computer.

What Rstudio does is anchor the link from the project directory. So if I ever send Chris my `markdown` project, and the data is stored in `data/data.csv`, it will work on both myself and Chris' computer.

Create the following folder structure in your new project.

```
.
└── markdown
    ├── scripts
    ├── output
    └── data
        ├── raw
        └── processed
```

02:00

Software for analysis

We are also going to install some R packages to ensure that Rstudio can render our lab-books to both PDF and HTML.

write each of these lines in the command-line console of Rstudio and press `enter`. We will be diving deeper in the `R` universe later in this course. For now, just follow along with how I do it.

```
install.packages("devtools")
install.packages("rmarkdown")
install.packages("knitr")
install.packages(c("tinytex", "usethis", "rmdformats", "prettydoc"))
tinytex::install_tinytex()
devtools::install_github("holtzy/epuRate")
```

05:00

Last, but not least...

> Making your Rstudio look cool for you!

Take some time and go into preferences to choose your default color scheme that suites you. OR

Customize your own theme:

```
https://tmtheme-editor.herokuapp.com/#!/editor/theme/Monokai
```

Last, but not least...

> Making your Rstudio look cool for you!

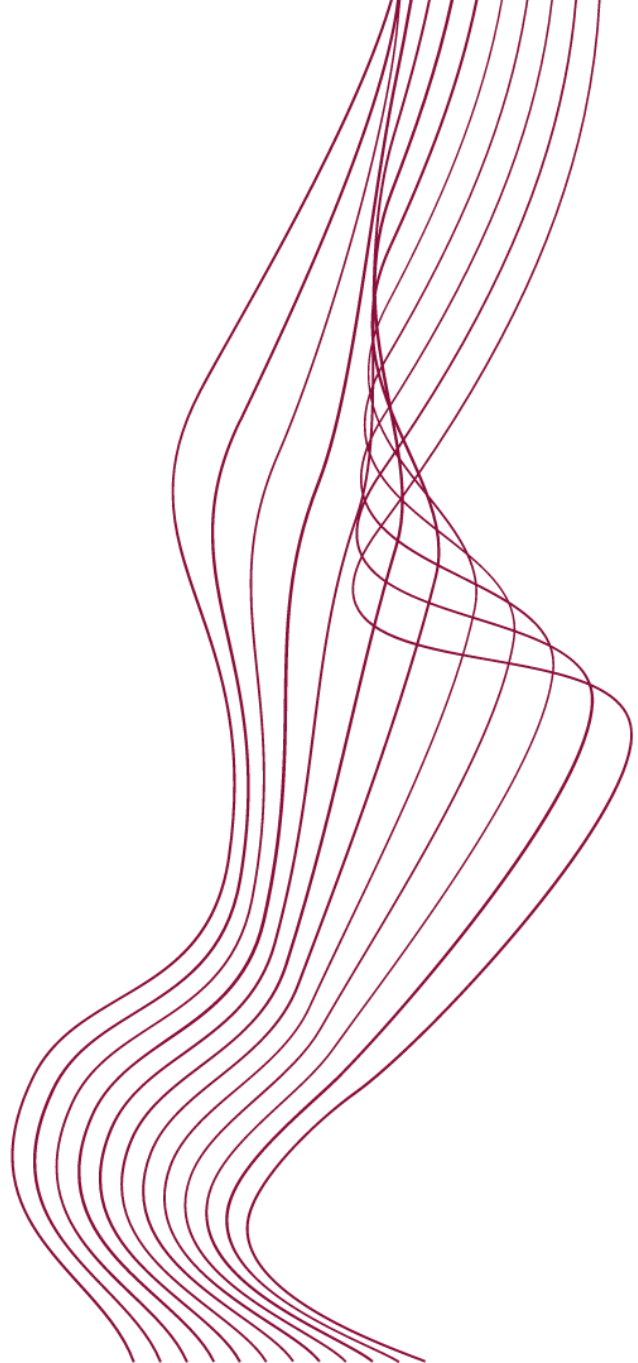
Some cool, **seriously advance**, tricks for `Rstudio` to use *ligatures*:

```
https://github.com/rstudio/rstudio/issues/2534
```

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 2 - Documenting & Databases)

Agenda

1) Homework
2) Documenting with Rstudio
3) Introduction to databases



Markdown 

What is markdown?

R Markdown wizard monsters creating a R Markdown document from a recipe. Art by Allison Horst

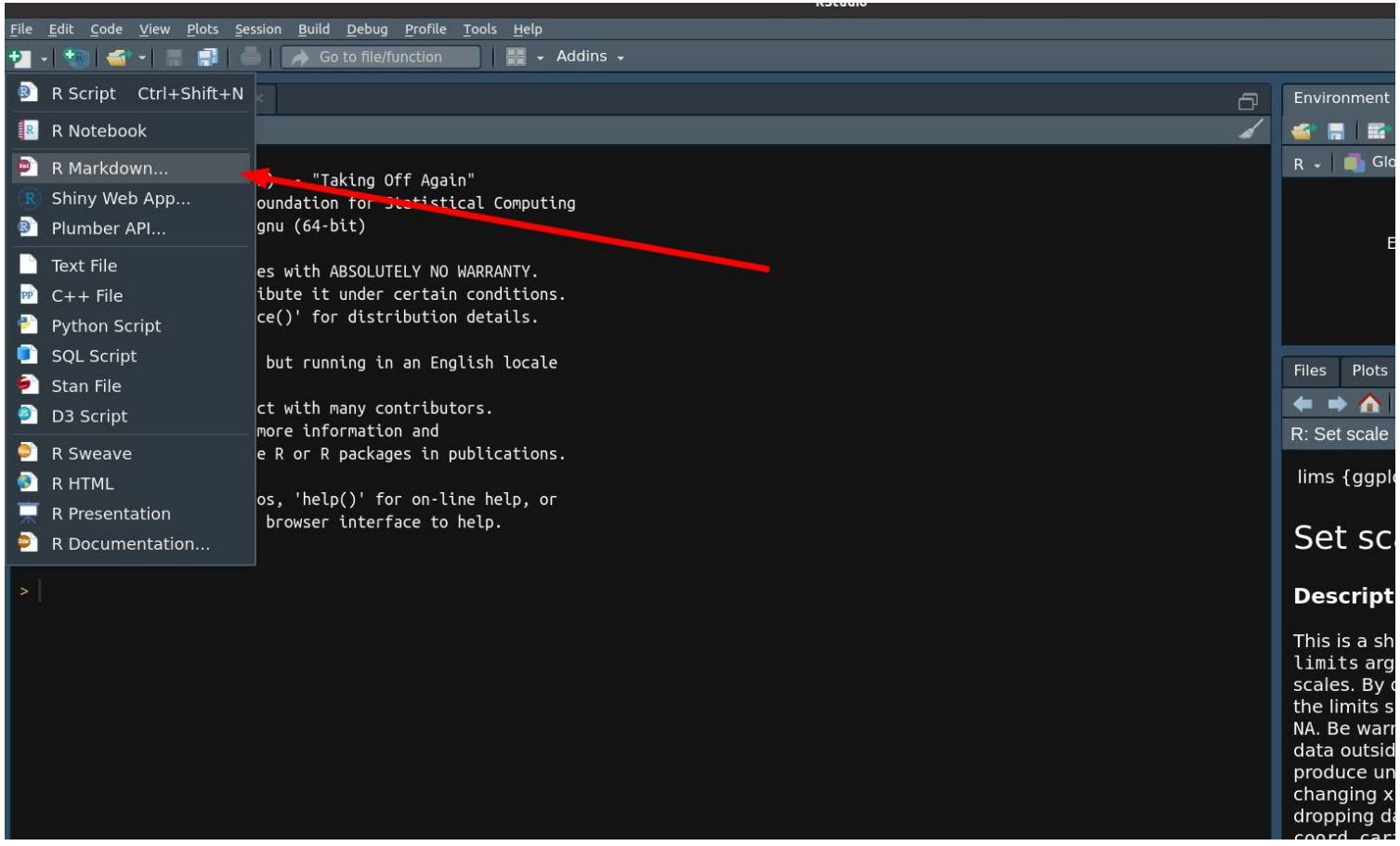
What is markdown?

> Markdown is a lightweight markup language for creating formatted text using a plain-text editor. *John Gruber* and *Aaron Swartz* created Markdown in 2004 as a markup language that is appealing to human readers in its source code form. Markdown is widely used in blogging, instant messaging, online forums, collaborative software, documentation pages, and readme files.
>
> — *Wikipedia*

- Abstraction layer *above* certain compiling formats such as PDF, HTML, Word (XML).
  - This is pretty cool as you only have to learn the very basic syntax of markdown to be able to convert your document to any of the formats.
- `Rstudio` uses a productive notebook interface (called *Rmarkdown*) to weave together narrative text and code to produce elegantly formatted output.
  - Great thing is it supports over 51 languages. Main ones are `R`, `python`, `shell` and `SQL`.

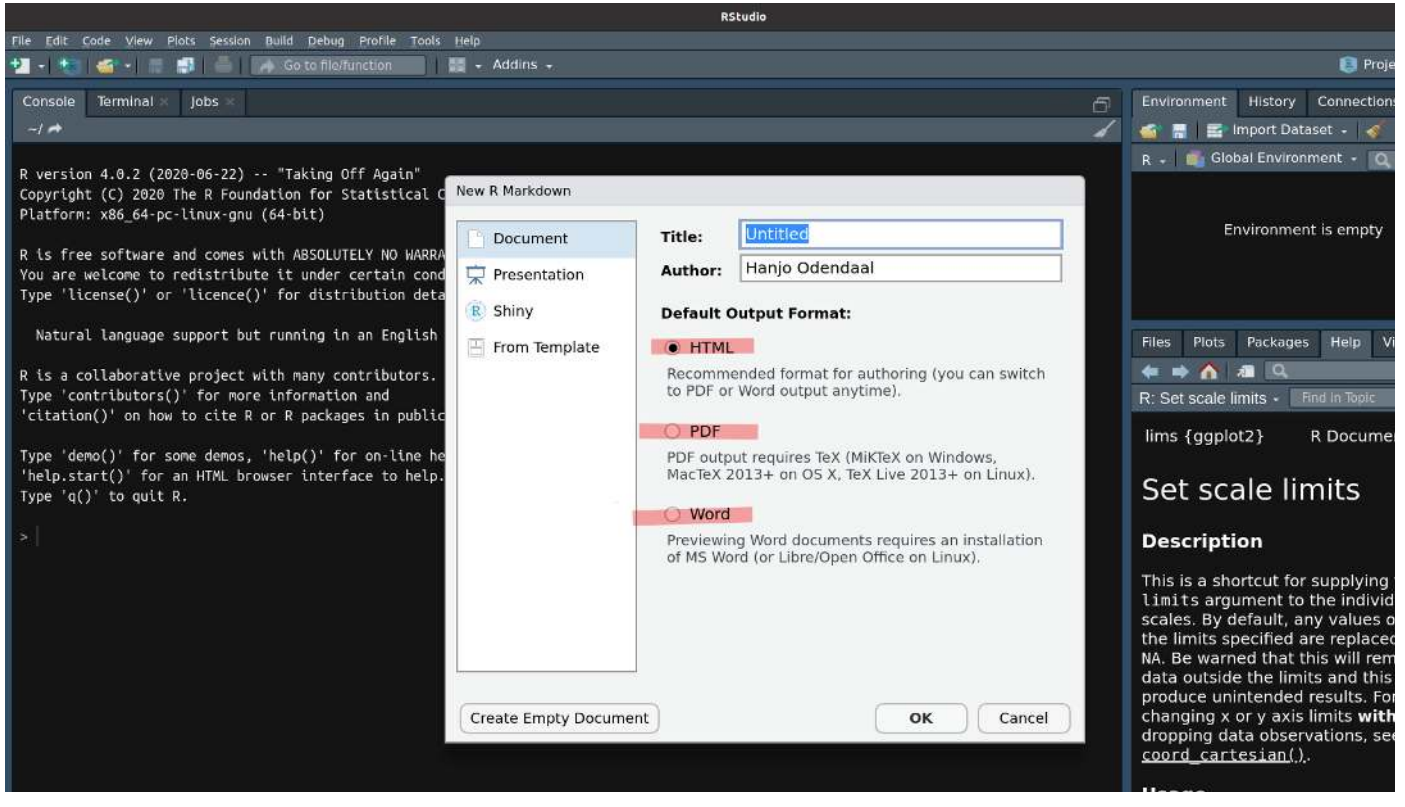
Understanding markdown in Rstudio

- Start by opening a new *Rmarkdown* file (`.rmd`) in your `markdown project`.



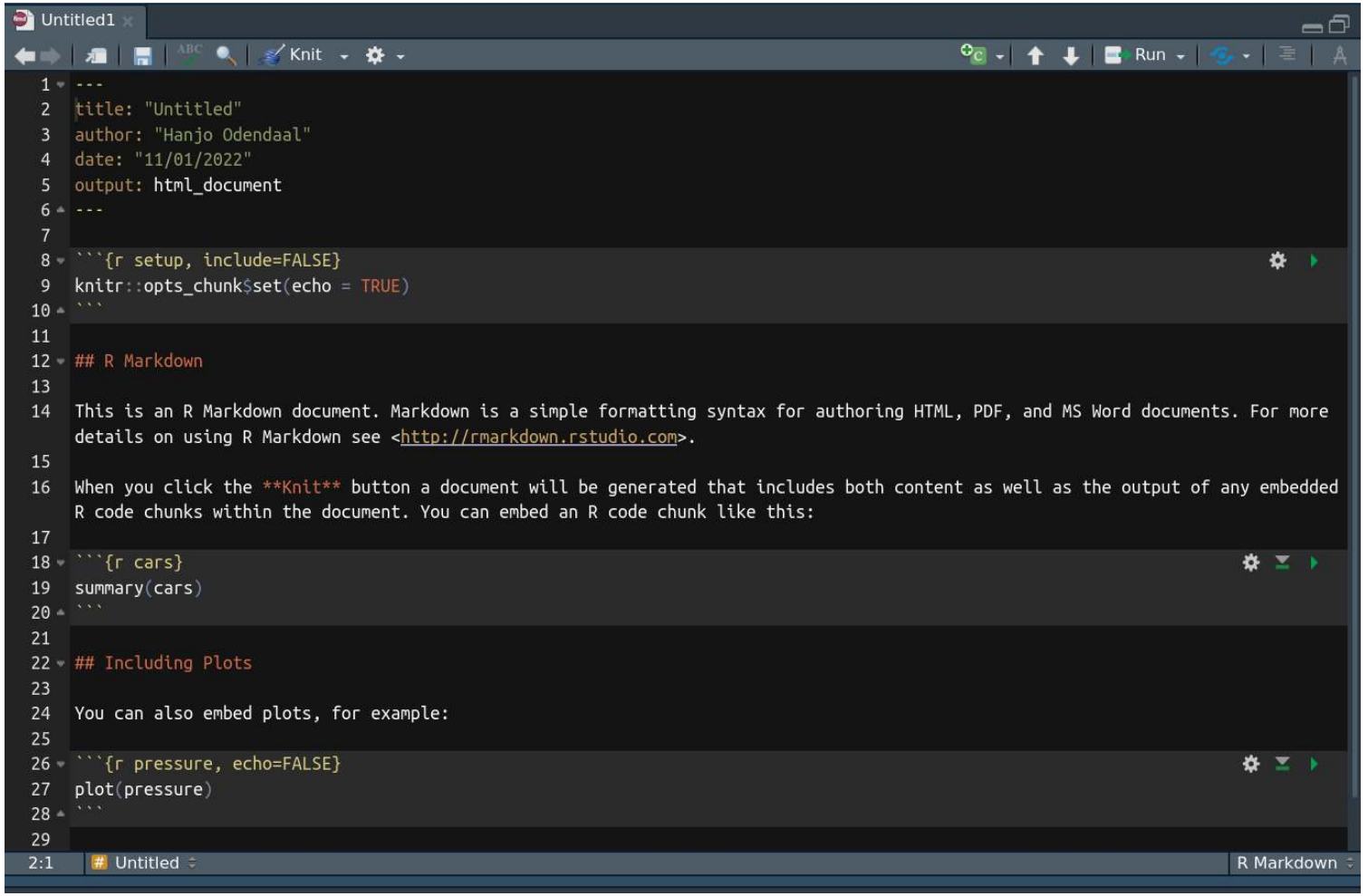
Understanding markdown in Rstudio

- Start by opening a new *Rmarkdown* file (`.rmd`) in your `markdown project`.

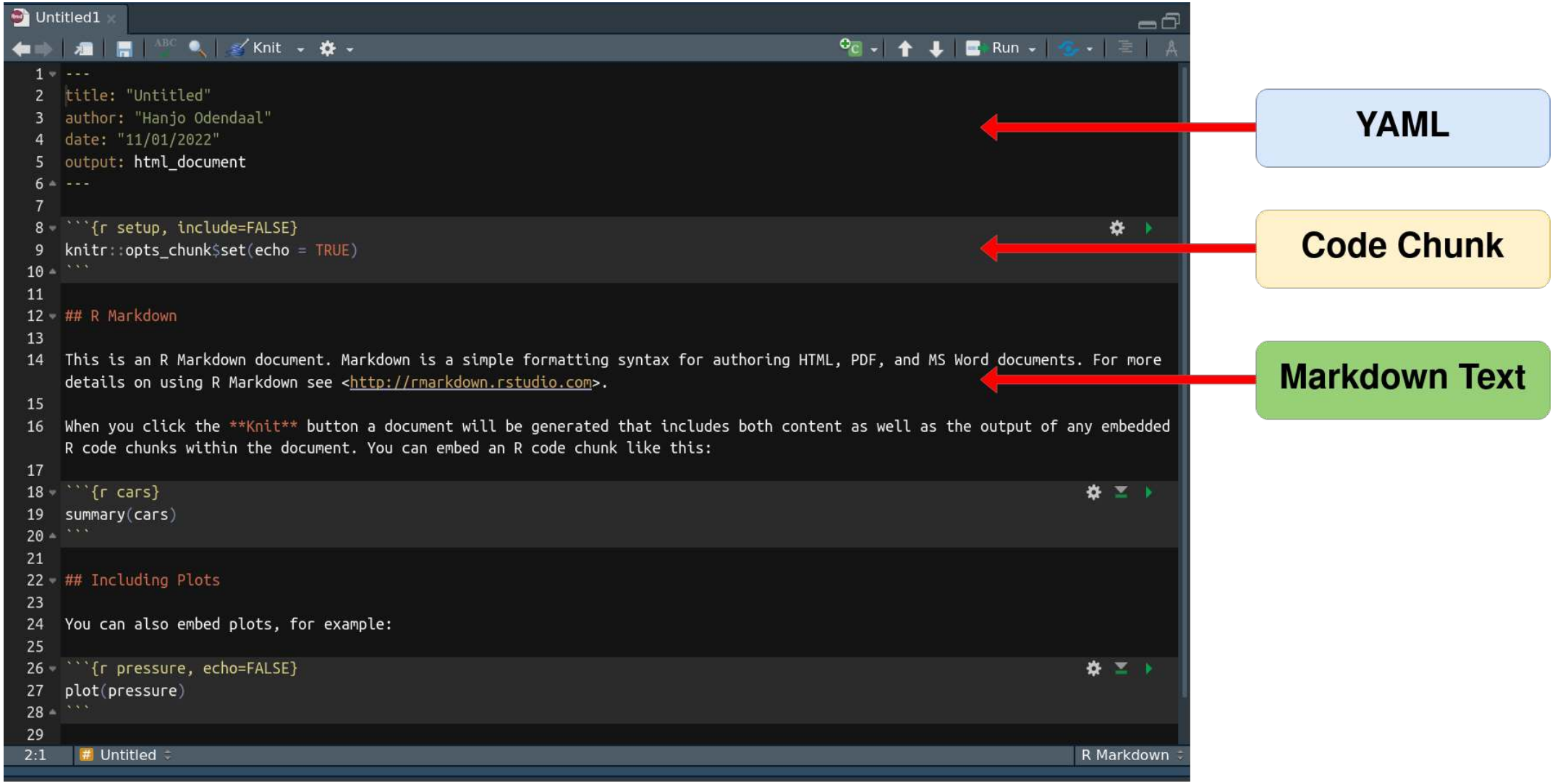


Understanding markdown in Rstudio

- Start by opening a new *Rmarkdown* file (`.rmd`) in your `markdown` project.



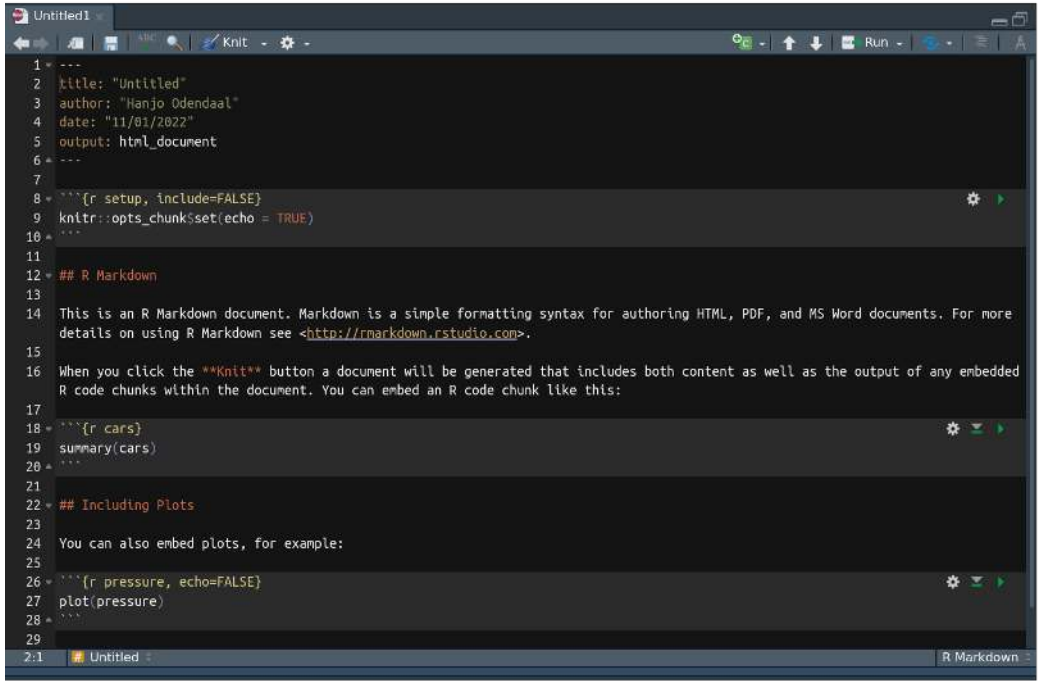
Components of markdown

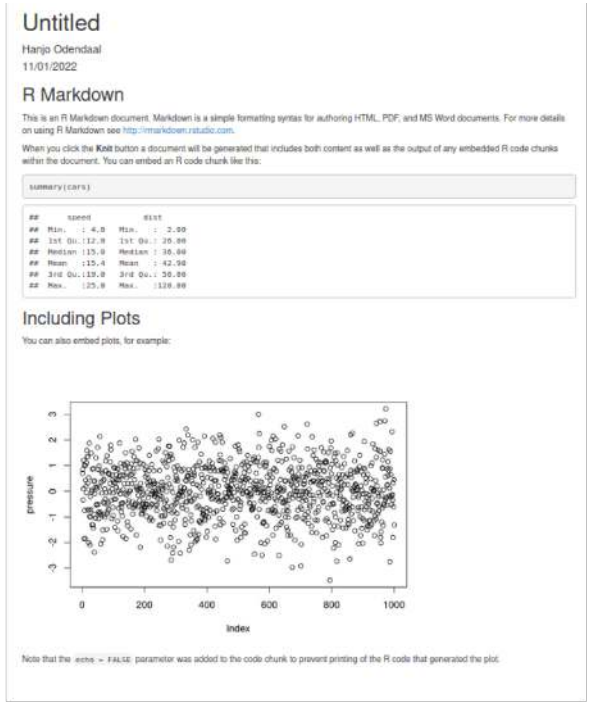


Understanding markdown in Rstudio

We need to `knit` our documents in order to produce the output.

- Save your `.rmd` document in your folder as `README.rmd`.
- Next, press the `knit` button at the top OR (be cool) and use `CTRL + SHIFT + k`!





Components of markdown: YAML

> YAML: YAML Ain't Markup Language

The *YAML* component specifies the metadata of the file:

- Type of output to produce
- Formatting preferences of things like tables
- Other metadata such as document title, author, and date.

YAML is dependent on indentation so be careful:

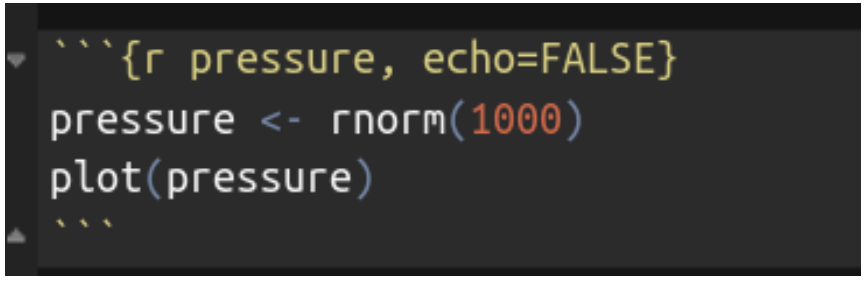
```
---
title: "My cool document"
author: "Hanjo Odendaal"
date: "11/01/2022"
output: html_document
---
```

Components of markdown: Code Chunks

Code Chunks are the sections of the document where you will write your code that you wish to include into your document.

For now, we will only use the code chunks as a documentation tool for any code that we write. Later on in the course we will actually be executing the code to produce tables and plots in a document!

Each chunk is opened with a line that starts with three back-ticks, and curly brackets that contain parameters for the chunk (`{ }`). The chunk ends with three more back-ticks.



😉 use shortcut (`CTRL + ALT + i`) to open chunk

Components of markdown: Code Chunks

What do we mean by parameters in the `{}` brackets? Lets start with the programming language specification.

- They start with `r` to indicate that the language name within the chunk is `R` (we can also do `python` or `sql` etc.)
- After the `r` you can optionally write a chunk "name" - good practice for debugging later on

The curly brackets can include other options too, written as `tag = value`, such as:

- `eval = FALSE` to not run the R code.
- `echo = FALSE` to not print the chunk's `R` source code in the output document.
- `warning = FALSE` to not print warnings produced by code.
- `message = FALSE` to not print any messages produced by code.
- `include = TRUE/FALSE` whether to include chunk outputs (e.g. plots) in the document.
- `out.width` and `out.height` provide in style `out.width = "75%"`.
- `fig.align = "center"` adjust how a figure is aligned across the page.
- `fig.show='hold'` if your chunk prints multiple figures and you want them printed next to each other (pair with `out.width = c("33%", "67%")`. Can also set `animate` to concatenate multiple into an animation.

Components of markdown: Markdown Text

> Markdown Text is what makes using it as a lab-book (and writing journal articles) so versatile.

📔 Would you believe that these slides were all made in using `Rmarkdown`?

So lets start with some basics: *Headings* and *Formatting*

```
# Header 1

## Header 2

### Header 3
```

So *how* would **this** text `look`?

```
So _how_ would **this** text `look`?
```

Components of markdown: Markdown Text

Unordered list items start with `*`, `-`, or `+`, and you can nest one list within another list by indenting the sub-list:

```
- Fruits
- Vegtables
  * Carrot
  * Spinach
```

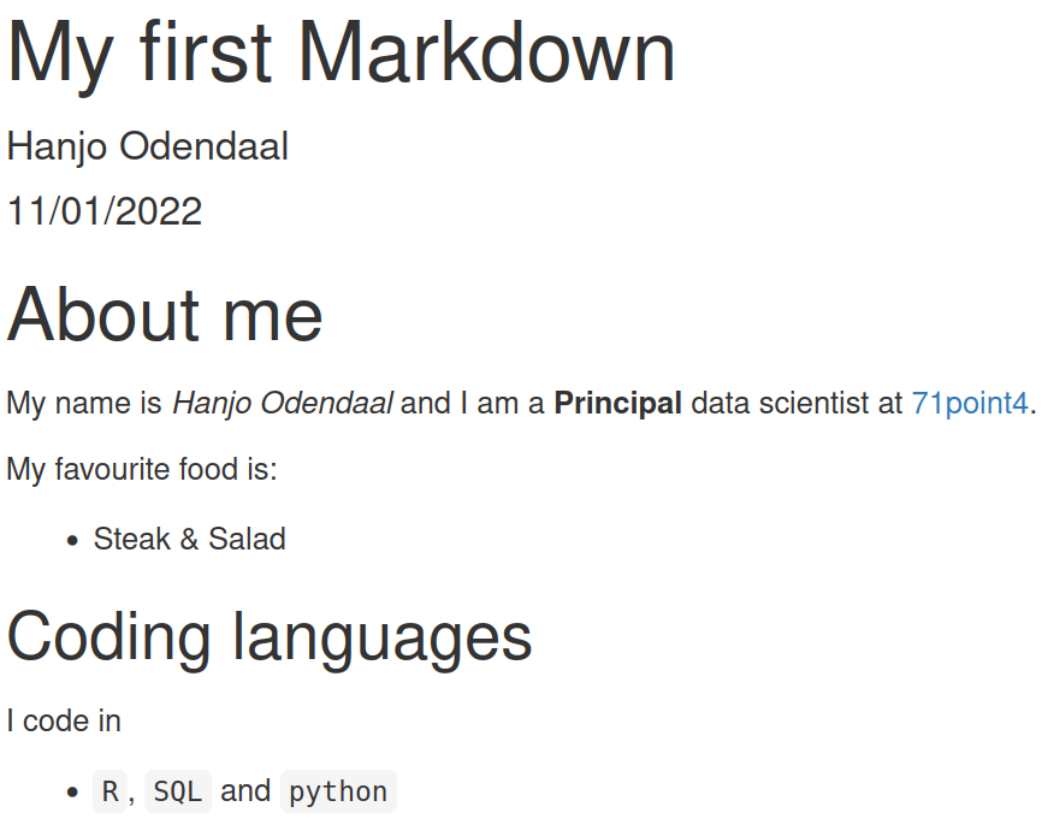
- Fruits
- Vegtables
  - Carrot
  - Spinach

```
1. Dog
    - German Shepherd #(two spaces)
    - Belgian Shepherd #(two spaces)
2. Cat
    - Siberian  #(two spaces)
    - Siamese  #(two spaces)
```

1. Dog
   - German Shepherd #(two spaces)
   - Belgian Shepherd #(two spaces)
2. Cat
   - Siberian #(two spaces)
   - Siamese #(two spaces)

Your turn!

> Can you produce the following document?



My first Markdown

Hanjo Odendaal

11/01/2022

About me

My name is *Hanjo Odendaal* and I am a **Principal** data scientist at 71point4.

My favourite food is:

- Steak & Salad

Coding languages

I code in

- `R`, `SQL` and `python`

10:00

Changing formats

- How does the following code affect your output?

```
---
title: "My first Markdown"
author: "Hanjo Odendaal"
date: "11/01/2022"
output: pdf_document
---
```

- Lets change the output to a `Word` document:

```
---
title: "My first Markdown"
author: "Hanjo Odendaal"
date: "11/01/2022"
output: word_document
---
```

Using advanced YAML

If we are *knitting* a document to `html` there are a couple of really cool things we can do in terms of formatting.

- How does the following code affect your output?
- All available themes: "cerulean", "cosmo", "flatly", "journal", "lumen", "paper", "readable", "sandstone", "simplex", "spacelab", "united", and "yeti".

```
---
title: "Your title here"
date: "Todays date"
output:
  html_document:
    theme: journal
    highlight: espresso
    toc: true
    toc_depth: 4
    toc_float: true
    code_folding: show
---
```

Using advanced YAML

We can also combine our outputs:

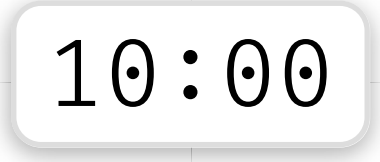
- Some of the `YAML` options might not be available for when you want to switch between formats. (example `PDF` does not take `theme` as a parameter)
- To account for those differences, we split up the yaml parameters between the different formats.

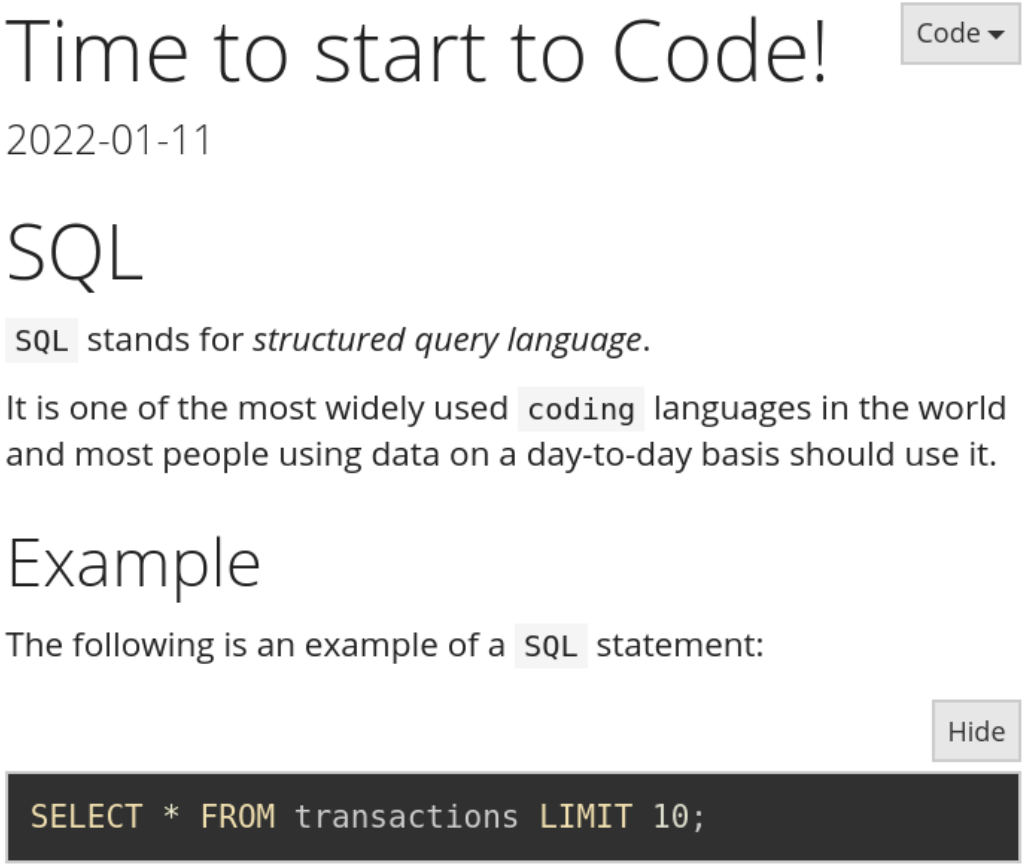
```
---
title: "Your title here"
date: "Todays date"
output:
  pdf_document:
    highlight: espresso
    toc: true
    toc_depth: 4
  html_document:
    theme: journal
    toc: false
    highlight: haddock
---
```

Your turn!

> Create the following output with a theme and format of your choice.

⚠️ Remember to use chunk option `eval = FALSE` & `echo = TRUE` to ensure code **doesn't** run, but is displayed.

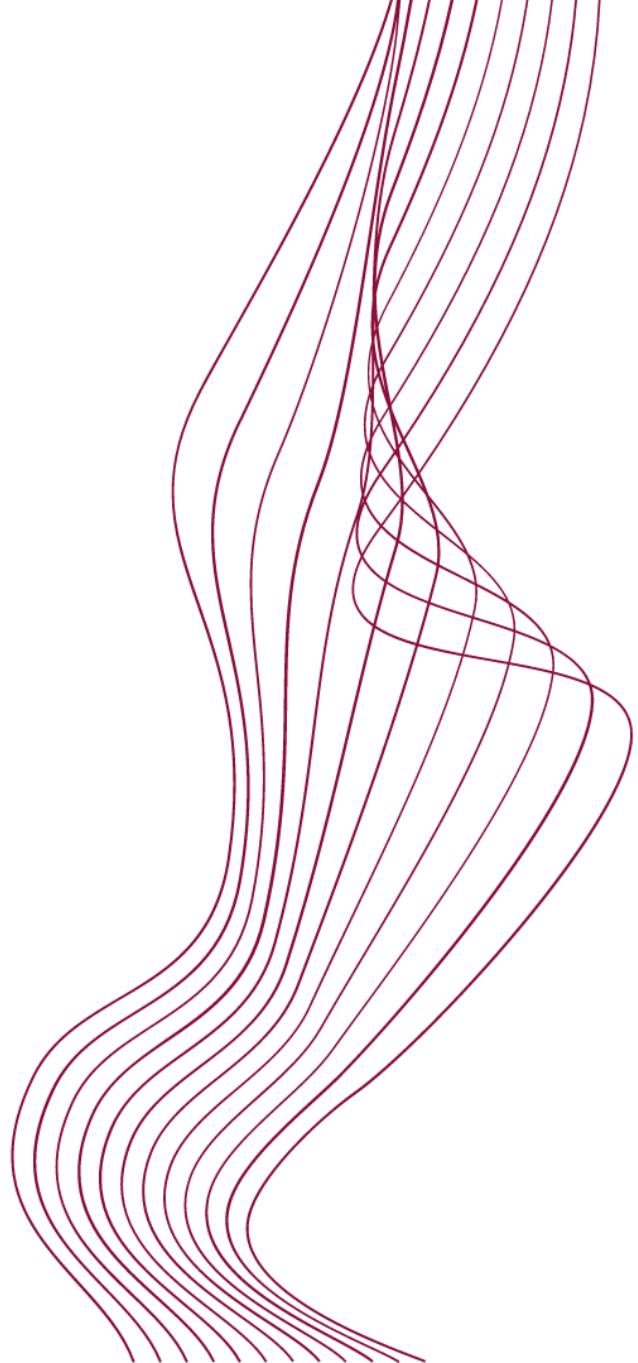




Foundation of Data-Driven Analysis for Policy Decisions Course (Day 2 - Documenting & Databases)

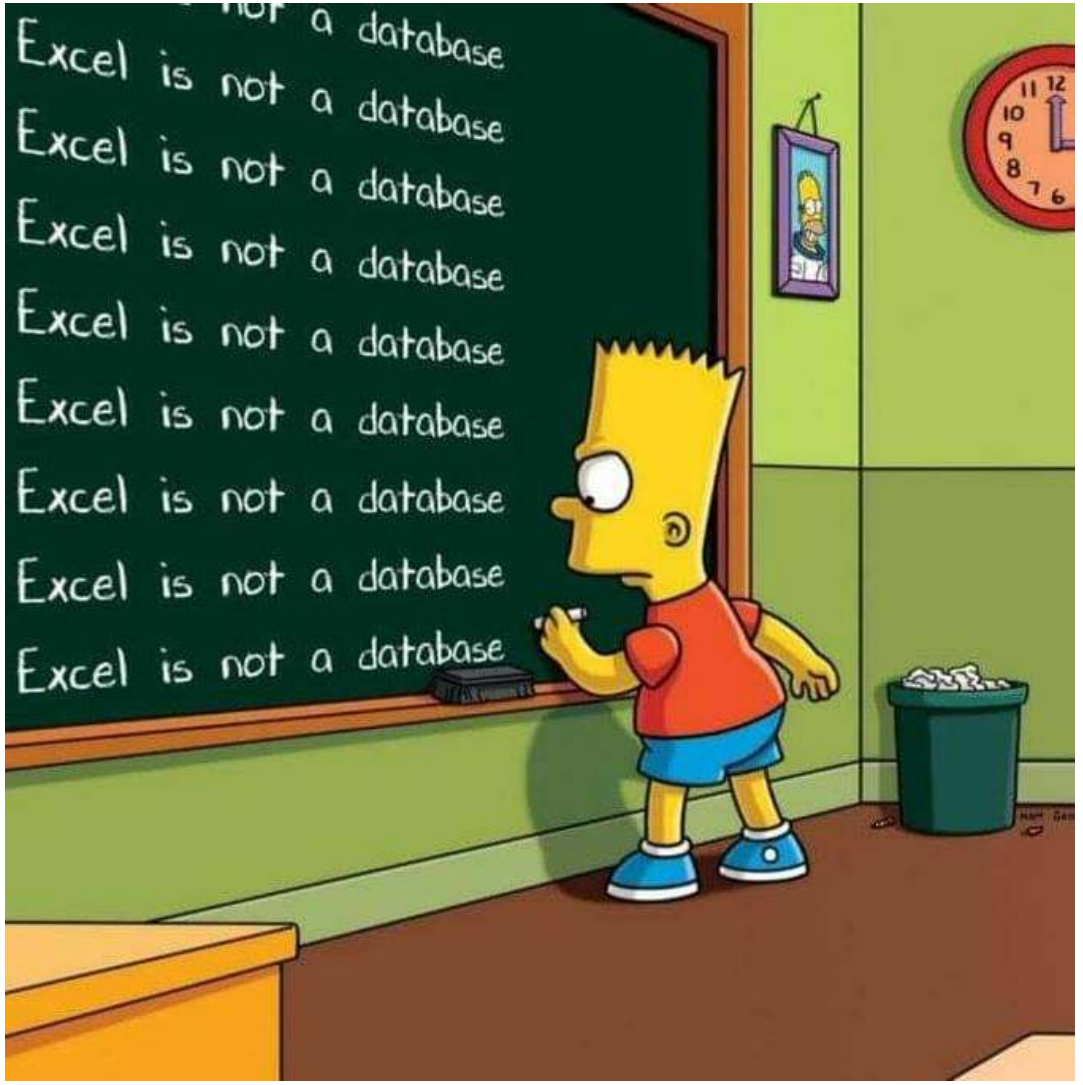
Agenda

1) Homework
2) Documenting with Rstudio
3) Introduction to databases



Databases design 

Why do we use databases?



Why do we use databases?

Although there are many reasons to shift from using Excel to databases, the most important cornerstones are: (i) Data Integrity, (ii) Redundancy, (iii) Error prone, (iv) User Access and Security and (v) Data Accessibility and Speed.

Data Integrity:

- There are a set of rules that govern the structure of the data, how different information relate to one another and also what input can be put into a certain column of a table (ex. numeric vs string).
- These rules and protocols build a general framework under which everyone must adhere to, and as such, increases reliability of the data.
- It also removes questions like "*What do I do when I get new data? Where do I store that data? How must the data look?*".
- Excel is limited to 1,000,000 rows of data in a single sheet. If you operating at close to the edge of that amount of data how difficult to you think it is to ensure the quality of the information?

Why do we use databases?

Redundancy:

- I think all of us know version control of files that are called `final_analytics_tom_v2.3_client_clean.xlsx`. These files eventually just become copies of the same data with small changes that are difficult to track and keep clean.
- Using relational databases also ensures that we separate information out across tables to ensure we do not have multiple versions of the same data in different places. Example would be to separate out customers and their purchases.
    - When updating a customer's information, we only have to update the customers table and not the purchases tables, as the purchases table only links back to the customer via a unique id (also called a key) of some sort.

Error prone:

- Excel sheets (as we saw earlier) is very much susceptible to proliferation of errors, especially when the data gets large. There is no way of know what changed and how when someone accidentally overwrites a cell/row/column.
- Also, because sheets are usually linked together, if the sheet changes unexpectedly (perhaps someone added a column), suddenly the formulas are no longer correct or links to the data are broken.

Why do we use databases?

Access and Security:

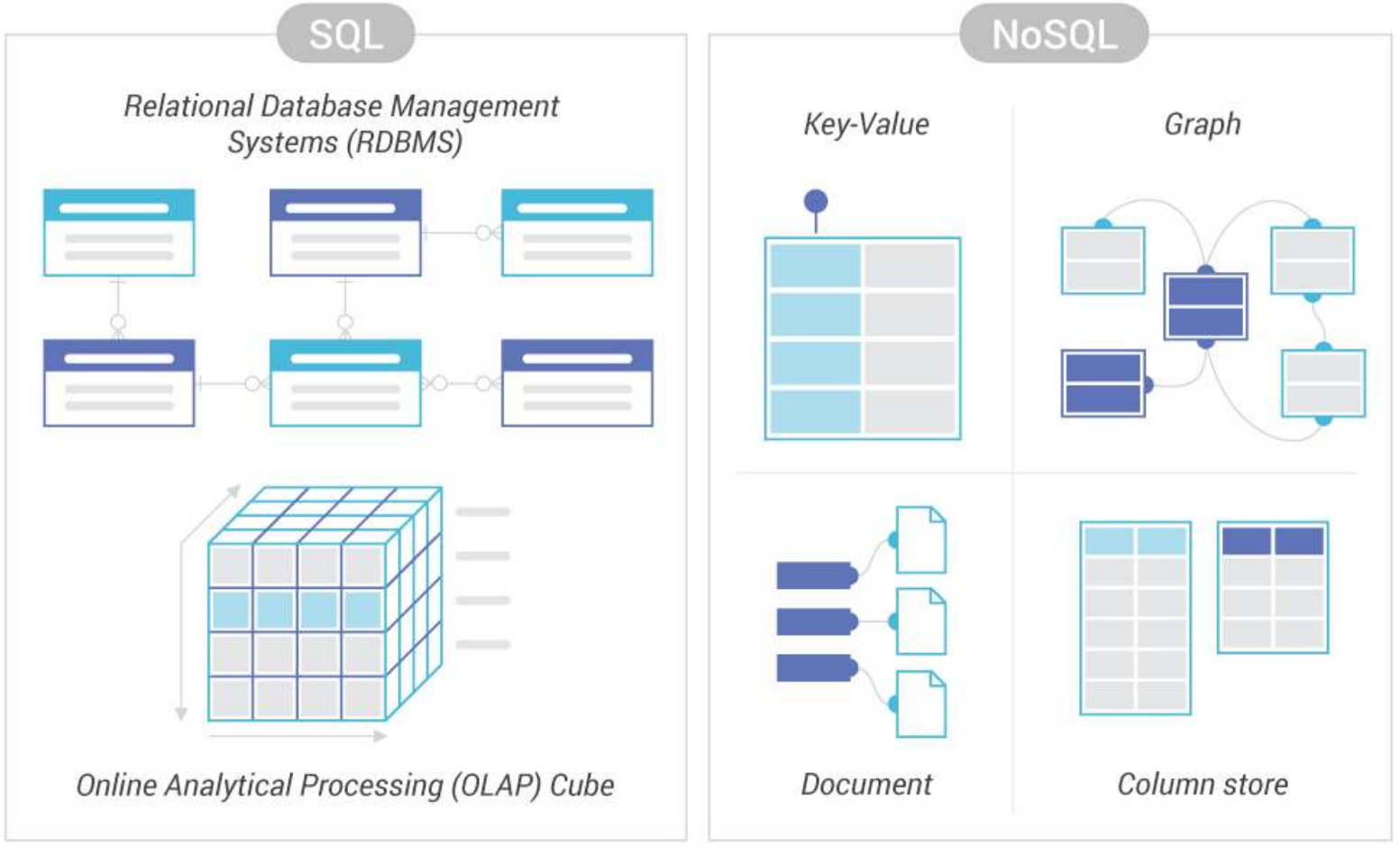
- Multiple users can work on a database at the same time. Most of the time end-users only need to collect or aggregate data for their purposes, which means we can create specials user that are only allowed those operations. This ensures that the risk of unknowledgeable users do not accidentally change the underlying data without them knowing.
- With personal information protection laws coming into force, one also wants to restrict access to certain types of information by only granting access to those who have the right clearance[1].

Data Accessibility and Speed:

- Ain't no one got time to work on an Excel sheet over 10,000 rows. Having to quickly analyse information using aggregation tools such as *pivot* or *vlookups* becomes a total nightmare. Excel is dynamic, which means every action causes all the information to automatically recalculate.
- Databases allows you to do deep analysis of data over millions (even billions) of rows of data in seconds.
- Because you most likely will be using a relational setup in a database, you only query the data you need, not the whole information set.

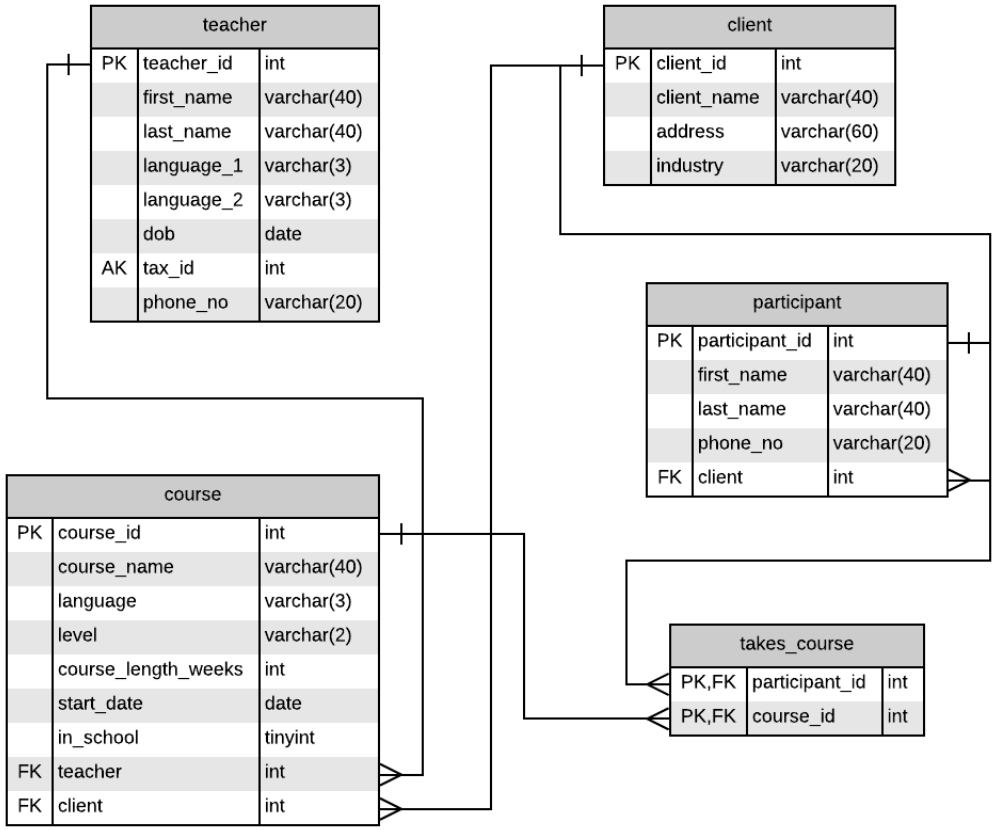
[1] *Please do NOT ever send personal information in plain text in Excels spreadsheets over email, unless you want to end up in jail.* 💀

Different database types



*link to original website

RDMS database example data schema



*link to original website

Different database structures

> Do note, that although we do not cover OLAP database in this course, they tend to be a little bit different as they try to avoid complex joins which could slow down analytics.

Database structures, or *schema* design, depends a lot on the application of the database. Although there are different schemas and designs, they do have some common traits:

- Includes the name of the fields in the table.
- The type that the field consists of (numeric, date, varchar etc.).
- Associations and keys linking fields.

Common schemas that you might encounter are:

- Star schema
- Snowflake schema
- Fact constellation

Different database structures

To understand schemas a bit better, we need an understanding of the pieces. The two most important components consist of: `fact` and `dimension` tables.

FACT:

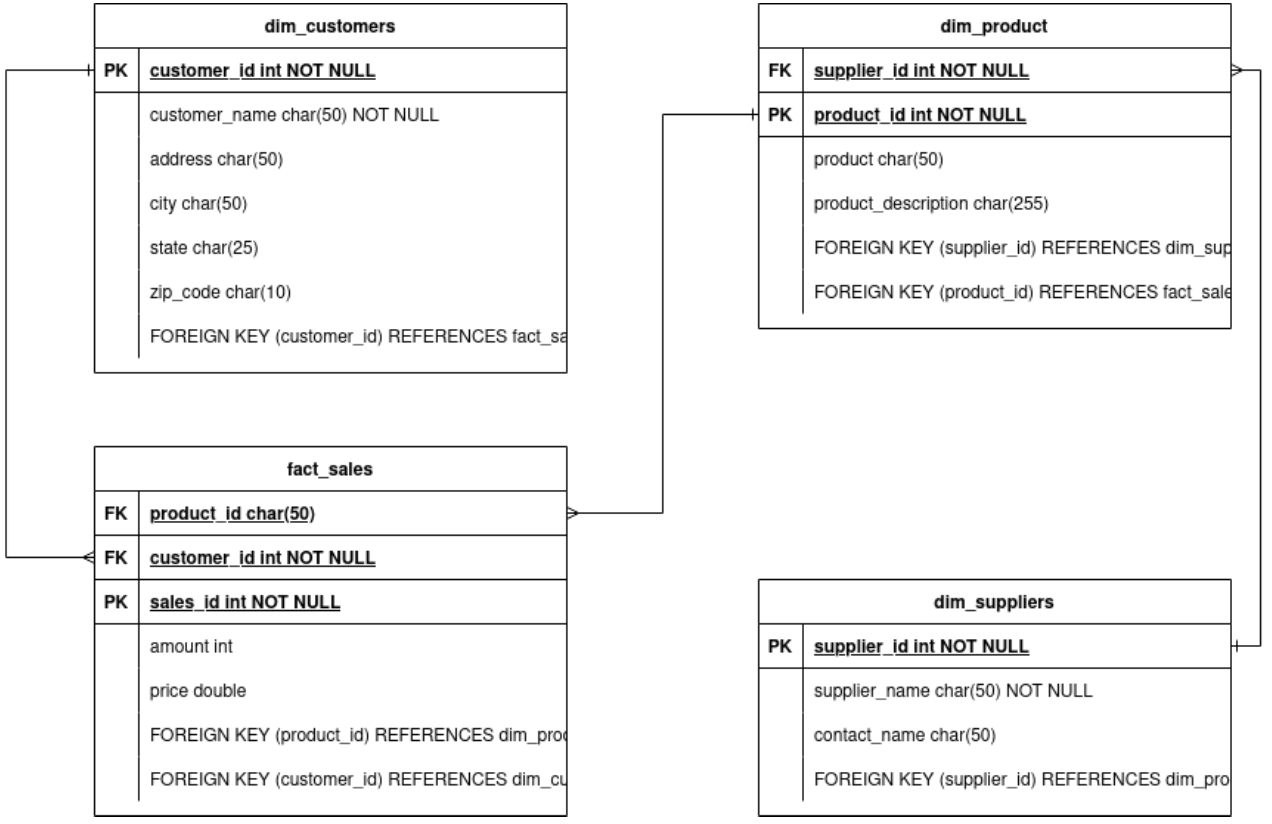
- Fact table contains measurements, metrics, and *facts* about a business process
  - ex. *sales* or *webpage visits*.
- Fact tables form the primary table of the design and are usually normalized
  - We assign a numerical number or code to an attribute for better performance
  - An example of this would be where we code GENDER as 1 for male and 2 for female.

DIMENSION:

- Provides the information about the facts
  - ex. *location* of transaction, *customer*
- These tables are de-normalized and have to be joined to the fact table table before analysis can happen.
- The tables also usually contain descriptions of the field in order to make it easier to understand.

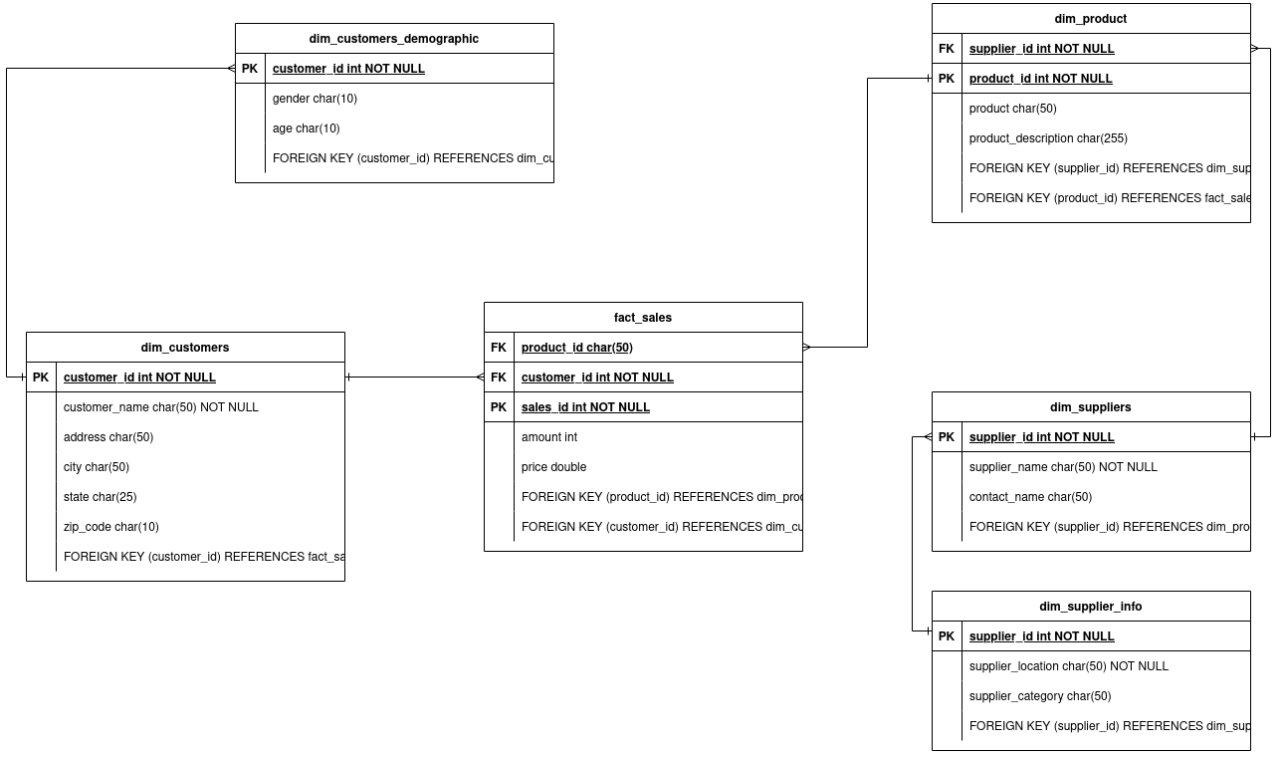
Different database structures: Star Schema

> Star schema design has only *one* fact table and multiple dimension tables. This is a very common design as its relational properties are easily understood.



Different database structures: Snowflake Schema

> Snowflake schema design extends the Star Schema by only *one* fact table, multiple dimension tables, each with their own dimension tables. This adds another layer of abstration to the dimension tables and could contain additional information about attributes not in use every day.



Loading data into SQL 🔎

Jumping in with both feet

> In the advanced Database course we will cover databases in much more depth. But for now, lets focus on getting you writing SQL queries and leave the database setup to the DBA.

Lets start with writing the most basic of all commands, `CREATE DATABASE {name}`.

I have already uploaded some data for you under the `data/` folder. If we use `head transactions.csv` we can see what the data format looks like:

```
transaction_date,transaction_id,customer,sku,amount
2020-10-11,txn-284-06765,cust42565,airtime,127.98253587552975
2020-08-06,txn-218-02867,cust509209,airtime,158.5837870902482
2020-12-21,txn-355-04261,cust77963,airtime,203.87837566253032
```

⚠️ This is a **synthetic** dataset I created from scratch! Later in the course we will work with some *real* economic data.

The transactions table consists out of 1,061,923 rows, so its not going to fit in excel.

Steps to create a database

The first step is to enter MySQL[1]. (to exit press `CTRL + D`)

```
hanjo@optimus:~$ mysql
```

You should see something like this:

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.1.48-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

The `MariaDB [(none)]>` is the console interface and indicates that we have not selected a Database. Run `show databases` and tell me what you see.

[1] *We are actually using MariaDB, an open-source version of MySQL. Go read up on the fascinating history if you are interested.*

Steps to create a database

Lets create a database to house our tables in for the rest of the workshop. I am going to call this database: `workshop`, I suggest you do the same.

```
MariaDB [(none)]> CREATE DATABASE workshop;
Query OK, 1 row affected (0.00 sec)
```

⚠️ Important, notice the `;` at the end of the command. This signifies to MySQL that, that is the end of my command. Be careful not to forget that in all your SQL commands.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| workshop           |
+--------------------+
4 rows in set (0.00 sec)
```

Steps to create a database

Now that you have your database created, there are two ways to access it:

From inside MySQL using the command `use workshop`

```
MariaDB [(none)]> use workshop;
```

OR from the command-line:

```
hanjo@optimus:~$ mysql workshop
```

Both results show bring you to the interface where you can query what tables are in the database:

```
MariaDB [workshop]> show tables;
Empty set (0.00 sec)
```

Steps to create a tables

In order to upload our synthetic transactions data set, we need to create the table structure into which the data must go.

These structures can become quite complex, but for now we only going to create a *fact* and *dimension* table (transactions and customer information respectively). We allocate our `PRIMARY KEY` as the date of transaction, the id and the customer who performed the transaction.

```
CREATE TABLE transactions(
  transaction_date DATE,
  transaction_id  VARCHAR(20),
  customer VARCHAR(20),
  sku VARCHAR(100),
  amount DOUBLE,
  PRIMARY KEY(transaction_date, transaction_id, customer)
)
;
```

🔐 Keys are an important feature which can optimize looking up a transaction and also ensuring performance while maintaining data integrity. We don't cover keys in this course, but its something to be aware of, especially when we start learning about joins.

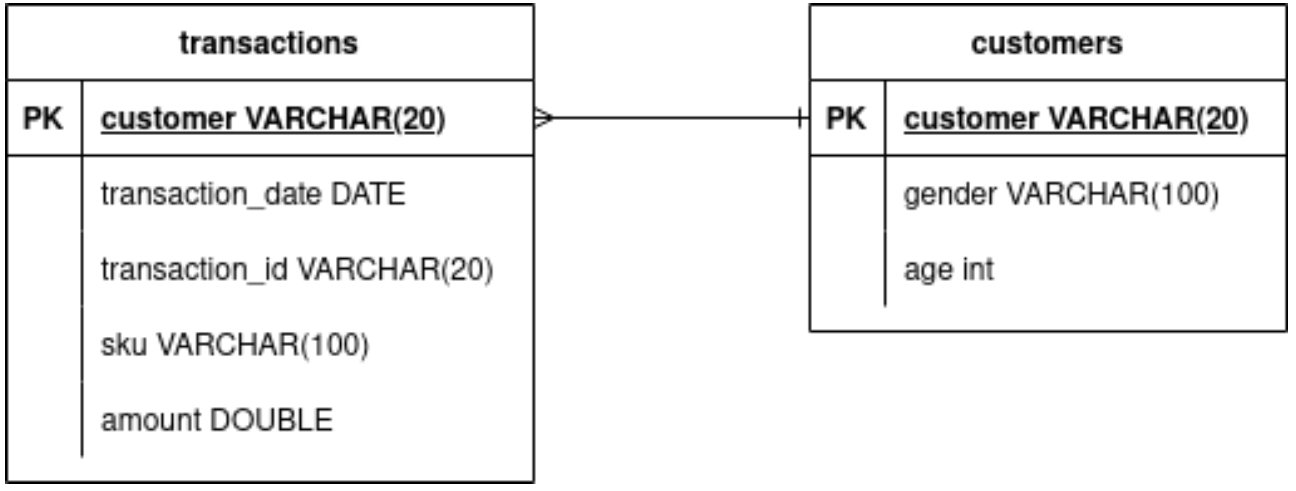
Steps to create a tables

Now we gonna load the customer information. This table contains the demographic information associated with each customer.

```
CREATE TABLE customers(
  customer VARCHAR(20),
  gender VARCHAR(100),
  age int,
  PRIMARY KEY(customer)
)
;
```

From the two `CREATE TABLE` statements you can see that the two tables are linked via the customer column. We will be learning `JOINS` near the end of today, which will *join* the tables together so that we can get demographic information on transactions.

Our final transactions database




```
MariaDB [workshop]> SHOW FULL TABLES;
+--------------------+------------+
| Tables_in_workshop | Table_type |
+--------------------+------------+
| customers          | BASE TABLE |
| transactions       | BASE TABLE |
+--------------------+------------+
2 rows in set (0.00 sec)
```

Our final transactions database

The `explain {table}` commands helps us to understand a little bit more of the meta data of the tables:

```
MariaDB [workshop]> explain customers;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| customer | varchar(20)  | NO   | PRI | NULL    |       |
| gender   | varchar(100) | YES  |     | NULL    |       |
| age      | int(11)      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
MariaDB [workshop]> explain transactions;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| transaction_date | date        | NO   | PRI | NULL    |       |
| transaction_id   | varchar(20)  | NO   | PRI | NULL    |       |
| customer         | varchar(20)  | NO   | PRI | NULL    |       |
| sku              | varchar(100) | YES  |     | NULL    |       |
| amount           | double       | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
```

Final step: Load data

Final step: Load customers

⛏ To speed up the upload, we gonna drop the keys and then create them again after data has been uploaded:

```
ALTER TABLE customers DISABLE KEYS;
```

The `LOAD DATA INFILE` statement loads data from a text file. It's a versatile SQL statement with several options and clauses. It also comes with a handy `SHOW WARNINGS;` functions that we run afterwards if there are any warnings generated when we upload.

```
LOAD DATA LOCAL INFILE '/home/ubuntu/data/transactions/customers.csv'
INTO TABLE workshop.customers
FIELDS TERMINATED BY ','
;
```

Remember to `ENABLE` your keys after loading:

```
ALTER TABLE customers ENABLE KEYS;
```

Final step: Load transactions

⛏ To speed up the upload, we gonna drop the keys and then create them again after data has been uploaded:

```
ALTER TABLE transactions DISABLE KEYS;
```

The `LOAD DATA INFILE` statement loads data from a text file. It's a versatile SQL statement with several options and clauses. It also comes with a handy `SHOW WARNINGS;` functions that we run afterwards if there are any warnings generated when we upload.

```
LOAD DATA LOCAL INFILE '/home/ubuntu/data/transactions/transactions.csv'
INTO TABLE workshop.transactions
FIELDS TERMINATED BY ','
IGNORE 1 LINES
;
```

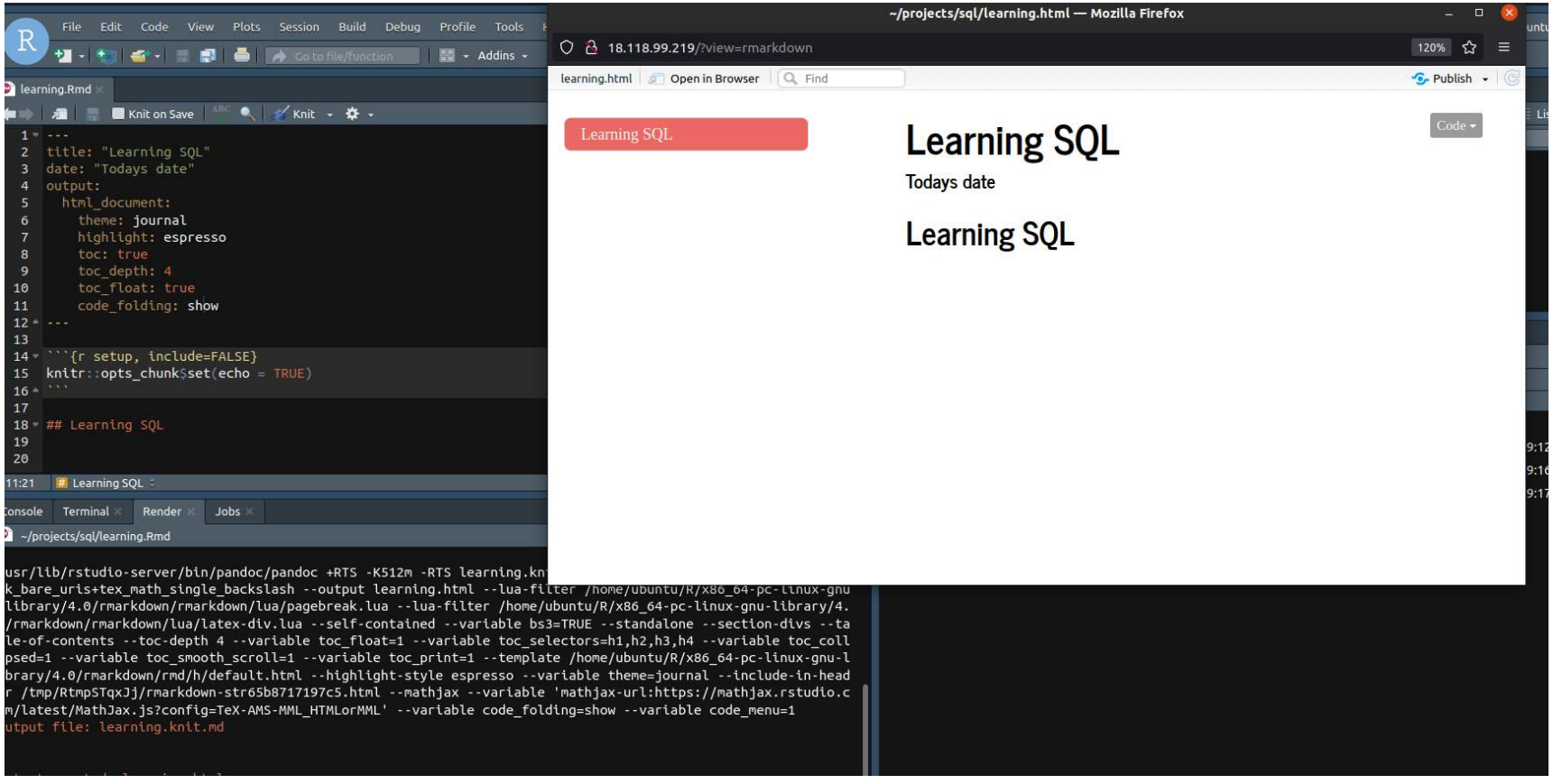
Remember to `ENABLE` your keys after loading:

```
ALTER TABLE transactions ENABLE KEYS;
```

Basic SQL ⚒

Start your notebook for this section.

Open a new `Rmarkdown` document with the name: *class_sql.rmd* in a new project folder called `projects/sql`:



The KING of all statements: SELECT

> The way to think about SQL is in terms of english commands. Also, also start from the inside and work your way out (you will see what I mean).

⚠️ It is good practice to always end your statements with `LIMIT 10` until you are sure that the correct results is returned. Working on billion row tables and forgetting to limit your results can crash tables.

Lets start with the two statements you will most likely use every day:

- Counting how many rows there are

```
SELECT COUNT(*) FROM transactions LIMIT 10;
SELECT COUNT(*) FROM customers LIMIT 10;
```

- Getting a 10 row sample

```
SELECT * FROM transactions LIMIT 10;
SELECT * FROM customers LIMIT 10;
```

The KING of all statements: SELECT

Previously I decided I wanted to return all the columns (`*`), but what if I only want to return one or two of the columns?

```
SELECT {column1}, {column2} FROM table LIMIT 10;
```

Lets only return the customer id and the amount:

```
SELECT customer, amount FROM transactions LIMIT 10;
```

👍 It is good practice to not have *long* SQL statements in one row.

> Code Needs a lot of whitespace.
> That is how it breaths
>
> — Roger Peng, Jenny Bryan, useR 2018

The KING of all statements: SELECT

Lets build a *bigger* `SELECT` statement (I like 3 tab indentation):

```
SELECT
	transaction_date,
	customer,
	sku,
	amount
FROM
	transactions
LIMIT 10;
```

SELECT but with filter criteria

What happens if we only want to return transactions of a certain type?

Well, then we can employ the `WHERE` statement. We are going to collect the same columns as previously, but now we will specify the `WHERE` criteria on `sku` column where equal to `airtime`:

```
SELECT
      transaction_date,
      customer,
      sku,
      amount
FROM
      transactions
WHERE
      sku = 'airtime'
LIMIT 10;
```

- In your notebook, write the code to bring back 100 examples where the `sku` is `p2p` and `ORDER BY` transaction_date.

10:00

SELECT but with filter criteria and order

In certain circumstances, it is necessary to order your date to get the correct output. For instance if we want to get the top 10 largest amounts:

```
SELECT
      transaction_date,
      customer,
      sku,
      amount
FROM
      transactions
ORDER BY
      amount DESC
LIMIT 10
;
```

Aggregations (Pivoting) in SQL

Pivoting forms part of the aggregation function of SQL. This helps us answer questions like:

- What is the average amount of spent per gender?
- Total value and volume per date?
- Total volume and value dissagregated by gender and age?

As you can see, aggregations or `GROUP BY` clase gets used OFTEN, so learn it well and get comfortable with it.

Aggregations (Pivoting) in SQL

Lets illustrate a basic example before moving onto complex queries. What is the total value per date?

```
SELECT
      transaction_date,
      ROUND(SUM(amount))
FROM
      transactions
GROUP BY
      transaction_date
ORDER BY
      transaction_date DESC
LIMIT 10
;
```

Once you are comfortable that you have the query correctly specified, drop the `LIMIT` and scroll through your magnificent piece of work! (Tip: press `q` to exit the viewer)

Aggregations (Pivoting) in SQL

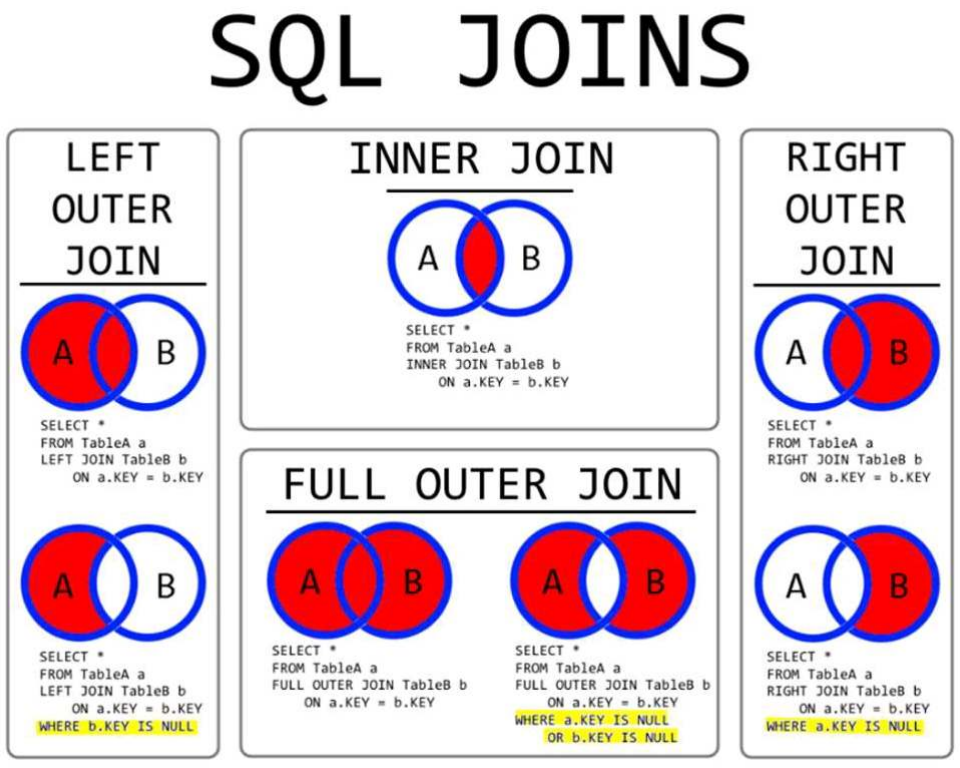
> What is the total value, volume and distinct customers doing p2p per day?

```sql
SELECT
      transaction_date,
      ROUND(SUM(amount)) value,
      COUNT(*) volume,
      COUNT(DISTINCT customer) distinct_cust
FROM
      transactions
WHERE
      sku = 'p2p'
GROUP BY
      transaction_date
ORDER BY
      transaction_date DESC
LIMIT 10
;
```

Notice how I `ALIAS` my aggregations as {aggregation} then {name}. This will make your life a lot easier and in some case it is mandatory... as in joins.

Last but not least: JOINS

By now you are asking yourself, if we designed our database in the beautiful star schema that we talked about earlier, how do we *join* all the information together again? This is where `JOINS` come in and there are a multitude of them. Most important one is `LEFT JOIN` and `INNER JOIN`:



Last but not least: JOINS

Lets attempt a basic join before we combine joins with aggregations. To start off we will `JOIN` the `customers` table onto the `transactions` table:

```
SELECT * FROM transactions AS trans
LEFT JOIN customers as cust
ON trans.customer = cust.customer
LIMIT 10
;
```

There are ways to optimize your joins to be extremely fast.

- One is keys (which is why we used primary keys in our tables).
- Another is query optimization through column selection and subqueries.
    - Although we do not cover these in this course, having knowledge of advance backend mechanics can sometimes take your execution time from days to minutes.

Last but not least: JOINS

Last but not least: JOINS

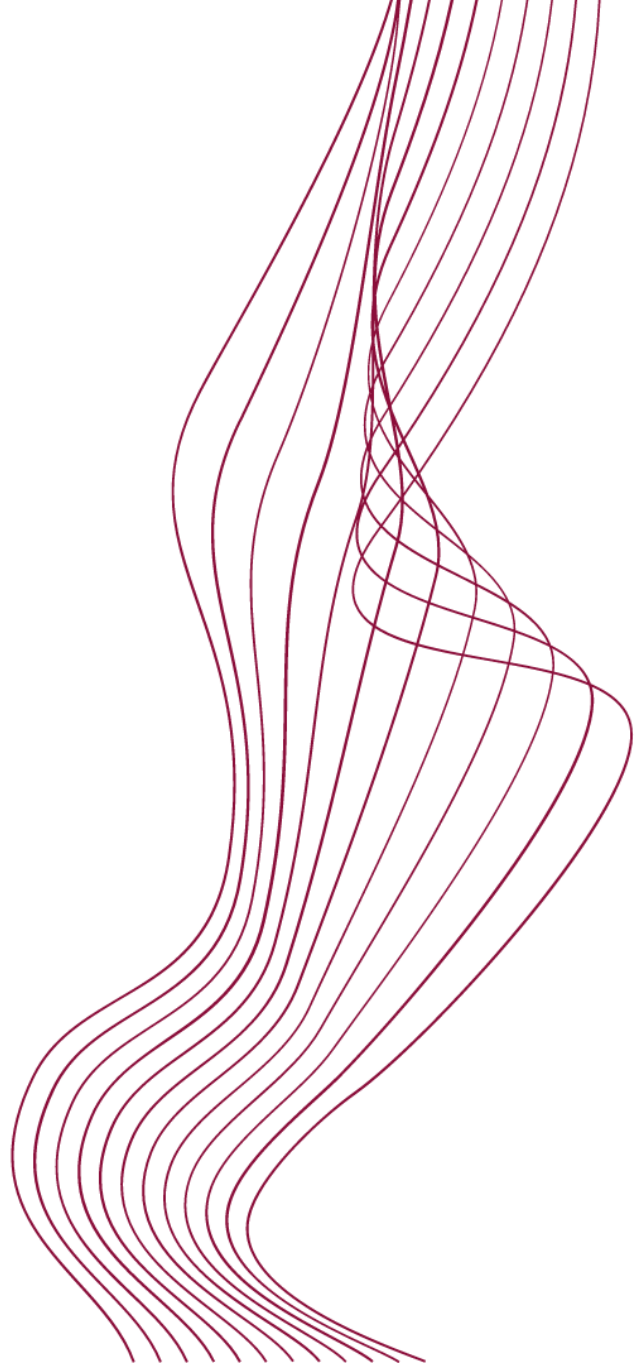
⚠️ Notice how I use the term `USING` and not `ON`. If your columns are named the same in both tables this is a much cleaner way to join.

```
SELECT
      transaction_date, gender, age,
      COUNT(*) volume,
      ROUND(SUM(amount)) value,
      COUNT(DISTINCT customer) distinct_customers
FROM transactions AS trans
LEFT JOIN customers as cust
USING(customer)
WHERE sku = 'p2p'
GROUP BY
      transaction_date,
      gender,
      age
LIMIT 10
;
```

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 3 - SQL for Policy)

Agenda

1) Homework
2) Creating Database
3) Property Market Analysis



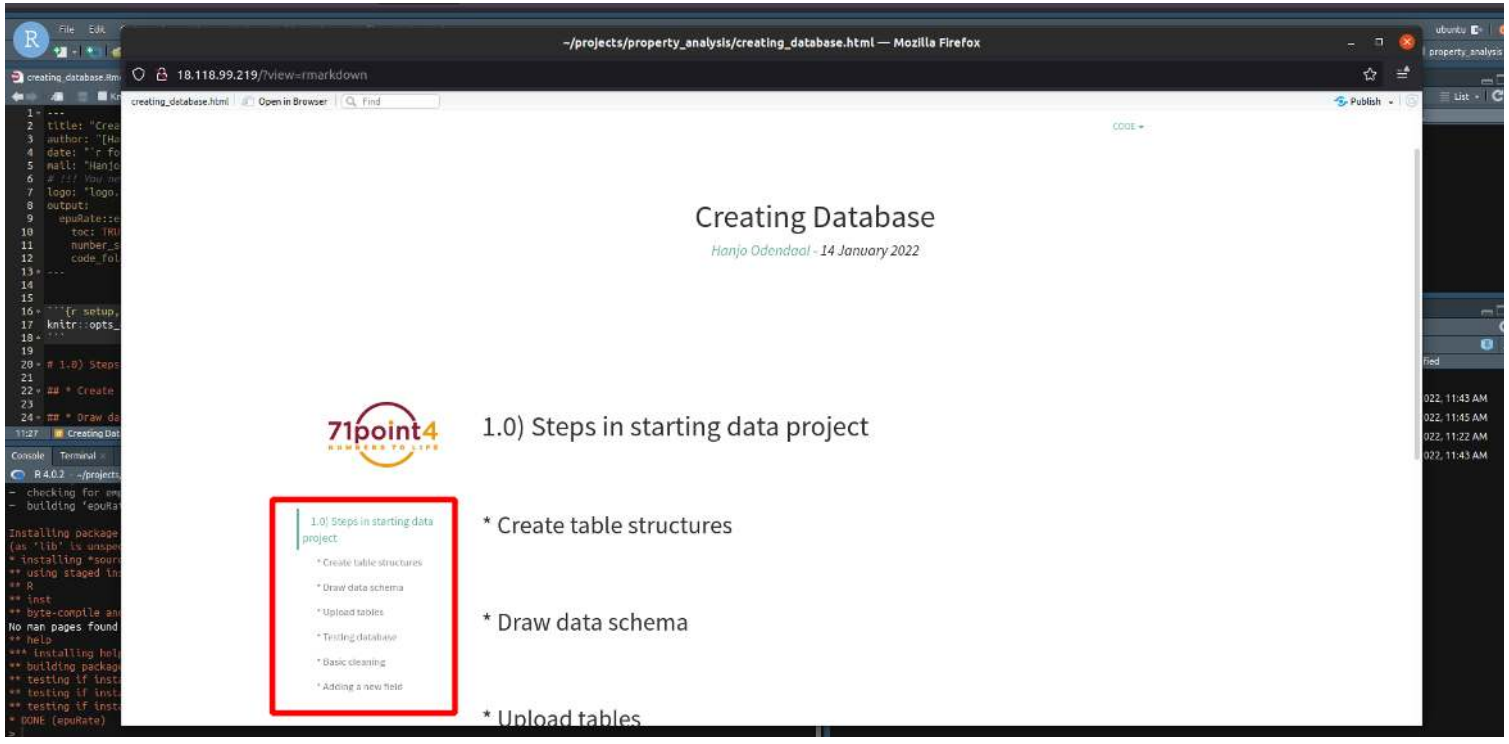
Homework 

Creating Database 💻

Starting your project

Whats our first step when starting any new analysis?

Correct! Start a new *project* in Rstudio and open a new `Rmarkdown` document with the name: *database_setup.rmd* in your new project folder called `projects/property_analysis`:



What are the steps?

Start a new *project* in Rstudio and open a new `Rmarkdown` document with the name: *database_setup.rmd* in your new project folder called `projects/property_analysis`.

Please create a `markdown` document with the following headings

```
# 1.0) Steps in starting data project

## * Create table structures

## * Draw data schema

## * Upload tables

## * Testing database

## * Basic cleaning
```

05:00

Create table structures

Yesterday we learned out to create two tables that are linked via `foreign` key. Today we will expand on this concept to create our `property` database. The database design I chose is a simple *snowflake schema*, which means that we will have a single *fact* table and multiple *dimension* tables.

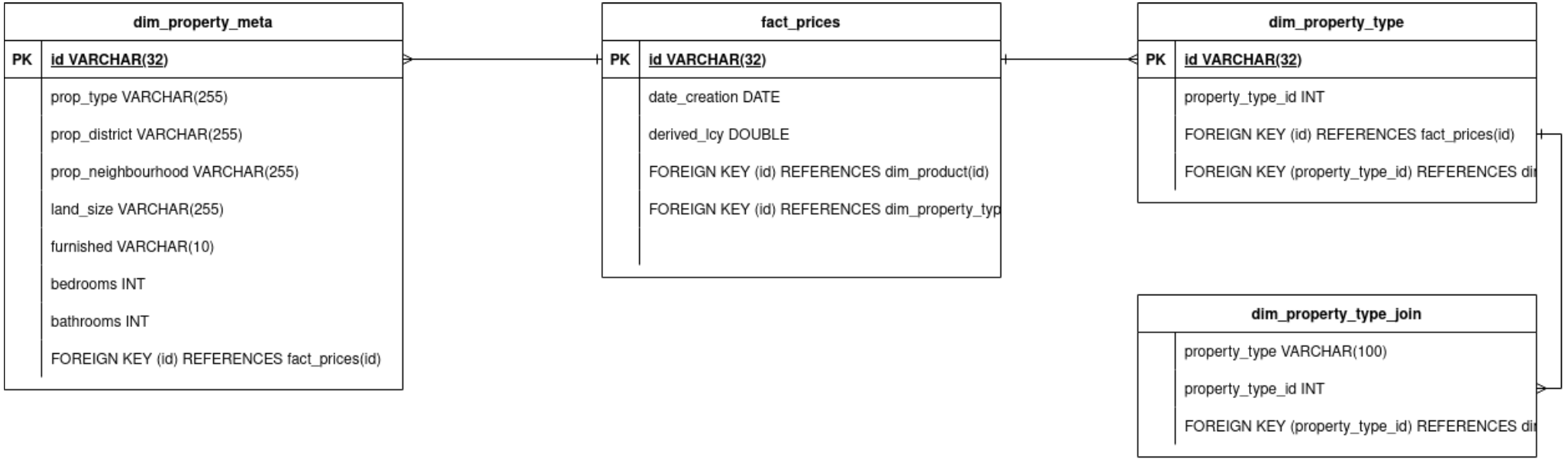
First things first, lets get an idee of the contents of the files. To do this I will use `less` from Day 1 of the workshop:

```
hanjo@optimus:~$ cd data/property
hanjo@optimus:~$ head fact_price.csv
```

```
hanjo@optimus:~/data/property$ head fact_price.csv
id,date_creation,derived_lcy
0cc5b20524c8b1a8f0ea7e7e8f4473fa,2021-03-09,448485.8045523818
fb395afb897e8c89f3a66c25b40ecaac,2021-03-09,915543.9302822812
0ed51896c7d589294369d165e36332bd,2021-03-09,606136.4057399756
6d2b335d459c6c5567829e4cfa975d0d,2021-03-09,1472279.0389409545
9f4eef5930e883f13925603dbc6707b7,2021-03-07,2077863.8082291621
8e30dc1965adf9bde649422c33e9be4d,2021-03-04,694528.3559667798
da86902a806a9461f33329b5f1bfc0c5,2021-03-04,1062383.21428563
3fb76d2ce9828ee329eeea849340dfc9,2021-03-01,659784.7312270133
09285d4fa1c5ea6fcce3c509e780c16c,2021-03-01,537751.7653602718
```

Create table structures

The first trick is to draw the relationships of the tables. Using `less` and `head` have a look at the files and *design* on a piece of paper how you would design this database. Keep in mind the tip I gave you about it being a *snowflake schema*.



15:00

Create table structures

If you need a refresher on the fields available to you, you can go to MariaDb website.

```
CREATE TABLE fact_prices(
  id  VARCHAR(32),
  date_creation DATE,
  derived_lcy DOUBLE,
  PRIMARY KEY(id, date_creation)
)
;
```

```
DROP TABLE IF EXISTS dim_property_type;
CREATE TABLE dim_property_type(
  id  VARCHAR(32),
  property_type_id INT,
  PRIMARY KEY (id),
  KEY(property_type_id),
  FOREIGN KEY (id) REFERENCES fact_prices(id)
)
;

DROP TABLE IF EXISTS dim_property_type_join;
CREATE TABLE dim_property_type_join(
  property_type  VARCHAR(100),
  property_type_id INT,
  CONSTRAINT foreign_key_property_type  FOREIGN KEY (pr
)
;
```

Create table structures

If you need a refresher on the fields available to you, you can go to MariaDb website.

```
CREATE TABLE dim_property_meta(
  id  VARCHAR(32),
  prop_type VARCHAR(255),
  prop_district VARCHAR(255),
  prop_neighbourhood VARCHAR(255),
  land_size VARCHAR(255),
  furnished VARCHAR(10),
  bedrooms INT,
  bathrooms INT,
  PRIMARY KEY(id),
  FOREIGN KEY (id) REFERENCES fact_prices(id)
)
;

SHOW INDEXES FROM dim_property_type_join\G
SHOW INDEXES FROM dim_property_meta\G
```

Draw data schema

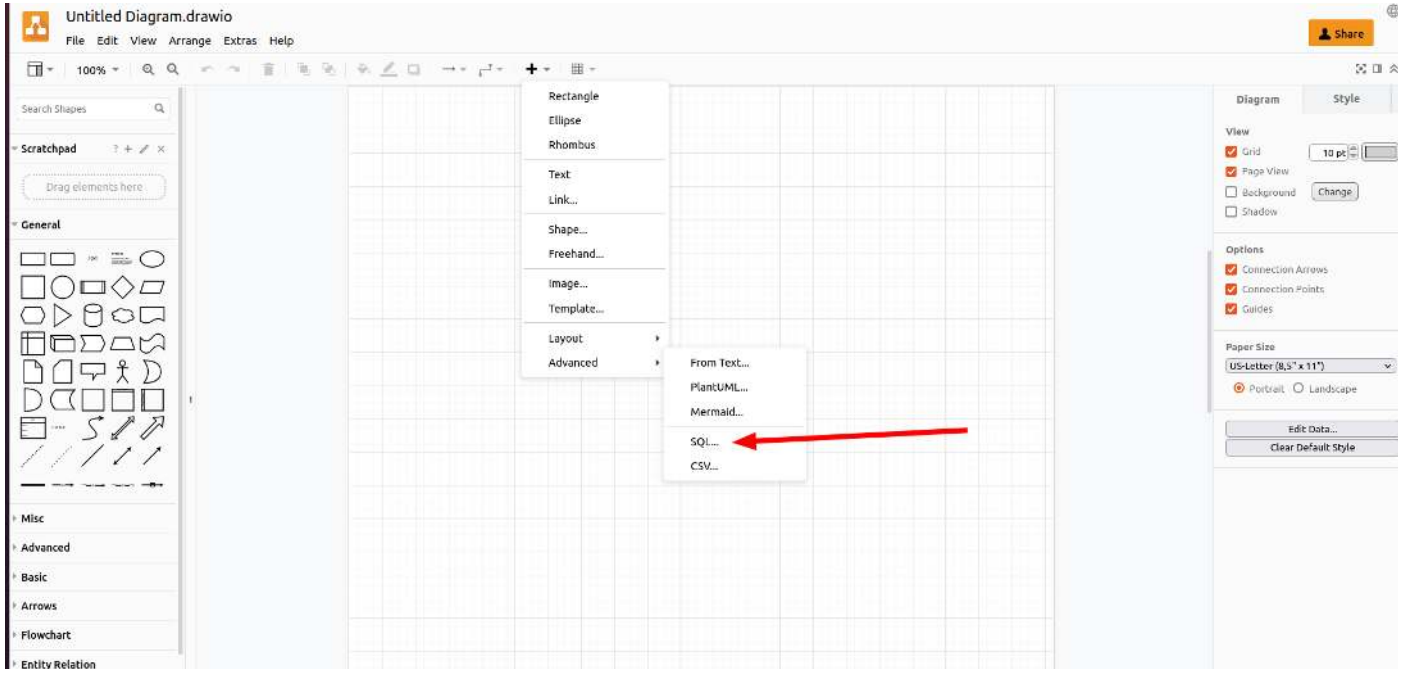
One of my favourite tools in the world is Draw.io.

I use it almost on a daily basis for all kinds of tasks.

- Drawing system architecture.
- Constructing database table relationships.
- Putting elements together for slides.

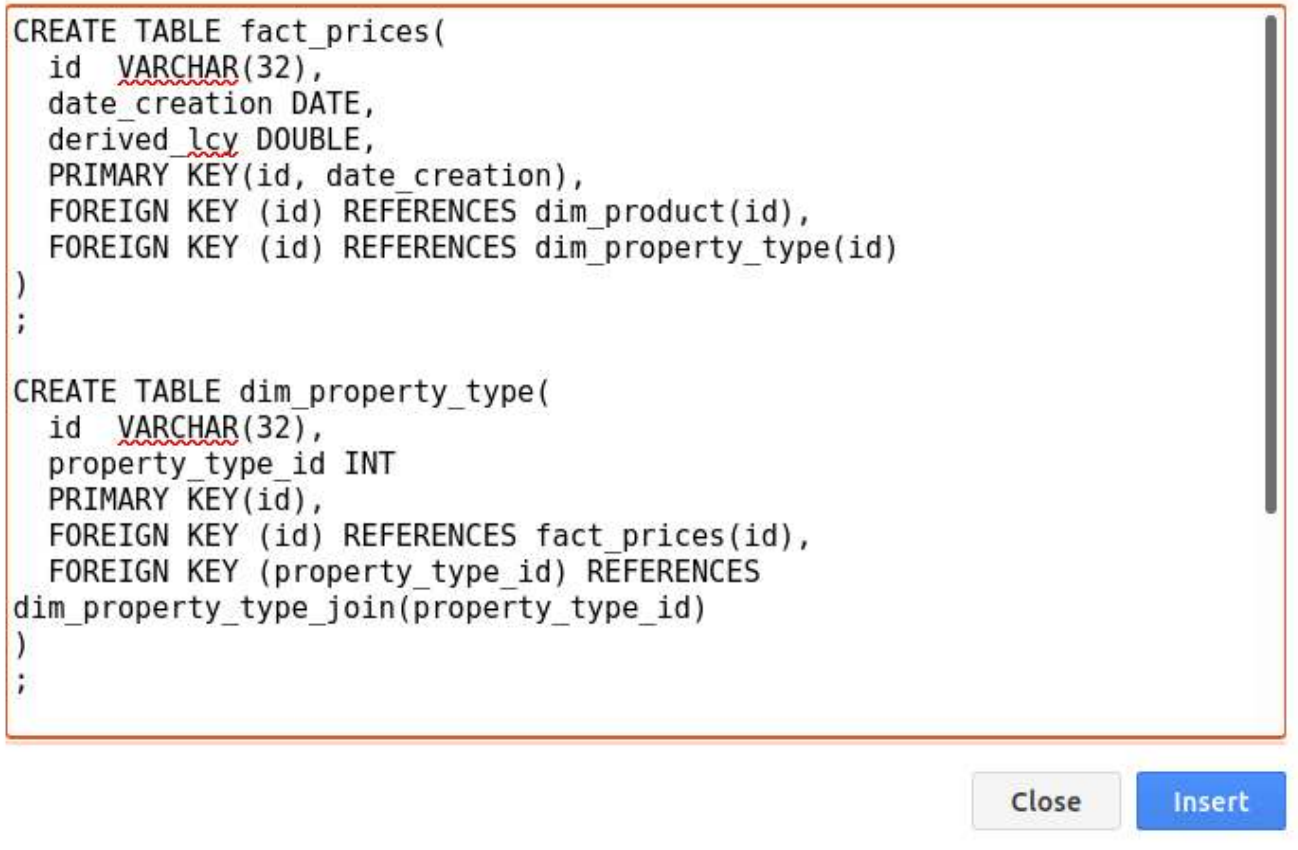
Draw data schema: Step 1

What is really nice is that once you have your create table statements, you can just paste the code into the console and it will create the tables for you! There is more expensive software that will even draw the relationships for you, but draw.io is free 💸



Draw data schema: Step 2

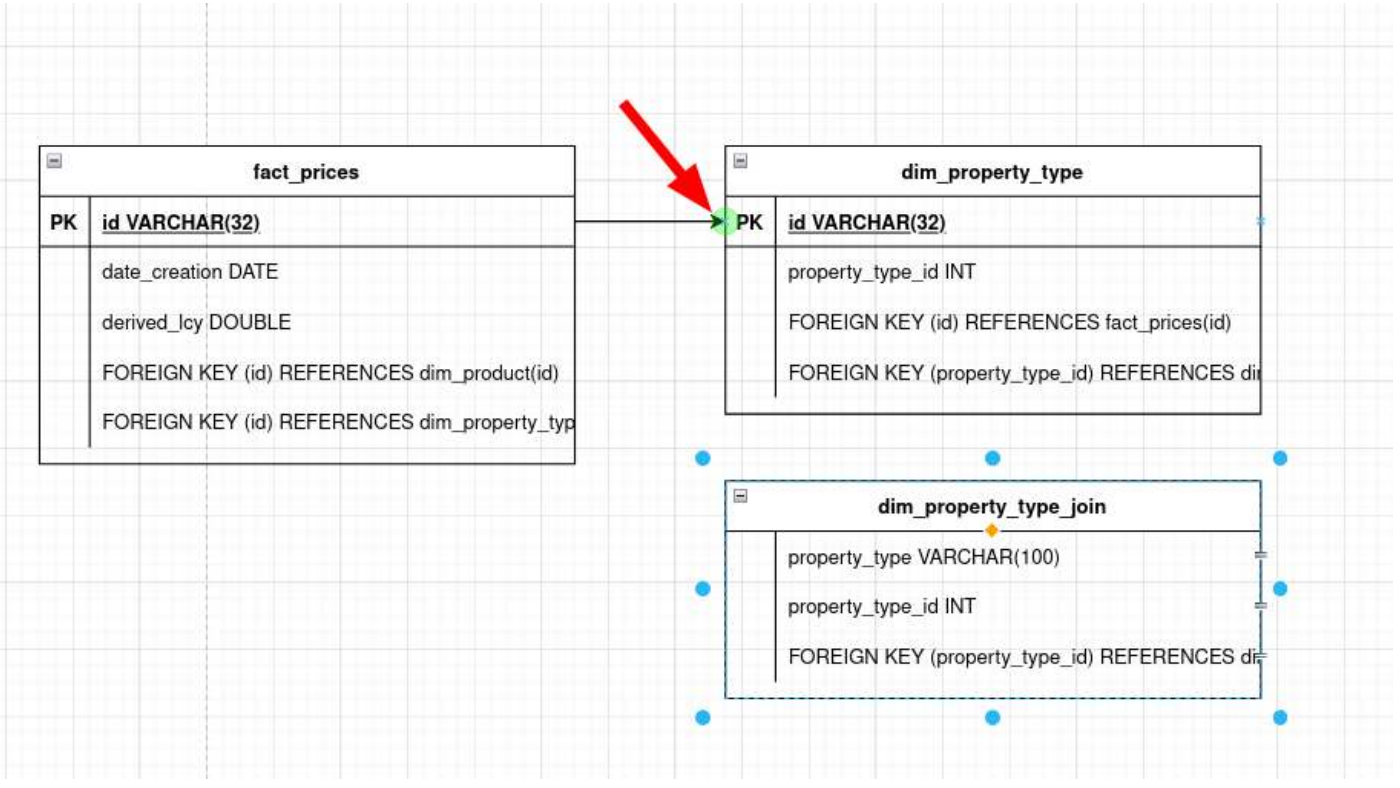
What is really nice is that once you have your create table statements, you can just paste the code into the console and it will create the tables for you! There is more expensive software that will even draw the relationships for you, but draw.io is free 🤑



```
CREATE TABLE fact_prices(
  id  VARCHAR(32),
  date_creation DATE,
  derived_lcy DOUBLE,
  PRIMARY KEY(id, date_creation),
  FOREIGN KEY (id) REFERENCES dim_product(id),
  FOREIGN KEY (id) REFERENCES dim_property_type(id)
)
;

CREATE TABLE dim_property_type(
  id  VARCHAR(32),
  property_type_id INT
  PRIMARY KEY(id),
  FOREIGN KEY (id) REFERENCES fact_prices(id),
  FOREIGN KEY (property_type_id) REFERENCES
dim_property_type_join(property_type_id)
)
;
```

Draw data schema: Step 3

What is really nice is that once you have your create table statements, you can just paste the code into the console and it will create the tables for you! There is more expensive software that will even draw the relationships for you, but draw.io is free 💸



Upload property tables

Once the proper database schema set up, its time to upload the tables!

Use the code template below to write the load script in your `markdown` file for each of the tables:

- `fact_price`
- `dim_property_meta`
- `dim_property_type`
- `dim_property_type_join`

```
LOAD DATA LOCAL INFILE '/home/ubuntu/data/property/{filename}.csv'
INTO TABLE property.{tablename}
FIELDS TERMINATED BY ','
IGNORE 1 LINES
;
```

05:00

Test your uploads

Its always good practice to test whether you tables uploaded correctly. I usually do three very basic tests:

- Query 10 rows of the data

```
SELECT * FROM {table_name} LIMIT 10;
```

- Count the number of rows in database

```
SELECT COUNT(*) FROM {table_name} LIMIT 10;
```

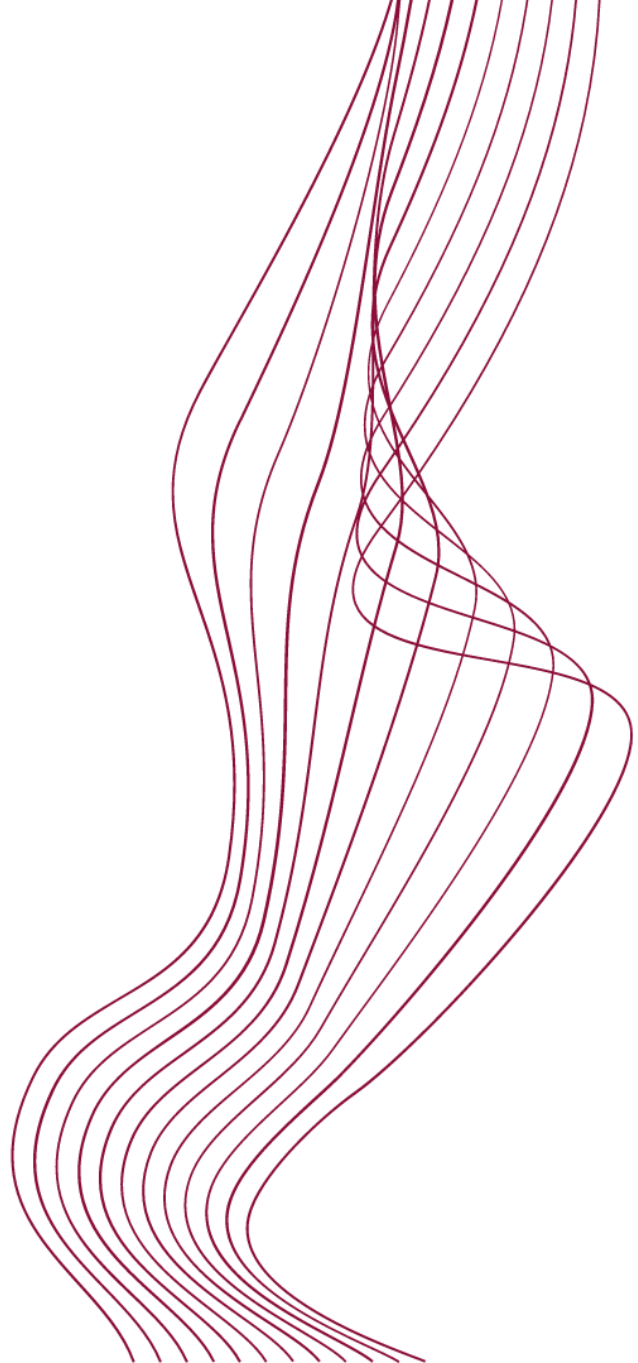
- Count number of rows in csv file

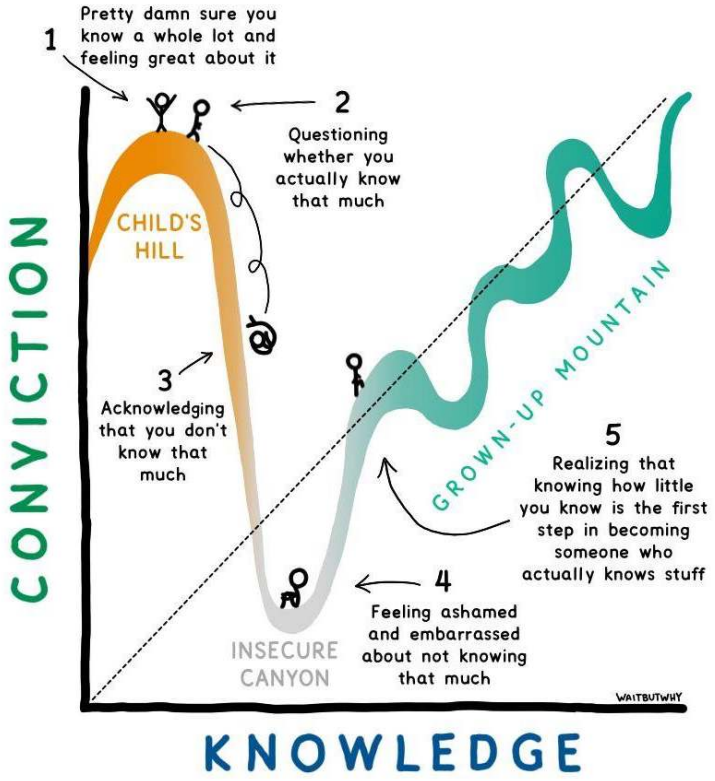
```
hanjo@optimus:~/data/property$ wc -l {filename}
```

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 3 - SQL for Policy)

Agenda

1) Homework
2) Creating Database
3) Property Market Analysis





> Things always seem impossible until they are done.
>
> — *Nelson Mandela*

https://www.reddit.com/r/PhD/comments/elnrtu/dunningkruger_effect_been_there_done_that/

Property Market Analysis 🧪

Starting your analysis

Whats our first step when starting any new analysis? Correct! Start a new *project* in Rstudio and open a new `Rmarkdown` document with the name: *analysis.rmd* in your new project folder called `projects/property_analysis`:

Creating a derived field

One of the things all analyst do is to add additional information into datasets by deriving new information using the data available to them. If you have done a bit of machine learning, you will be familiar with the term *feature engineering*.

What we are going to do is to identify columns which we think look like outliers. In this case, we are interested in flagging data if they are more than 2 standard deviations from the mean (a common outlier identifier technique).

```
SELECT
      AVG(derived_lcy),
      STDDEV(derived_lcy)/1000 as price_sd
FROM
      fact_prices
LIMIT 10
;
```

⚠️ Welcome to real world data! In the *advance* database course we will shift to analytical databases which can do a lot of statistical analysis on data, which `MariaDB` unfortunately does not have built in.

Creating a derived field

Creating a derived field

As we saw, we have two *real* problems with the data set:

- The data is dirty.
- MariaDB does not allow for `median` calculations would could help our cause.

Luckily we learned `Linux` 🤹! So lets use some google and see if we can get the *median* of a column using `bash`:

```
hanjo@optimus:~$ sudo apt install datamash
hanjo@optimus:~$ datamash -H --field-separator="," \
count 3 mean 3 \
median 3 perc:90 3 perc:95 3 \
pstdev 3 < \
fact_price.csv
```

This is a pretty cool functionality straight from the command-line! You learn something new every day...

Creating a derived field

Now that we have a much better idea of the data (Heavy right-tail distribution), lets create a column to identify outliers in the data set defined as:

- More than 1 million dollars (or 1 Billion RWF, `1e9`)

Welcome `CASE WHEN`:

```
SELECT
      *,
      CASE
          WHEN derived_lcy > 1e9 THEN 1
      ELSE 0
      END AS outlier
FROM
      fact_prices
LIMIT 10
;
```

Creating a derived field

It is always good to check how much of your observations might be in this subset of data. Do accomplish this we will use *subqueries*. Essentially we wrap our *original* query into a new query using `()`. Much easier than it sounds:

```
SELECT
      SUM(outlier)
FROM (
    SELECT
          *,
          CASE
              WHEN derived_lcy > 1e9 THEN 1
          ELSE 0
          END AS outlier
    FROM
          fact_prices
) as tbl_inner
LIMIT 10
;
```

Creating our analytical data set

At the moment, we would need to keep building sub-queries, joins etc if we want to do analysis on our *clean* data set.

What we could do is to create a new table in a single query were we incorporate all of the information into a single table. What are the steps?

- Join property meta.
- Join property type info onto property type and then onto prices.
- Filter out outliers.

⚠️ Although this seems like A LOT of information, remember what I said at the beginning of the course: *we build SQL inside-out*!

Using the principle of *inside-out* we approach the problem one piece at a time.

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LIMIT 10
;
```

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LEFT JOIN dim_property_type as proptype
USING(id)
LIMIT 10
;
```

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LEFT JOIN dim_property_type as proptype
USING(id)
LEFT JOIN dim_property_type_join as proptypejoin
USING(property_type_id)
LIMIT 10
;
```

Deciding on the final fields

Watch me/colleague go through the table to decide on the final columns for next few minutes.

Deciding on the final fields

```sql
CREATE TABLE property_global AS (
  SELECT
      id, date_creation, prop_type,
      prop_district, prop_neighbourhood,
      land_size, furnished,
      bedrooms, bathrooms,
      property_type, ROUND(derived_lcy) AS price
  FROM
  (
      SELECT * FROM fact_prices
      LEFT JOIN dim_property_meta as meta
      USING(id)
      LEFT JOIN dim_property_type as proptype
      USING(id)
      LEFT JOIN dim_property_type_join as proptype
      USING(property_type_id)
      WHERE
          derived_lcy < 1e9
  ) tbl_inner
)
;
```

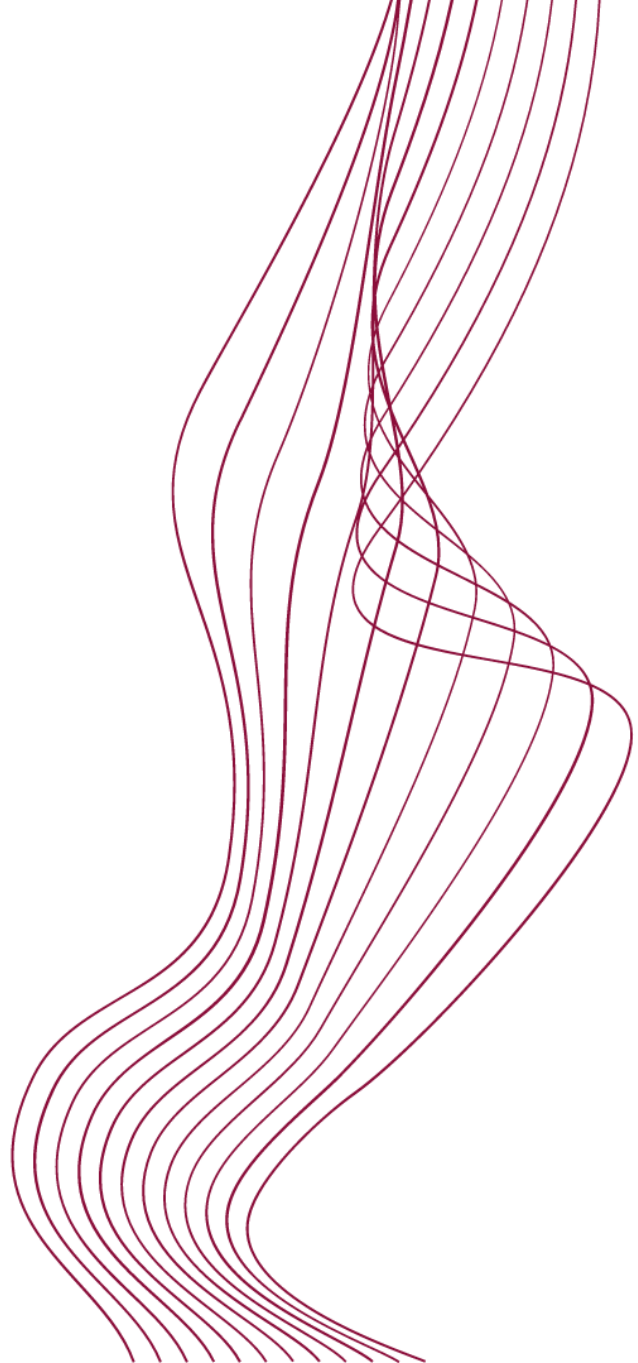
```sql
CREATE TABLE property_{type} AS (
  SELECT
      id, date_creation, prop_type,
      prop_district, prop_neighbourhood,
      land_size, furnished,
      bedrooms, bathrooms,
      property_type, ROUND(derived_lcy) AS price
  FROM
  (
      SELECT * FROM fact_prices
      LEFT JOIN dim_property_meta as meta
      USING(id)
      LEFT JOIN dim_property_type as proptype
      USING(id)
      LEFT JOIN dim_property_type_join as proptypejoin
      USING(property_type_id)
      WHERE
          derived_lcy < 1e9
          AND date_creation =/= '0000-00-00'
          AND property_type = {type}
  ) tbl_inner
)
;
```

Deciding on the final fields

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 3 - SQL for Policy)

Agenda

1) Homework
2) Creating Database
3) Property Market Analysis



Property Market Analysis 🧪

Starting your analysis

Still working on your rmarkdown file `projects/property_analysis`:

Property Analysis

Hanjo Odendaal - 15 January 2022

1.0) Analysing property data

* Creating a derived field.

* Which areas have the highest franc/sqm?

* Has priced increased over time?

* What premium is there for an extra bedroom in Kigali vs Outside of Kigali?

1.0) Analysing property data
* Creating a derived field.
* Which areas have the highest franc/sqm?
* Has priced increased over time?
* What premium is there for an extra bedroom in Kigali vs Outside of Kigali?
* Is it better to rent or buy in Kigali?

Creating our analytical data set

At the moment, we would need to keep building sub-queries, joins etc if we want to do analysis on our *clean* data set.

What we could do is to create a new table in a single query were we incorporate all of the information into a single table. What are the steps?

- Join property meta.
- Join property type info onto property type and then onto prices.
- Filter out outliers.

⚠️ Although this seems like A LOT of information, remember what I said at the beginning of the course: *we build SQL inside-out*!

Using the principle of *inside-out* we approach the problem one piece at a time.

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LIMIT 10
;
```

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LEFT JOIN dim_property_type as proptype
USING(id)
LIMIT 10
;
```

Creating our analytical data set

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LEFT JOIN dim_property_type as proptype
USING(id)
LEFT JOIN dim_property_type_join as proptypejoin
USING(property_type_id)
LIMIT 10
;
```

Creating our analytical data set

Once we have a good understanding of the data, we can easily create a table using the following:

```
SELECT * FROM fact_prices
LEFT JOIN dim_property_meta as meta
USING(id)
LEFT JOIN dim_property_type as proptype
USING(id)
LEFT JOIN dim_property_type_join as proptypejoin
USING(property_type_id)
LIMIT 10
;
```

```
CREATE TABLE property_global AS (
% SOME SQL STATEMENT
)
```

Deciding on the final fields

Watch me/colleague go through your ideas to decide on the final columns for next few minutes.

Deciding on the final fields

```sql
CREATE TABLE property_global AS (
  SELECT
      id, date_creation, prop_type,
      prop_district, prop_neighbourhood,
      land_size, furnished,
      bedrooms, bathrooms,
      property_type, ROUND(derived_lcy) AS price
  FROM
  (
      SELECT * FROM fact_prices
      LEFT JOIN dim_property_meta as meta
      USING(id)
      LEFT JOIN dim_property_type as proptype
      USING(id)
      LEFT JOIN dim_property_type_join as proptype
      USING(property_type_id)
      WHERE
          derived_lcy < 1e9
  ) tbl_inner
)
;
```

```sql
CREATE TABLE property_{type} AS (
  SELECT
      id, date_creation, prop_type,
      prop_district, prop_neighbourhood,
      land_size, furnished,
      bedrooms, bathrooms,
      property_type, ROUND(derived_lcy) AS price
  FROM
  (
      SELECT * FROM fact_prices
      LEFT JOIN dim_property_meta as meta
      USING(id)
      LEFT JOIN dim_property_type as proptype
      USING(id)
      LEFT JOIN dim_property_type_join as proptypejoin
      USING(property_type_id)
      WHERE
          derived_lcy < 1e9
          AND date_creation =/= '0000-00-00'
          AND property_type = {type}
  ) tbl_inner
)
;
```

Answering Policy Questions 

Recap on our questions

Lets recap on what we want to know about the property market:

- Which areas have the highest franc/sqm?
    - This requires us to *only* look at land (`WHERE`) and then derive a new column where we take price and divide by land size.
    - We then also need to order (`DESC`) on the derived column
- Has priced increased over time?
    - We need to calculate the `AVG` price per time period.
    - Perhaps year is a good idea? So we will need to round the date column to first date of the year.
- What premium is there for an extra bedroom in Nyarugenge vs Outside of Nyarugenge?
    -
        - How much will it cost me to live inside Nyarugenge for 4 and 5 bedroom *houses* vs outside of Nyarugenge?
    - Here we will need a `CASE WHEN` to create a new column that calculates `AVG` price when location is Nyarugenge and `AVG` when not.
    - The do this by `GROUP BY` bedroom.

Which areas have the highest franc/sqm?

- This requires us to *only* look at land (`WHERE`) and then derive a new column where we take price and divide by land size.
- We then also need to order (`DESC`) on the derived column
- Then calculate the `AVG` per district using a *SUB QUERY*

prop_neighbourhood	avg_sqm_price
Kiyovu	234024
Gacuriro	91356
Rebero	73316
Kabeza	71345
Kibagabaga	66425
Remera	65870
Gikondo	51860
Gisozi	50504
Niboye	48829
Kimironko	43856

20:00

Which areas have the highest franc/sqm?

- This requires us to *only* look at land (`WHERE`) and then derive a new column where we take price and divide by land size.
- We then also need to order (`DESC`) on the derived column
- Then calculate the `AVG` per district using a *SUB QUERY*

```
SELECT
	*
FROM
	property_sale
WHERE
	prop_type = 'Land'
	AND land_size =/= ''
	AND price IS NOT NULL
;
```

Which areas have the highest franc/sqm?

- This requires us to *only* look at land (`WHERE`) and then derive a new column where we take price and divide by land size.
- We then also need to order (`DESC`) on the derived column
- Then calculate the `AVG` per district using a *SUB QUERY*

```
SELECT
      prop_district,
      price,
      land_size,
      ROUND(price/land_size) AS sqm_price
FROM
      property_sale
WHERE
      prop_type = 'Land'
      AND land_size =/= ''
      AND price IS NOT NULL
;
```

Which areas have the highest franc/sqm?

- This requires us to *only* look at land (`WHERE`) and then derive a new column where we take price and divide by land size.
- We then also need to order (`DESC`) on the derived column
- Then calculate the `AVG` per district using a *SUB QUERY*

```
SELECT
      prop_neighbourhood,
      ROUND(AVG(sqm_price)) AS avg_sqm_price
FROM(
  SELECT
        prop_neighbourhood,
        price,
        land_size,
        price/land_size AS sqm_price
  FROM
        property_sale
  WHERE
        prop_type = 'Land'
        AND land_size =/= ''
        AND price IS NOT NULL
) tbl_inner
GROUP BY
      prop_neighbourhood
ORDER BY
      avg_sqm_price DESC
LIMIT 10
;
```

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.

date_year	avg_price	nr_prices
2021-01-01	71776446	22
2020-01-01	77370796	136
2019-01-01	64184032	107

20:00

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.

```
SELECT
      LEFT(date_creation, 4)
FROM
      property_sale
LIMIT 10
;
```

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.

```
SELECT
      date_creation,
      CONCAT(LEFT(date_creation, 4),'-01-01') as date_y
FROM
      property_sale
LIMIT 10
;
```

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.

```
SELECT
      date_creation,
      CONCAT(LEFT(date_creation, 4),'-01-01') as date_y
      price
FROM
      property_sale
WHERE
      prop_type = 'House'
LIMIT 10
;
```

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.

```
SELECT
      date_year,
      ROUND(AVG(price)) as avg_price,
      COUNT(*) nr_prices
FROM(
      SELECT
            date_creation,
            CONCAT(LEFT(date_creation, 4),'-01-01') as
            price
      FROM
            property_sale
      WHERE
            prop_type = 'House'
) tbl_inner
GROUP BY
      date_year
ORDER BY
      date_year DESC
;
```

Has prices increased over time?

- This is an example of some advanced *window* functions. We will not cover these in this course, but in the *advanced sql* course we will go in-depth on these *analytical* functions.

```
SELECT
      *,
      CONCAT(FORMAT((
            avg_price/
              (LEAD(avg_price) OVER (ORDER BY avg_price)) - 1
            ) * 100,
          2),
      '%') perc_change
FROM(
  SELECT
        date_year,
        ROUND(AVG(price)) as avg_price,
        COUNT(*) nr_prices
  FROM(
        SELECT
              date_creation,
              CONCAT(LEFT(date_creation, 4),'-01-01') as date_year,
              price
        FROM
              property_sale
        WHERE
              prop_type = 'House'
  ) tbl_inner
  GROUP BY
        date_year
  ORDER BY
        date_year DESC
) tbl_outer
;
```

Premium for extra bedroom in Nyarugenge?

- How much will it cost me to live inside Nyarugenge for 3, 4 and 5 bedroom *houses* vs outside of Nyarugenge?
- Here we will need a `CASE WHEN` to create a new column that calculates `AVG` price when location is Kigali and `AVG` when not.
- Then calculate the metrics by `GROUP BY` bedroom.

```
CASE
  WHEN {SOMETHING} THEN {THIS}
ELSE {THIS OTHER THING}
END {NAME}
```

bedrooms	nr_houses	nyarugenge	other	price_diff
3	44	32654964	38898614	-6243650
4	168	92804800	61526121	31278679
5	35	48931040	141911931	-92980891

30:00

Premium for extra bedroom in Nyarugenge?

- How much will it cost me to live inside Nyarugenge for 3, 4 and 5 bedroom *houses* vs outside of Nyarugenge?
- Here we will need a `CASE WHEN` to create a new column that calculates `AVG` price when location is Kigali and `AVG` when not.
- Then calculate the metrics by `GROUP BY` bedroom.

```
CASE
  WHEN {SOMETHING} THEN {THIS}
ELSE {THIS OTHER THING}
END {NAME}
```

```
SELECT
      *
FROM
      property_sale
WHERE
      prop_type = 'House'
      AND bedrooms BETWEEN 3 AND 5
;
```

Premium for extra bedroom in Nyarugenge?

- How much will it cost me to live inside Nyarugenge for 3, 4 and 5 bedroom *houses* vs outside of Nyarugenge?
- Here we will need a `CASE WHEN` to create a new column that calculates `AVG` price when location is Nyarugenge and `AVG` when not.
- Then calculate the metrics by `GROUP BY` bedroom.

```
CASE
  WHEN {SOMETHING} THEN {THIS}
ELSE {THIS OTHER THING}
END {NAME}
```

```sql
SELECT
      bedrooms,
      CASE
          WHEN prop_district = 'Nyarugenge' THEN price
      ELSE NULL END nyarugenge,
      CASE
          WHEN prop_district = 'Nyarugenge' THEN NULL
      ELSE price END other
FROM
      property_sale
WHERE
      prop_type = 'House'
      AND bedrooms BETWEEN 3 AND 5
;
```

Premium for extra bedroom in Nyarugenge?

- How much will it cost me to live inside Nyarugenge for 3, 4 and 5 bedroom *houses* vs outside of Nyarugenge?
- Here we will need a `CASE WHEN` to create a new column that calculates `AVG` price when location is Nyarugenge and `AVG` when not.
- Then calculate the metrics by `GROUP BY` bedroom.

```
CASE
  WHEN {SOMETHING} THEN {THIS}
ELSE {THIS OTHER THING}
END {NAME}
```

```
SELECT
      bedrooms,
      COUNT(*) nr_houses,
      ROUND(AVG(nyarugenge)) AS nyarugenge,
      ROUND(AVG(other)) AS other,
      ROUND(AVG(nyarugenge)) - ROUND(AVG(other)) as price_diff
FROM
(
  SELECT
        bedrooms,
        CASE
            WHEN prop_district = 'Nyarugenge' THEN price
        ELSE NULL END nyarugenge,
        CASE
            WHEN prop_district = 'Nyarugenge' THEN NULL
        ELSE price END other
  FROM
        property_sale
  WHERE
        prop_type = 'House'
        AND bedrooms BETWEEN 3 AND 5
) tbl_inner
GROUP BY
    bedrooms
;
```

Premium for extra bedroom in Nyarugenge?

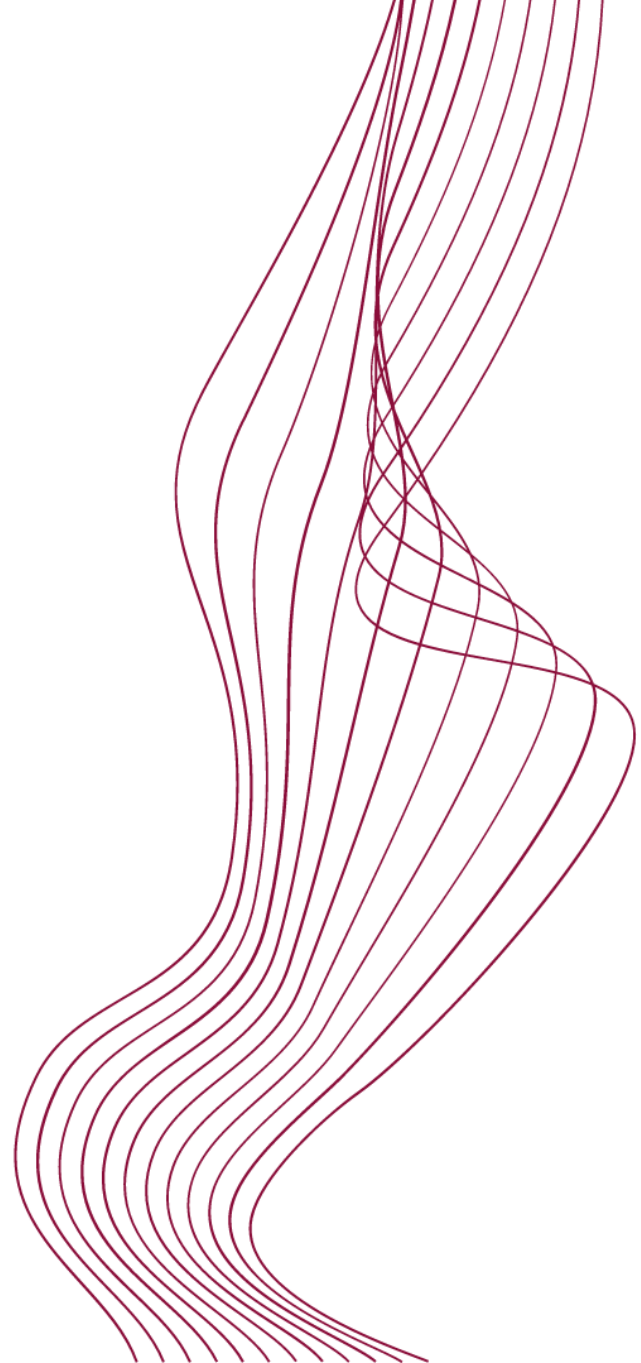
- Some extra functions and features

```
SELECT
      bedrooms,
      COUNT(*) nr_houses,
      FORMAT(ROUND(AVG(nyarugenge)), 2) AS nyarugenge,
      FORMAT(ROUND(AVG(other)), 2) AS other,
      FORMAT(ROUND(AVG(nyarugenge)) - ROUND(AVG(other)), 2) AS price_diff
FROM
(
  SELECT
        bedrooms,
        CASE
            WHEN prop_district = 'Nyarugenge' THEN price
        ELSE NULL END nyarugenge,
        CASE
            WHEN prop_district = 'Nyarugenge' THEN NULL
        ELSE price END other
  FROM
        property_sale
  WHERE
        prop_type = 'House'
        AND bedrooms BETWEEN 3 AND 5
) tbl_inner
GROUP BY
    bedrooms
;
```

Foundation of Data-Driven Analysis for Policy Decisions Course (Day 4 - RStudio)

Agenda

1) Homework
2) Building a data dictionary
3) Tidyverse
4) R Analysis



Homework 

Tidyverse and Rstudio 

What is the tidyverse?

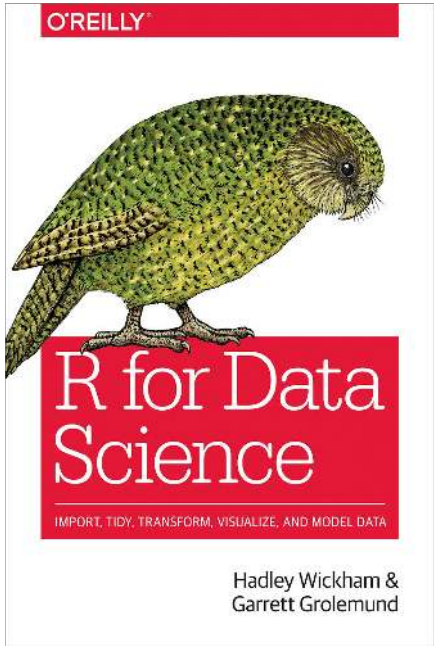
Art by Allison Horst

What is the tidyverse?

> The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

This collection contains some of the most used libaries that an `R` data scientist will use on a daily basis. The most used packages are probably `dplyr` and `ggplot`. Today we gonna explore the *basics* of these two amazing packages.

- `dplyr` is the grammar of data manipulation (`select`, `filter`, `group_by`, `mutate`)
- `ggplot` is the grammar of graphics



What is the tidyverse?

Although we only going to be learning the basics of the tidyverse universe, there is A LOT more to explore in terms of the power of programmin languages like `R` (and `Python`).

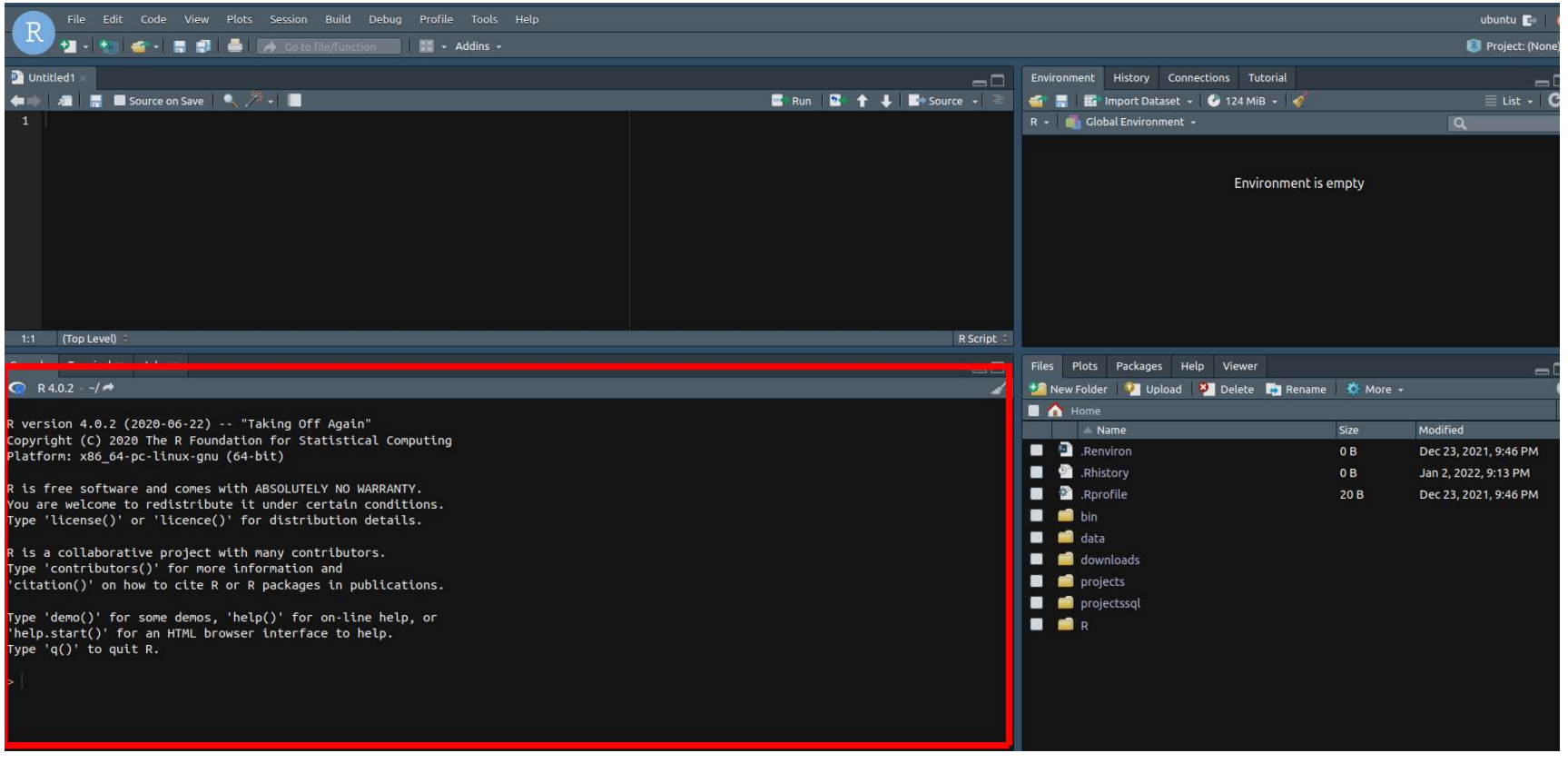
We recommend R for data analyses due to its firm pedigree in statistical analysis. `Python` is getting better at manipulating data with packages like `pandas` and alike, while `R` has become a more general language over the last few years.

Even though `python` does offer some nice integration features, `R` offers a much better ecosystem that supports reproducible research and data analysis (`Rmarkdown`, `blogdown`, `targets` etc.).

Also, once you grasp the fundamentals of programming, it is very easy to learn another language if it is better suited towards what you want to achieve.

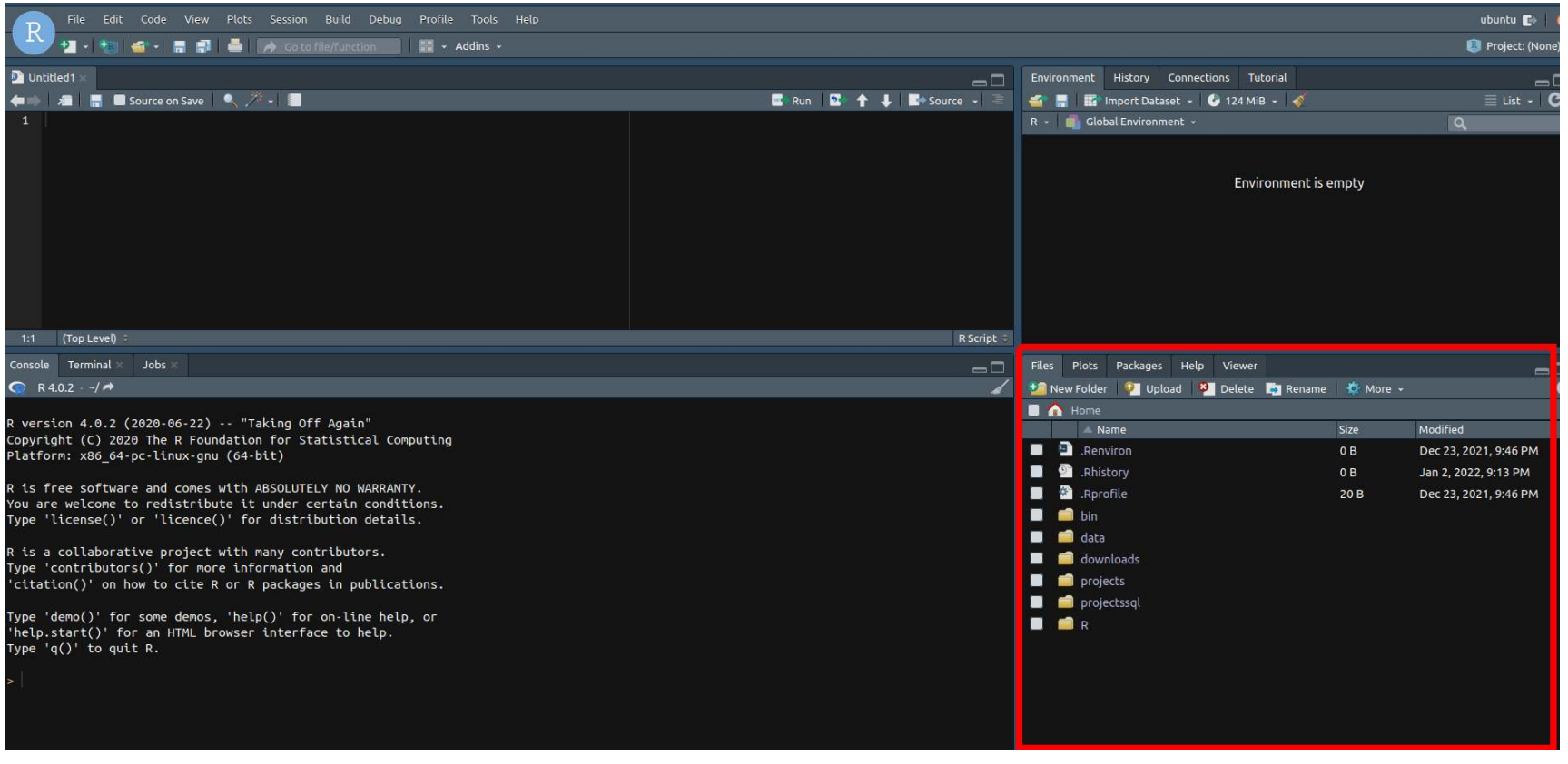
Rstudio recap

The console give you a place to execute commands written in R.



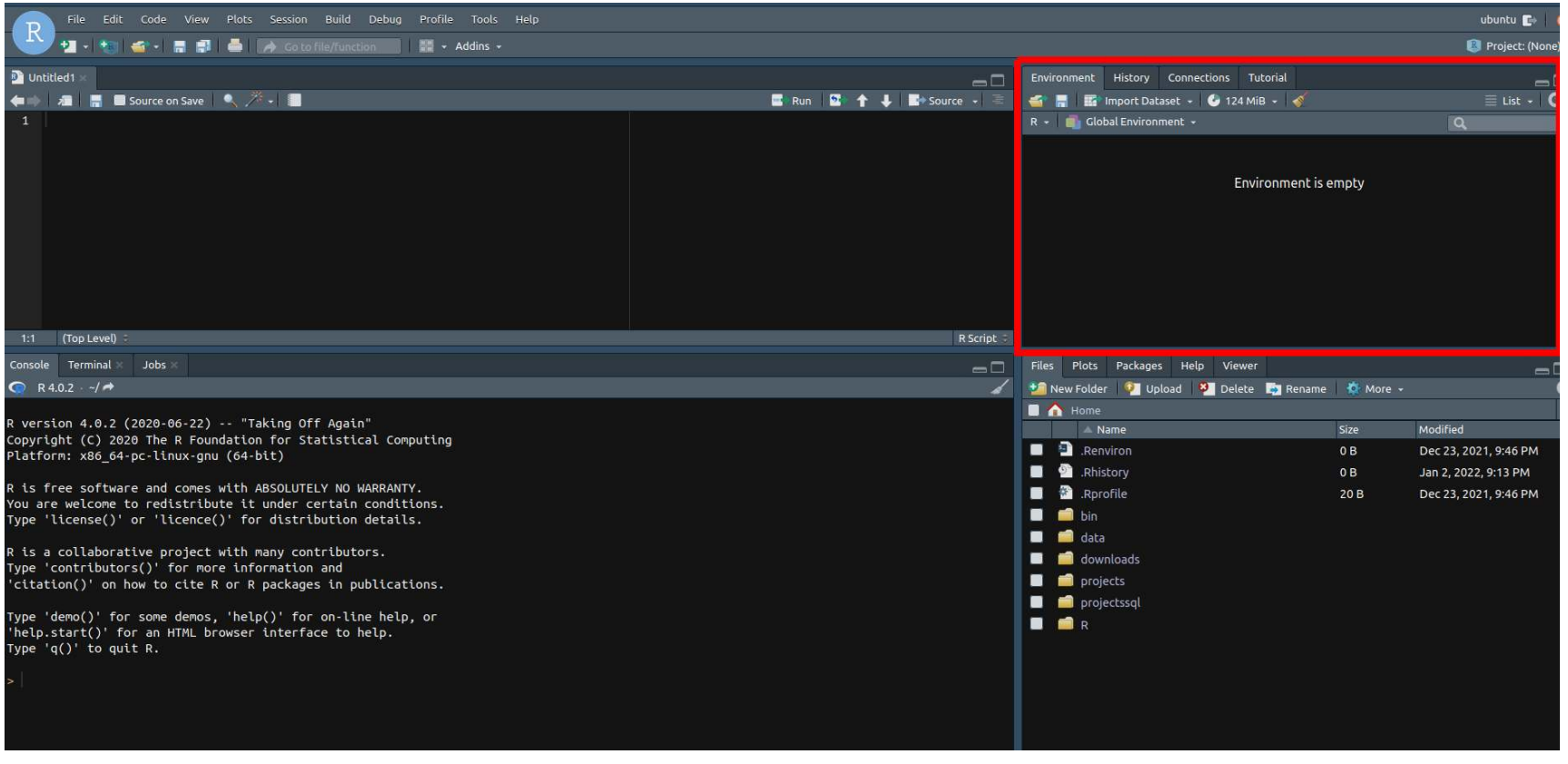
Rstudio recap

Rstudio also provides a *file explorer* which allows you user to navigate the folders easily.



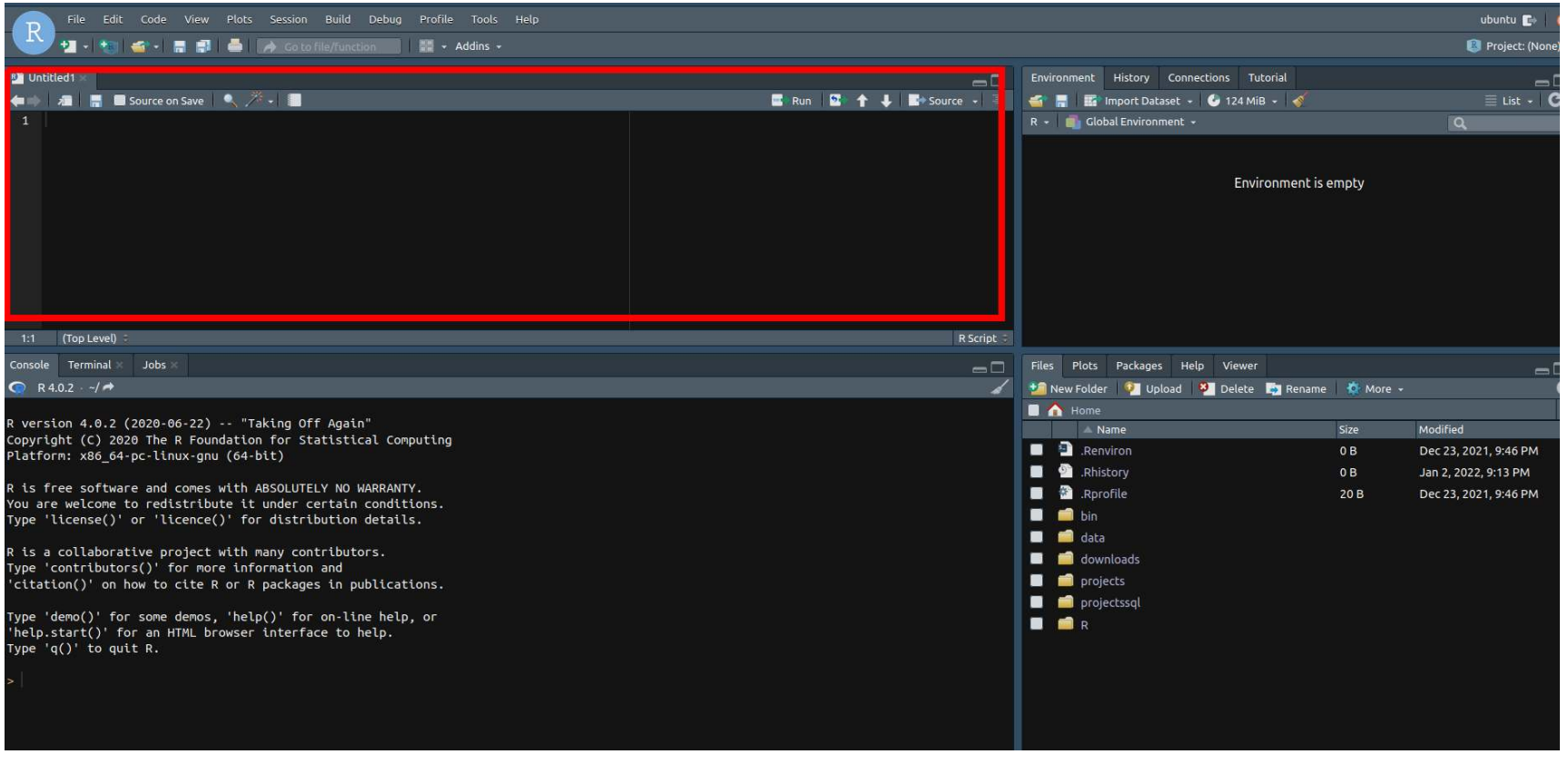
Rstudio recap

Once we start *assiging* outputs to objects, they will appear in the environment window.



Rstudio recap

Lastly, and most importantly, we want to write scripts that we can rerun at a later time.



Using Projects

Remember I have been pushing you to use projects 🤔? Now they will become really important in your analysis.

We want to avoid feeling like that by keeping all our *notes*, *scripts*, *data* and *output* in one single place. This is where Rstudio makes it easy by creating a project.

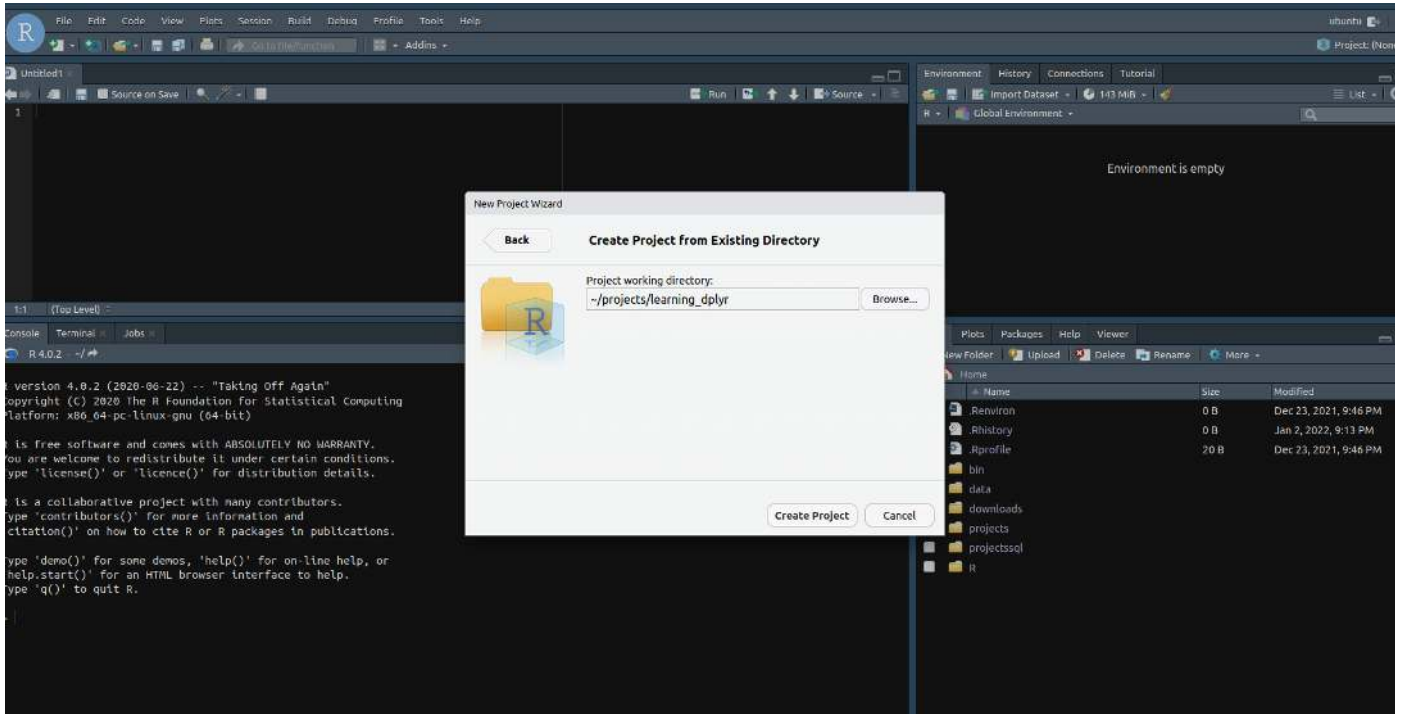
Start by creating a folder in your `home` directory called `projects` and starting a project called `learning_dplyr`:

```
hanjo@optimus:~$ mkdir -p projects/learning_dplyr
```

- Next click on the menu:
- `File > New Project > Existing Directory`

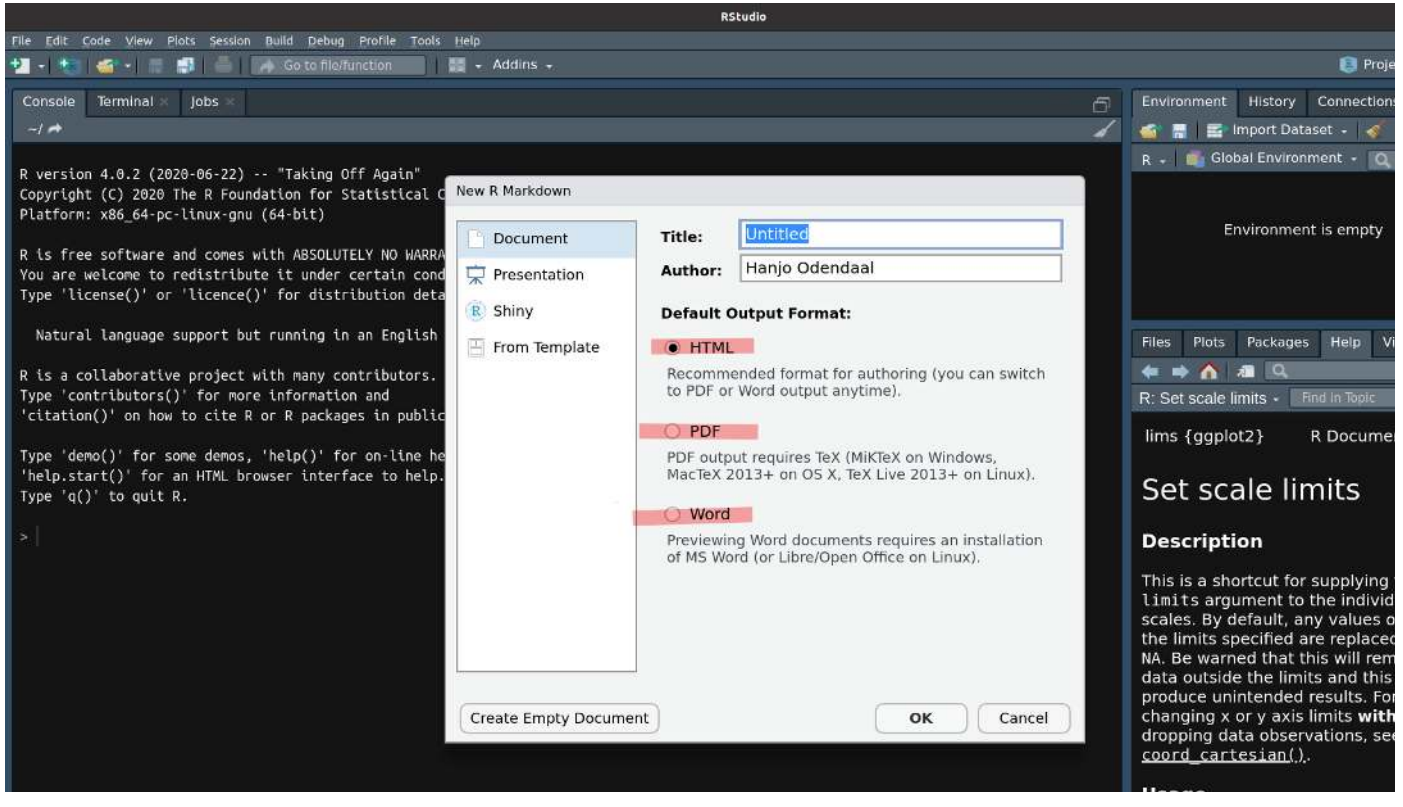
Using Projects

Using the file browser, you should see the following screen below:



Using Rmarkdown

- Start by opening a new *Rmarkdown* file (`.rmd`) in your `learning_dplyr project` and call it `dplyr_lesson.Rmd`:



Using Rmarkdown for analysis

Lets make it look nice by changing the `yaml` at the top:

```
---
title: "Your title here"
date: "Todays date"
output:
  html_document:
    theme: journal
    highlight: espresso
    toc: true
    toc_depth: 4
    toc_float: true
    code_folding: show
---
```

Now, very important, lets set the default settings for the document:

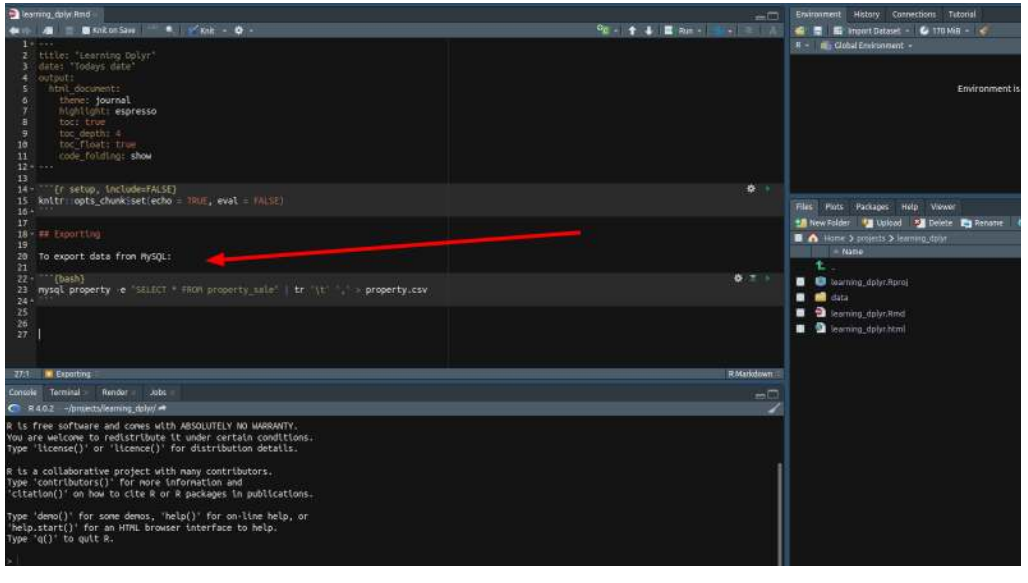
```
knitr::opts_chunk$set(echo = TRUE, eval = FALSE)
```

Using Rmarkdown for analysis

In order to use the data that we have in `SQL` in `R` we need to export the table. We can accomplish this with a very simple function:

```
hanjo@optimus:~$ cd projects/learning_dplyr
hanjo@optimus:~/projects/learning_dplyr$ mkdir data
hanjo@optimus:~/projects/learning_dplyr/data$ mysql property -e "SELECT * FROM property_sale" | \
tr '\t' ',' > property.csv
```

This function will output the data from `MySQL` and replace all `\t` (tabs is default from `MySQL`) with commas. Remember to add this to your markdown for later use!



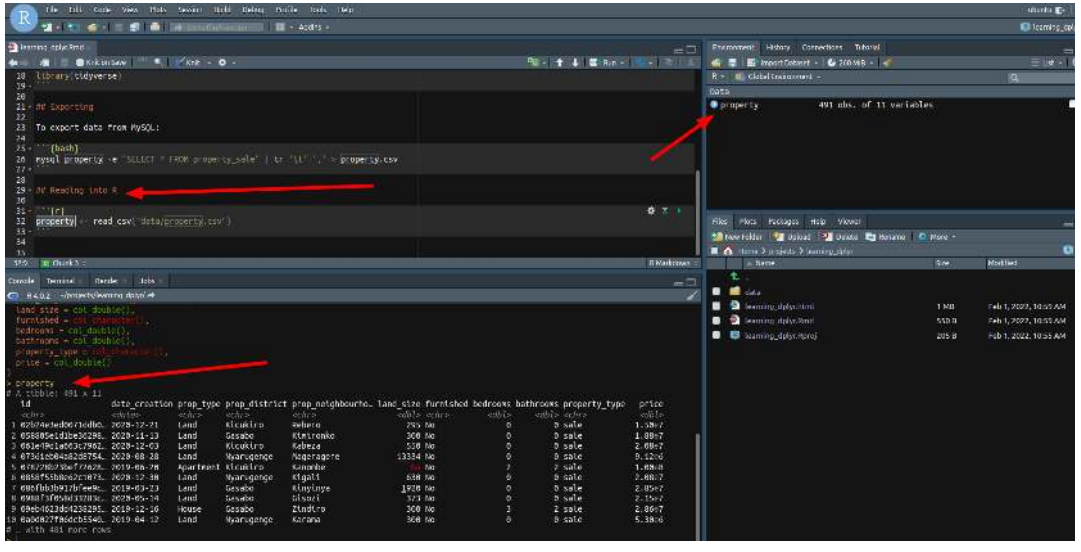
Reading data into R

To analyse the data, we need to read it into `R`. What do you think the function will be to do that?

Correct, its `read_csv`!

```
property ← read_csv("data/property.csv")
```

- `read_csv`: Function that reads file
- `"~/data/property/fact_price.csv"`: Path to file (note in quotes which means character)
- `←`: Assignment operator, so *assign* output to `property`
- `property`: Name of object that we asign output to



Manipulating object with dplyr

Art by Allison Horst

Manipulating object with dplyr

Now that the object is in `R` environment, we can use the `dplyr` library to manipulate the data using very basic functions:

- `select`: Selects specific columns by name.
- `filter`: Filter data based on certain criteria.
- `mutate`: Create a new column.
- `group_by`: Column to aggregate on.
- `summarise`: How do you want to summarise the data?

In `R` we gonna *chain* these commands using whats called the `pipe` operator: `%>%`. The shortcut to print this symbol is: `Ctrl + Shift + m`.

> We read this `%>%` symbol as: *and then*

- So for instance `property %>% select(price)` reads in english as: Take object `property` *and then* select column price.
- Another would be `property %>% filter(price < 10e6)` : Take object `property` *and then* filter where price more than 10 million.

select()

Extract columns by name: `select(.data, ...)`

```
property %>% select(id, prop_type, prop_district)
```

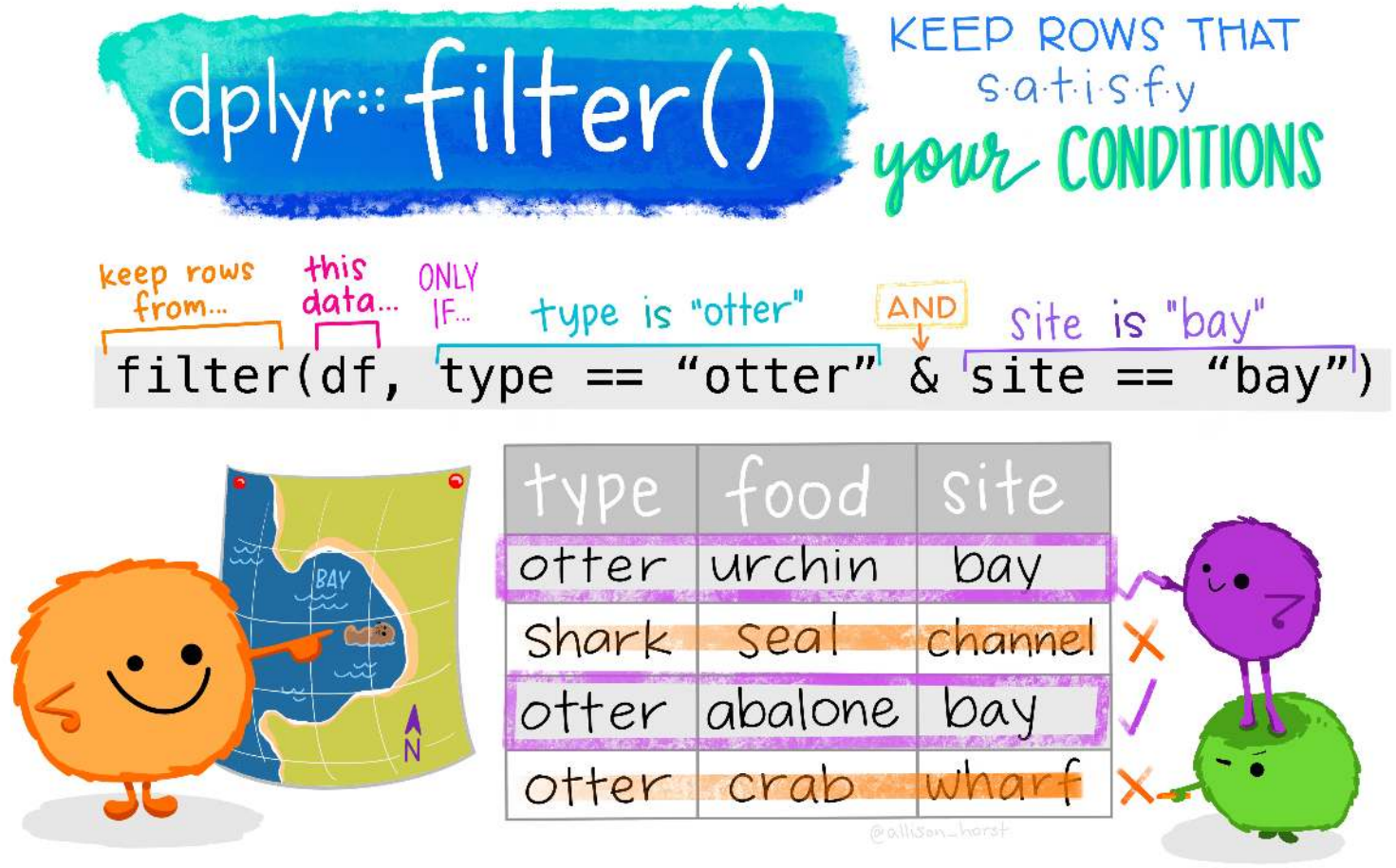
We can also use smart selectors:

```
property %>% select(id, contains("prop"))
```

Or even ranges:

```
property %>% select(id, bedrooms:price)
```

filter()



Art by Allison Horst

filter()

What if we want to only analyze certain rows? In `SQL` we used `WHERE`, while in `dplyr` we gonna use `filter`:

```
property %>% filter(price < 10e6)
```

We can use multiple conditions to filter (this represents an `AND`):

```
property %>% filter(bedrooms > 3, bathrooms > 2, price < 100e6)
```

The special function `%in%` also gets used often to specify multiple conditions:

```
property %>% filter(bedrooms %in% c(3, 4, 5))
```

Its also possible to create `OR` filters using the `pipe` delimiter ("|"):

```
property %>% filter(prop_district == 'Gasabo' | prop_district == 'Kicukiro')
```

mutate()

Art by Allison Horst

mutate()

Often you will need to add a new column that you derive. To accomplish this using `dplyr` we use `mutate`. Lets calculate the price per sqm as we did in `SQL`:

```
property %>% mutate(sqm = price/land_size)
```

We might also want to get the price in Dollar and not RWF, so lets create a `price_usd` column:

```
property %>% mutate(price_usd = price/1000)
```

Another nice feature we can use is the `case_when` function inside the mutate:

```
property %>%
  mutate(nyarugenge = case_when(
    prop_district == "Nyarugenge" ~ "yes",
    TRUE ~ "no"
  ))
```

group_by() & summarise()

Just as in the `SQL` sessions we did, we might want to run an aggregation over a certain variable: district, age, rooms etc.

The same can be done in `R` using dplyr `group_by` and `summarise`. What do you think would the following return?

```
df %>%
  group_by(account_type) %>%
  summarise(total = sum(amount))
```

account_type	amount
savings	100
cheque	200
savings	100
savings	50
cheque	100
savings	500

account_type	total
savings	750
cheque	300

Ceci n'est pas une pipe %>%

Learning to love the pipe

Lets go through an example of where we extensively make use of the *and then* or *pipe* operator (`%>%`)

```
customer ← read_csv("~/data/transactions/customers.csv")
# A tibble: 569,887 × 3
```

customer	gender	age
cust346663	male	38
cust331051	male	20
cust174980	male	63
cust566268	female	17
cust70652	male	67
:	:	:
cust86436	female	31

Can you write the code to get the `mean` age per gender?

10:00

Learning to love the pipe

Lets go through an example of where we extensively make use of the *and then* or *pipe* operator (`%>%`)

```
customer ← read_csv("~/data/transactions/customers.csv")
# A tibble: 569,887 × 3
```

customer	gender	age
cust346663	male	38
cust331051	male	20
cust174980	male	63
cust566268	female	17
cust70652	male	67
:	:	:
cust86436	female	31

Can you write the code to get the `mean` age per gender?

```
customer %>%
  group_by(gender) %>%
  summarise(mean_age = mean(age))
```

Learning to love the pipe

```
transactions ← read_csv("~/data/transactions/transactions.csv")
# A tibble: 1,061,923 × 5
```

Can you write the code to get the `mean` and `median` amount (in $)[1] grouped by `transaction_date` and `sku`?

transaction_date	transaction_id	customer	sku	amount
2020-03-26	txn-85-1244	cust346663	cash_out	16763.6459
2020-04-05	txn-95-3398	cust331051	airtime	285.1081
2020-06-26	txn-177-2107	cust174980	p2p	12407.8653
2020-09-02	txn-245-0071	cust566268	p2p	10099.4391
2020-02-11	txn-41-4911	cust70652	p2p	4416.0183
2020-03-13	txn-72-0326	cust86436	momopay_other	4095.4230

15:00

[1] *Will have to divide* `amount` *by 1000.*

Learning to love the pipe

```
transactions %>%
  mutate(amount_dollars = amount/1000) %>%
  group_by(transaction_date, sku) %>%
  summarise(
    mean_amount = mean(amount),
    median_amount = median(amount)
      )
```

transaction_date	sku	mean_amount	median_amount
2020-01-02	airtime	297.3857	226.0844
2020-01-02	cash_in	18716.2975	15430.2207
2020-01-02	cash_out	12663.0678	11113.7757
2020-01-02	electricity	1358.6941	1135.4565
2020-01-02	from_bank	42178.9834	36886.1960
2020-01-02	momopay_other	22718.0697	19154.3812

[1] *Will have to divide* `amount` *by 1000.*

Housing Analysis with Rstudio 🏠

Recap on our questions

Lets recap on what we want to know about the property market:

- Which areas have the highest franc/sqm?
    - This requires us to *only* look at land (`filter`) and then derive a new column where we take price and divide by land size.
    - We then also need to order (`arrange`) on the derived column
- Has priced increased over time?
    - We need to calculate the `mean` price per time period.
    - Perhaps year is a good idea? So we will need to round the date column to first date of the year.
- What premium is there for an extra bedroom in Nyarugenge vs Outside of Nyarugenge?
    -
        - How much will it cost me to live inside Nyarugenge for 4 and 5 bedroom *houses* vs outside of Nyarugenge?
    - Here we will need a `case_when` to create a new column that calculates `mean` price when location is Nyarugenge and `mean` when not.
    - The do this by `group_by` bedroom.

Which areas have the highest franc/sqm?

- Which areas have the highest franc/sqm?
  - This requires us to *only* look at land (`filter`) and then derive a new column where we take price and divide by land size.
  - We then also need to order (`arrange`) on the derived column

prop_neighbourhood	avg_sqm_price
Kiyovu	234024
Gacuriro	91356
Rebero	73316
Kabeza	71345
Kibagabaga	66425
Remera	65870
Gikondo	51860
Gisozi	50504
Niboye	48829
Kimironko	43856

20:00

Which areas have the highest franc/sqm?

- Which areas have the highest franc/sqm?
  - This requires us to *only* look at land (`filter`) and then derive a new column where we take price and divide by land size.
  - We then also need to order (`arrange`) on the derived column

```
property %>%
  filter(prop_type == "Land") %>%
  mutate(sqm_price = price/land_size)
```

Which areas have the highest franc/sqm?

- Which areas have the highest franc/sqm?
  - This requires us to *only* look at land (`filter`) and then derive a new column where we take price and divide by land size.
  - We then also need to order (`arrange`) on the derived column

```
property %>%
  filter(prop_type == "Land") %>%
  mutate(sqm_price = price/land_size) %>%
  group_by(prop_neighbourhood) %>%
  summarise(avg_sqm_price = mean(sqm_price, na.rm = TRU
  arrange(desc(avg_sqm_price))
```

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.
  - Tip, use `lubridate::floor_date(date_creation, "year")` to `mutate` the date_creation column 😉

date_year	avg_price	nr_prices
2021-01-01	71776446	22
2020-01-01	77370796	136
2019-01-01	64184032	107

20:00

Has prices increased over time?

- We need to calculate the `AVG` price per time period.
- Perhaps month is a good idea? So we will need to round the date column to first date of the year.
  - Tip, use `lubridate::floor_date(date_creation, "year")` to `mutate` the date_creation column 😉

```
property %>%
  filter(prop_type == 'House') %>%
  mutate(date_creation = lubridate::floor_date(date_cre
  group_by(date_creation) %>%
  summarise(
    avg_price = mean(price),
    nr_prices = n()
  )
```

Premium for extra bedroom in Nyarugenge?

- What premium is there for an extra bedroom in Nyarugenge vs Outside of Nyarugenge?
    - How much will it cost me to live inside Nyarugenge for 4 and 5 bedroom *houses* vs outside of Nyarugenge?
    - Here we will need a `case_when` to create a new column that calculates `mean` price when location is Nyarugenge and `mean` when not.
        - Tip: in the summarise we can do conditionals using `ifelse`
        - `mean(ifelse(prop_district == "Nyarugenge", price, NA), na.rm = TRUE)`
    - The do this by `group_by` bedroom.

bedrooms	nr_houses	nyarugenge	other	price_diff
3	44	32654964	38898614	-6243650
4	168	92804800	61526121	31278679
5	35	48931040	141911931	-92980891

30:00

Premium for extra bedroom in Nyarugenge?

- What premium is there for an extra bedroom in Nyarugenge vs Outside of Nyarugenge?
  - How much will it cost me to live inside Nyarugenge for 4 and 5 bedroom *houses* vs outside of Nyarugenge?
  - Here we will need a `case_when` to create a new column that calculates `mean` price when location is Nyarugenge and `mean` when not.
    - Tip: in the summarise we can do conditionals using `ifelse`
    - `mean(ifelse(prop_district == "Nyarugenge", price, NA), na.rm = TRUE)`
  - The do this by `group_by` bedroom.

```
property %>%
  filter(bedrooms %in% c(3, 4, 5), prop_type == "House"
  group_by(bedrooms) %>%
  summarise(
    nr_houses = n()
  )
```

Premium for extra bedroom in Nyarugenge?

- What premium is there for an extra bedroom in Nyarugenge vs Outside of Nyarugenge?
  - How much will it cost me to live inside Nyarugenge for 4 and 5 bedroom *houses* vs outside of Nyarugenge?
  - Here we will need a `case_when` to create a new column that calculates `mean` price when location is Nyarugenge and `mean` when not.
    - Tip: in the summarise we can do conditionals using `ifelse`, `ifelse(prop_district ≠ "Nyarugenge", price, NA)`
  - The do this by `group_by` bedroom.

```
property %>%
  filter(bedrooms %in% c(3, 4, 5), prop_type == "House"
  group_by(bedrooms) %>%
  summarise(
    nr_houses = n(),
    nyarugenge = mean(
      ifelse(prop_district == "Nyarugenge", price, NA),
      na.rm = TRUE
      ),
    other = mean(
      ifelse(prop_district ≠ "Nyarugenge", price, NA),
      na.rm = TRUE
      ),
    price_diff = nyarugenge - other
  )
```