

# Research Fellows - Session Two

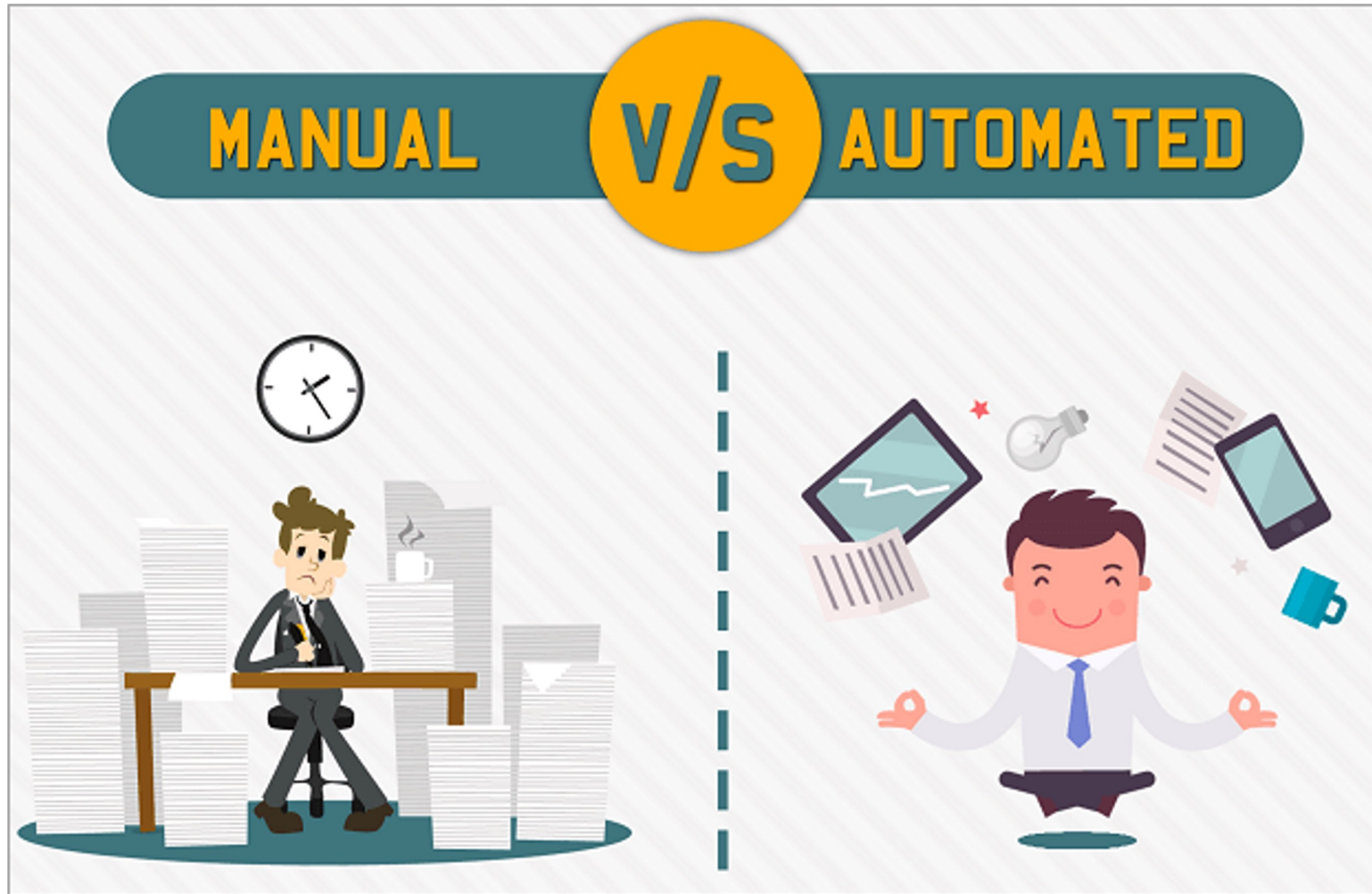
Automating Research Work

---

**Summer 2022**

Northwestern | Kellogg

# Automating and Scripting on KLC



# Today we will Cover

- Automation Advice in your Code
- Running the Code from a Terminal on KLC
- Scripting Multiple Tasks at Once
- Scheduling processes in cron

# Automate Checks/Changes in Code

# Automating Changes to Data

	A	B	C	D	E	F	G
			Commercial		Treasury	Treasury	
1	Month	Fed Funds Rate	Paper 3 month	Prime Rate	Rate 3 month	Rate 30 year	
227	2000-09	6.52	6.47	9.5	6.18	5.83	
228	2000-10	6.51	6.51	9.5	6.29	5.8	
229	2000-11	6.51	6.5	9.5	6.36	5.78	
230	2000-12	6.4	6.34	9.5	5.94	5.49	
231	2001-01	5.98	5.49	9.05	5.29	5.54	
232	2001-02	5.49	5.14	8.5	5.01	5.45	
233	2001-03	5.31	4.78	8.32	4.54	5.34	
234	2001-04	4.8	4.44	0.78	3.97	5.65	
235	2001-05	4.21	3.95	7.24	3.7	5.78	
236	2001-06	3.97	3.67	6.98	3.57	5.67	
237	2001-07	3.77	3.59	6.75	3.59	5.61	
238	2001-08	3.65	3.42	6.67	3.44	5.48	
239	2001-09	3.07	2.81	6.28	2.69	5.48	
240	2001-10	2.49	2.28	5.53	2.2	5.32	
241	2001-11	2.09	1.97	5.1	1.91	5.12	
242	2001-12	1.82	1.78	4.84	1.72	5.48	
243	Month	Fed Funds Rate	Commercial Paper 3 month	Prime Rate	Treasury Rate 3 month	Treasury Rate 30 year	
244	2002-01	1.73	1.7	4.75	5.45	1.68	
245	2002-02	1.74	1.79	4.75	5.4	1.76	
246	2002-03	1.73	1.86	4.75	ND	1.83	
247	2002-04	1.75	1.81	4.75	ND	1.75	
248	2002-05	1.75	1.78	4.75	ND	1.76	
249	2002-06	1.75	1.76	4.75	ND	1.73	

1. Change out of Range data points
2. Remove repeated header rows
3. Fix missing value notation: NDs to NAs

# Opening a GUI on KLC

Recall that no modules are preloaded in a new KLC session. You will need to load everything you use.

To see what version of a software package are available type:

```
module avail <software name>
module avail R
module avail stata
module avail python
```

To load something type:

```
module load <software version>
module avail R/4.0.3
module load stata/17
module load python/anaconda3.6
```

To launch a GUI:

```
rstudio
xstata-mp
spyder
```

# Launching Code from Command Line

# Launching Code from Terminal

To launch an R file:

```
Rscript <file_name.R>
```

To launch a python file:

```
python <file_name.py>
```

To launch a stata do file:

```
stata-mp -b do <file_name.do>
```

For the FEC example:

```
python 1_fec_extract.py --start=2018 --end=2020
```



# Scripting Multiple Tasks at Once

# Creating a Shell Script

If you'd like to run multiple files in sequence, you can create a shell script on KLC using the nano editor.

```
nano <file_name.sh>
```

Here is a simple example file:

```
#!/bin/bash

clear

# load modules
module load python/anaconda3.6
module load R/4.0.3

# run scripts
python file.py
Rscript file.R
```

To make the file executable:

```
chmod +x <file_name.sh>
```

# Running a Shell Script

To make the file executable:

```
chmod +x <file_name.sh>
```

To run the script:

```
./<file_name.sh>
```

or

```
source <file_name.sh>
```

For the FEC example:

```
source fec_all.sh
```

# Cron Jobs

# Creating a Cron Job

To create a cron job type the following from any node:

```
crontab -e
```

Note that you will need to enter your job in a vi editor.

- To insert text type “i”
- To exit text type “esc”
- To save changes type “:wq”
- To exit the file without saving changes type “:q!”

To enter a cron job for the `cron_example.sh` shell script and print the results to a file:

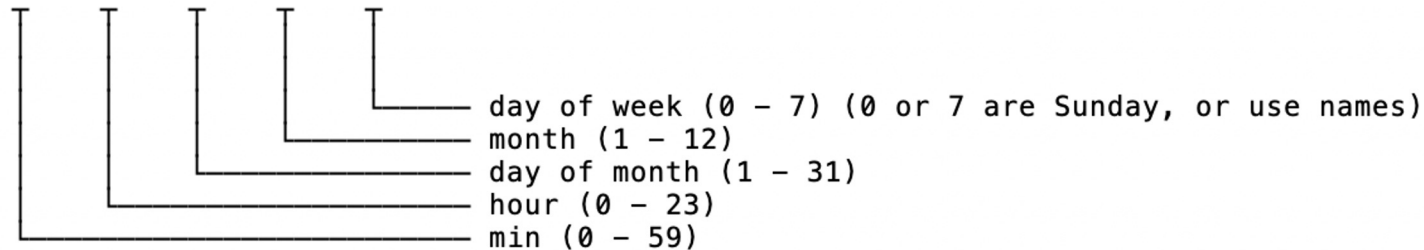
```
15 12 * * * sh ~/cron_example.sh > ~/cron_output.txt
```

```
45 12 * * * sh ~/fellows_workshop_2022/  
Session2_Automation/FEC/fec_cron.sh >  
~/fellows_workshop_2022/Session2_Automation/  
FEC/fec_cron_output.txt
```

# Crontab Guide

Entry	Description	Equivalent To	
@yearly (or @annually)	Run once a year at midnight in the morning of January 1		0 0 1 1 *
@monthly	Run once a month at midnight in the morning of the first of the month		0 0 1 * *
@weekly	Run once a week at midnight in the morning of Sunday		0 0 * * 0
@daily	Run once a day at midnight		0 0 * * *
@hourly	Run once an hour at the beginning of the hour		0 * * * *
@reboot	Run at startup	@reboot	

\* \* \* \* \* command to be executed



Useful Link to Cron Examples:  
<https://crontab.guru/examples.html>

# Crontab Examples

The following example will run every 10 minutes

```
*/10 * * * * /usr/bin/somedirectory/somecommand
```

The following example will run every 3 hours and 30 minutes

```
*/30 */3 * * * /usr/bin/somedirectory/somecommand
```

The following example will run every day at 8am

```
0 8 * * * /usr/bin/somedirectory/somecommand
```

The following example will run every Friday at 2pm

```
0 14 * * FRI /usr/bin/somedirectory/somecommand  
0 14 * * 6 /usr/bin/somedirectory/somecommand
```

The following example will run every month

```
0 0 1 * * /usr/bin/somedirectory/somecommand
```

The following example will run every quarter (4 times a year)

```
0 0 1 */3 * /usr/bin/somedirectory/somecommand
```

The following example will run the specified months only.

```
0 0 10 Jan, Apr, Jul, Oct * /usr/bin/somedirectory/somecommand
```

# Creating a Shell Script for Cron

Create a sample shell script for a cron job in the nano editor and follow the previous steps to make it executable:

```
nano cron_example.sh
```

In the nano editor, enter the following:

```
#!/bin/bash -l  
  
source ~/.bash_profile  
  
# load modules  
module load python/anaconda3.6  
  
# run scripts  
python file.py  
  
# print message  
echo "Cron Job is Running on KLC Node 5"
```



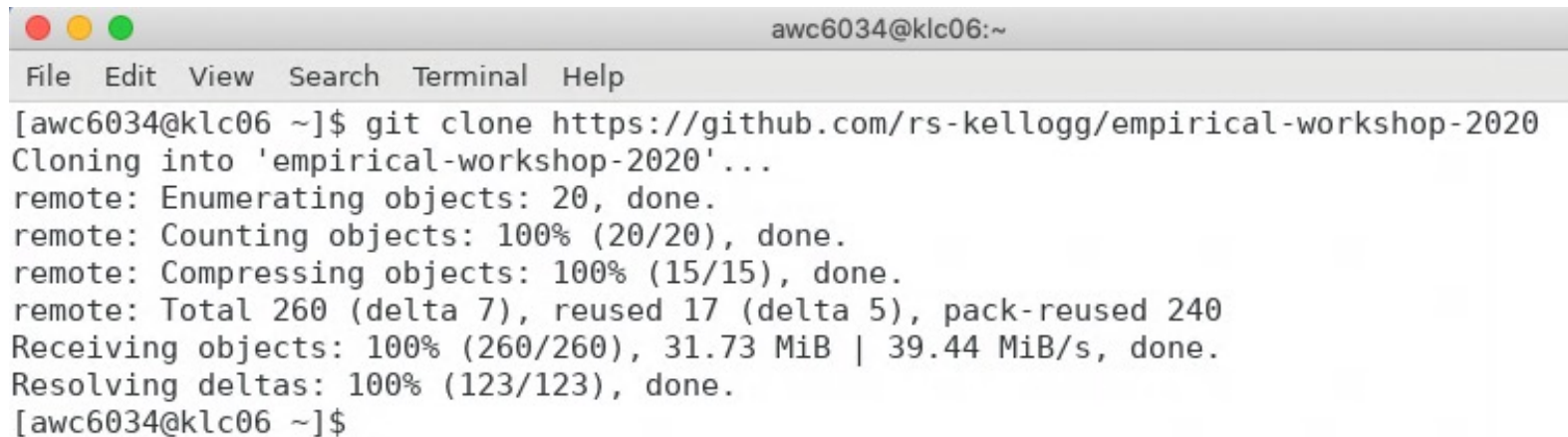
# Appendix

# Appendix: Git Clone Workshop to KLC

Recall that we'll first copy the contents of this week's github lecture notes/materials to our KLC home directories.

1. Launch a Terminal window on KLC
2. Type the following into the command line:

```
git clone https://github.com/rs-kellogg/fellows_workshop_2022/
```

A screenshot of a macOS Terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left and the text 'awc6034@klc06:~' on the right. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main area of the terminal displays the output of a 'git clone' command. The prompt is '[awc6034@klc06 ~]\$'. The output shows the repository being cloned into 'empirical-workshop-2020', followed by progress for enumerating, counting, and compressing objects, and finally receiving and resolving deltas. The prompt returns to '[awc6034@klc06 ~]\$' at the end.

```
awc6034@klc06:~  
File Edit View Search Terminal Help  
[awc6034@klc06 ~]$ git clone https://github.com/rs-kellogg/empirical-workshop-2020  
Cloning into 'empirical-workshop-2020'...  
remote: Enumerating objects: 20, done.  
remote: Counting objects: 100% (20/20), done.  
remote: Compressing objects: 100% (15/15), done.  
remote: Total 260 (delta 7), reused 17 (delta 5), pack-reused 240  
Receiving objects: 100% (260/260), 31.73 MiB | 39.44 MiB/s, done.  
Resolving deltas: 100% (123/123), done.  
[awc6034@klc06 ~]$
```

To update the contents of an existing cloned directory, navigate to the folder and type:

```
cd ~/fellows_workshop_2022  
git pull
```

# Appendix - Creating a Conda Environment

Instead of loading each module you would like to use in your shell script separately, you can also create conda environment. Follow the steps below only once to create the environment:

```
module load python-miniconda3/4.12.0
conda create -n automate_env r-essentials r-base
source activate automate_env
conda install r-fs
conda install pandas
```

To leave the environment:

```
source deactivate automate_env
```

In the future, you will only need to activate the environment to load all modules and libraries.

```
source activate automate_env
```