

FE8828 Programming Web Applications in Finance

Week 1: 0. Introduction 1. What's Internet? What's Web? 2. Launch into the Cloud 3. R Markdown and R Shiny/1: layout 4. R Programming/1

Dr. Yang Ye Email: yy@runchee.com

Nanyang Business School

Sep 17, 2020

- 1 0. Introduction to FE8828
- 2 Lecture 1: What's Internet? What's Web?
- 3 Lecture 2: Launch into the Cloud: Amazon Web Services
- 4 Lecture 3: R Markdown and R Shiny/1: layout
- 5 Lecture 4: R Programming

Section 1

0. Introduction to FE8828

Introduction

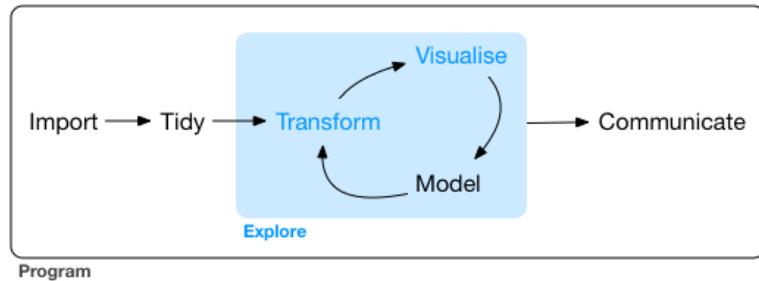
- With FE8828, I hope to bridge the theory and practice.
- Often, when we study, teachers feed us many theories. We learnt the “result” (and forgot most of them) but not the “process”.
- We learn about the “process” in this course.
- The “process” leads us to think how to apply theory, how theory work and how it doesn’t work. And better memory of the theory.
- Because finance is an empirical science. All theories come from observation and need to be validated by the real world.

Finance: a workflow point of view

- We go through such process with finance applications.
Data -> (Model <-> Application) -> Result
- Application is usually composed of
 - ▶ Input
 - ▶ Computing Engine for Valuation/Risk/Trading
 - ▶ Output
- Result is
 - ▶ Value
 - ▶ Risk
 - ▶ Trading signal
 - ▶ And ultimately, translate to profit and loss (“PnL”, “P&L”), or cost (i.e. different value and risk costs differently in capital lending rate).

Data science workflow

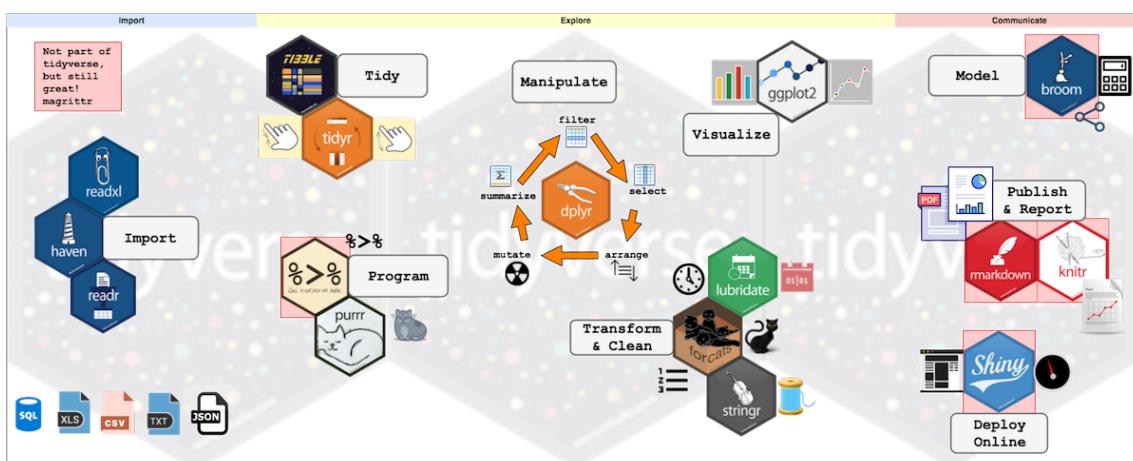
Data -> Model -> Application



- It resembles with the “Data Science” (DS) flow. Quants and DS share commonalities of working on data and model.
- That’s why “Quants” are pre-historical Data Scientist and DS are “modern quants in large”.

Programming/Internet is so powerful today!

- Programming practice has grown that there are tools for each step/stage of the workflow.
- With Internet/Web, data and computing power is also available at our fingers.



Learn the “process”

- It was difficult to do these in the past (see my later example on CAPM and Fama-French). By mastering the tools at different steps, we know the “process”. Basically, we know how to get things done!
- It is not going to be too easy, too. Everyone (in the profession) has the tool. We also need to be rooted in theory and a domain application to excel.

Example: How Fama-French 3-factor model expanded CAPM?

- CAPM (1961-1966):

$$R_a = R_{rf} + \beta_a * (R_m - R_{rf})$$

- Fama-French 3-Factor Model (1993):

$$R_{it} - R_{ft} = \alpha_{it} + \beta_1(R_{Mt} - R_{ft}) + \beta_2SMB_t + \beta_3HML_t + \epsilon_{it}$$

SMB stands for “Small [market capitalization] Minus Big” and HML for “High [book-to-market ratio] Minus Low”; they measure the historic excess returns of small caps over big caps and of value stocks over growth stocks.

Question

- What step/stage do we need for implementing Fama-French 3-Factor Model?

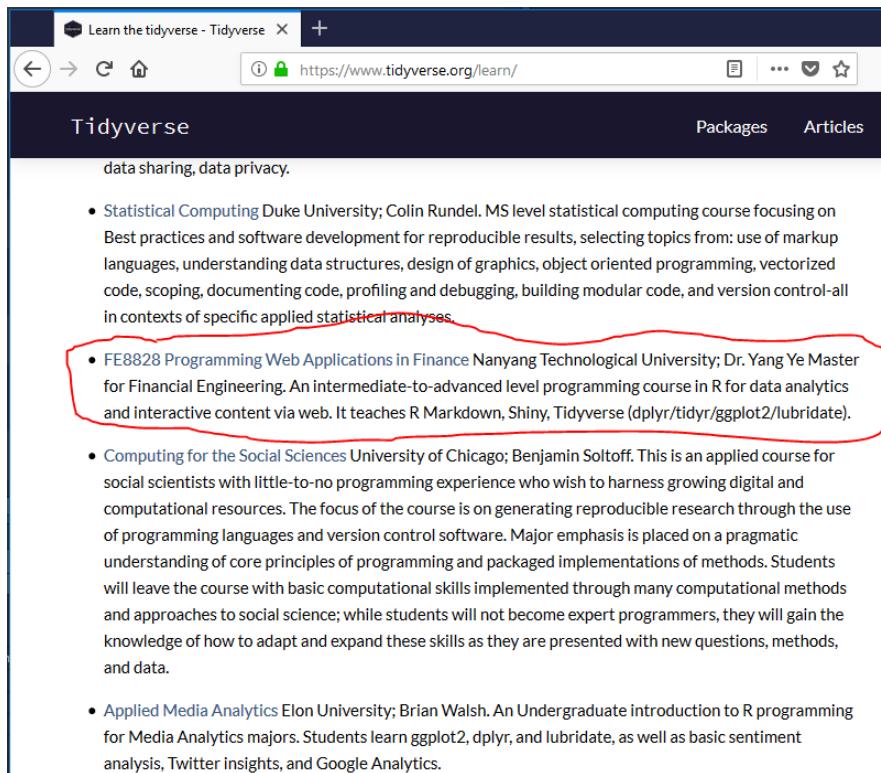
There are more endeavours to do!

- Descriptive data analysis
- EDA: exploratory data analysis
- Data cleaning
- ...
- Supply/demand analysis
- Option pricing and trading strategy
- Algorithmic trading
- Asset allocation
- Risk analytics
- Sentiment analysis
- Machine learning
- Deep learning
- ...

Our course: in detail

- Real-world finance application, asset allocation, trading strategy and derivatives valuation.
 - ▶ We will learn Programming/data science/finance in the context of Internet
 - ▶ We will build on from beginner R to intermediate R
- You will have ample exercise in this course - learn for your own achievement.

Our course: Tidyverse with a finance flavor



The screenshot shows a web browser window with the title "Learn the tidyverse - Tidyverse". The URL in the address bar is <https://www.tidyverse.org/learn/>. The page content is titled "Tidyverse" and includes a sub-section "data sharing, data privacy." Below this, there is a list of courses:

- Statistical Computing Duke University; Colin Rundel. MS level statistical computing course focusing on Best practices and software development for reproducible results, selecting topics from: use of markup languages, understanding data structures, design of graphics, object oriented programming, vectorized code, scoping, documenting code, profiling and debugging, building modular code, and version control-all in contexts of specific applied statistical analyses.
- FE8828 Programming Web Applications in Finance Nanyang Technological University; Dr. Yang Ye Master for Financial Engineering. An intermediate-to-advanced level programming course in R for data analytics and interactive content via web. It teaches R Markdown, Shiny, Tidyverse (dplyr/tidyr/ggplot2/lubridate).
- Computing for the Social Sciences University of Chicago; Benjamin Soltoff. This is an applied course for social scientists with little-to-no programming experience who wish to harness growing digital and computational resources. The focus of the course is on generating reproducible research through the use of programming languages and version control software. Major emphasis is placed on a pragmatic understanding of core principles of programming and packaged implementations of methods. Students will leave the course with basic computational skills implemented through many computational methods and approaches to social science; while students will not become expert programmers, they will gain the knowledge of how to adapt and expand these skills as they are presented with new questions, methods, and data.
- Applied Media Analytics Elon University; Brian Walsh. An Undergraduate introduction to R programming for Media Analytics majors. Students learn ggplot2, dplyr, and lubridate, as well as basic sentiment analysis, Twitter insights, and Google Analytics.

What does it take?

- Pick up a habit of good analyst:
 - ▶ Use reproducible research, well-organized.
 - ▶ Have a mind for data exploration
 - ▶ Have a mind for analysis: answer is not fixed but open-ended. You need to draw conclusion and make suggestion.
 - ▶ Have a mind for strategy thinking

Course Outline: Week 1-3

- Week 1:
 - ▶ What's Internet? What's Web?
 - ▶ (Optional) Launch into the Cloud: AWS
 - ▶ R Markdown and R Shiny/1: layout
 - ▶ R Programming/1
- Week 2:
 - ▶ R Programming/2
 - ▶ R Shiny/2: Building a web app
 - ▶ dplyr/1: Data Manipulation
- Week 3:
 - ▶ dplyr/2: EDA
 - ▶ Building Financial Applications/1

Course Outline: Week 4-6

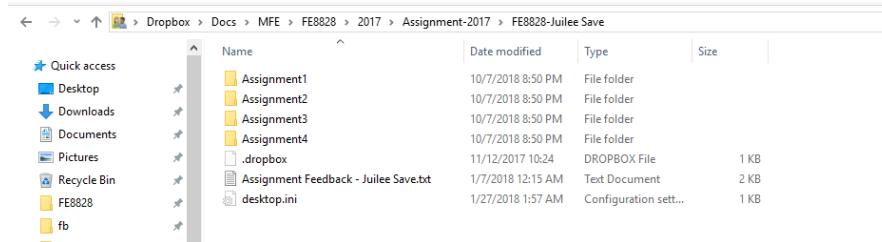
- Week 4
 - ▶ ggplot2: Data Visualization and EDA
 - ▶ R Shiny/3: Reactive
- Week 5
 - ▶ Building Financial Applications/2
 - ▶ Building Predictive Model
- Week 6:
 - ▶ Blockchain

Keep it Flexible

- I hope to cover some ad-hoc practical topics if feasible.

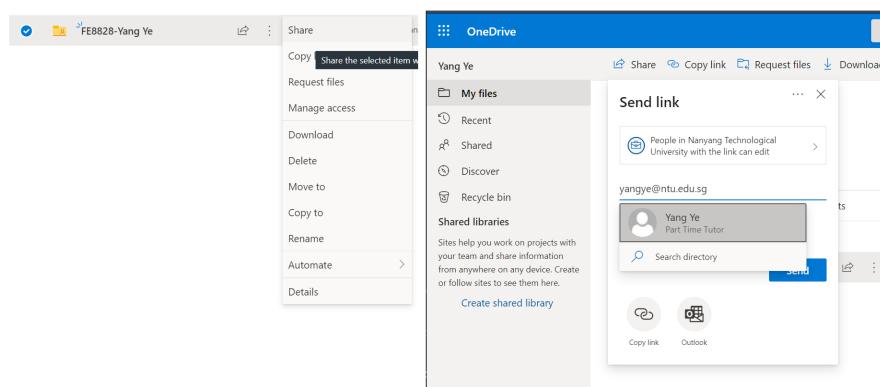
Assingment

- 4-5 individual assignments + 1 group assignment. No exam.
- Organization
 - ▶ Name your directory as FE8828-Your Name.
 - ▶ Organize your assignments into directories, e.g. Assignment 1, Assignment 2, ...



Submission by sharing

- Share the directory with *yy@runchee.com* on Google drive, or *yangye@ntu.edu.sg* with NTU One Drive.



Keep calm and code on



Questions?

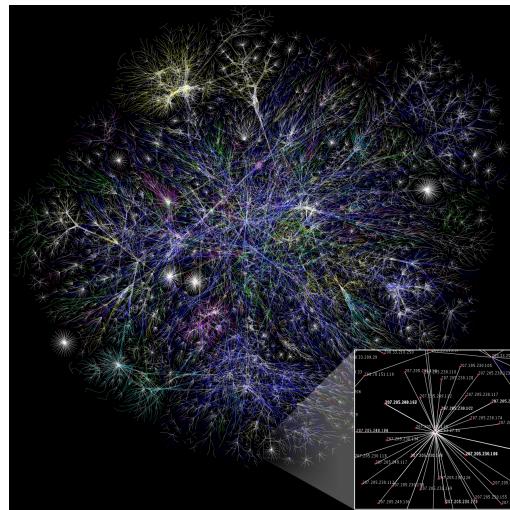
Section 2

Lecture 1: What's Internet? What's Web?

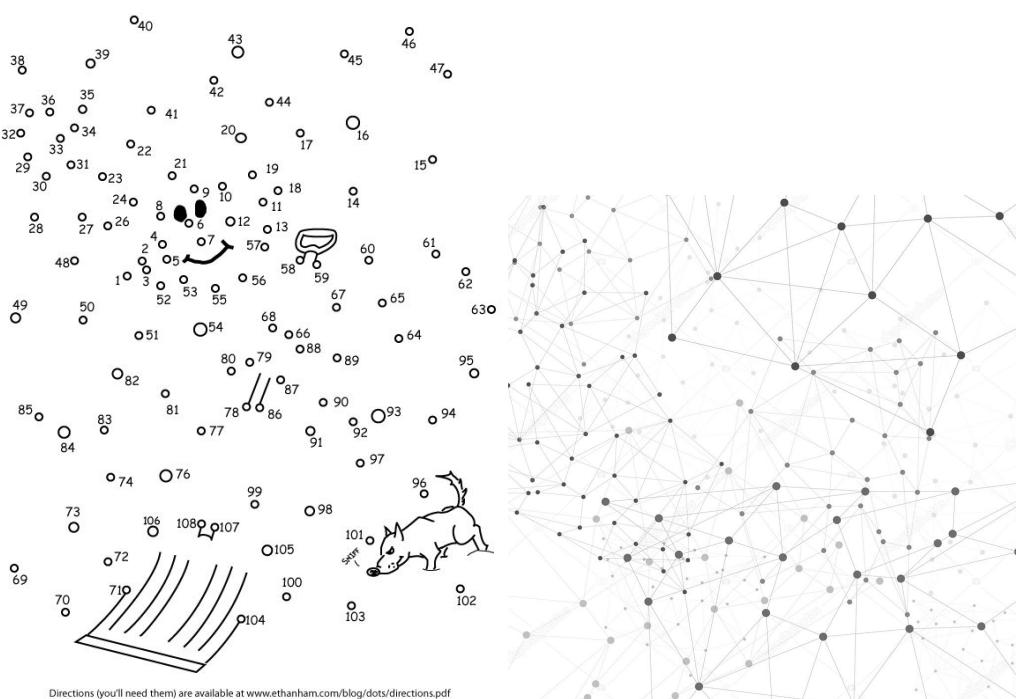
Lecture 1: What's Internet? What's Web?

Let's pick up the jargons!

- Network
- Internet
- HTTP/HTML/Web

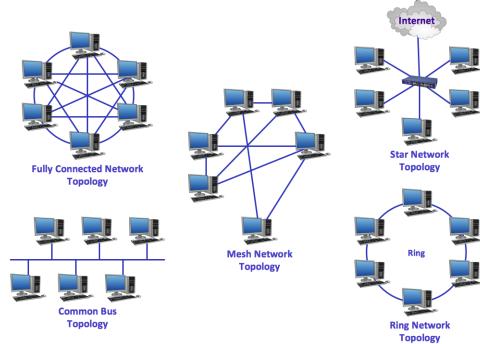


Network is to connect the dots/devices



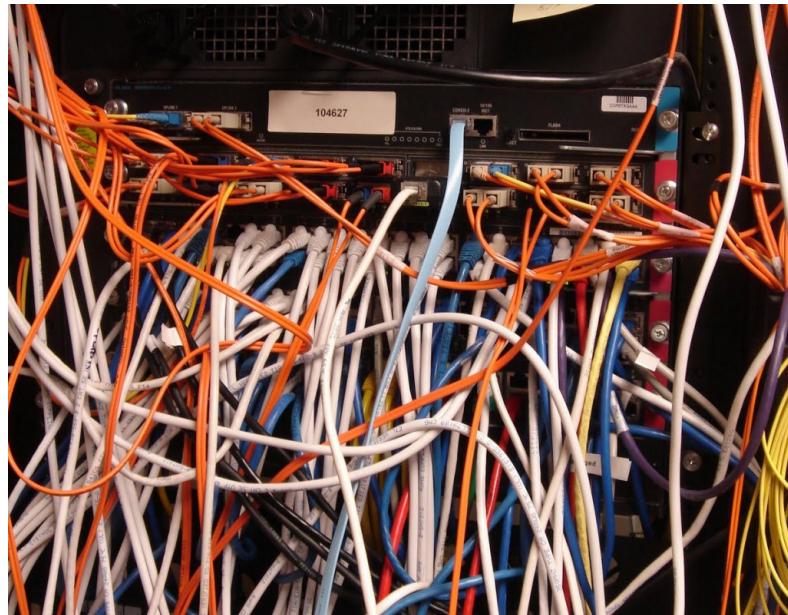
Building network

- There are many ways to connect the devices: Network topology
- Different network topology takes different way of communications.
- Fully-connected network is the most costly and robust. Ring is the cheapest but vulnerable.



Which network topology is our home Wi-Fi?

What runs inside the cables?

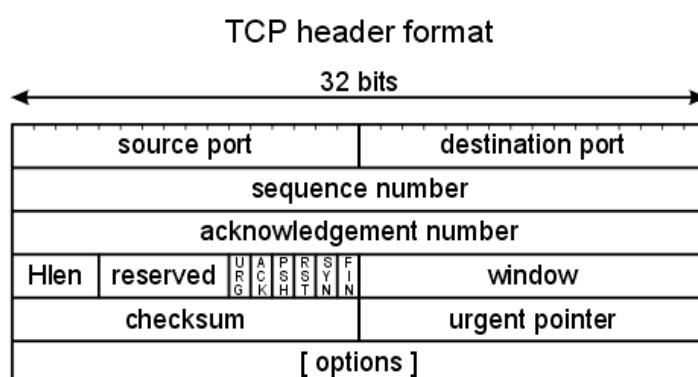


Network talks with Network Protocol

- ① Information turns to *packet* according to protocol specification.
- ② *Protocol* specifies the creation, transmission and receiving of *packet*.
- ③ Hardware infrastructure *routes* the *packets* to the destination.

Packet

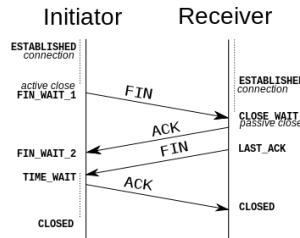
- ① Information turns to packet



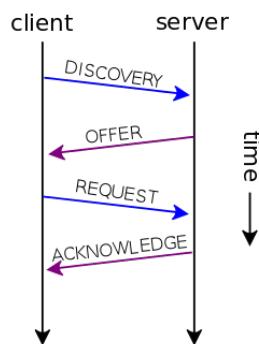
Protocol

- ② Protocol designs the packet and process

TCP Session

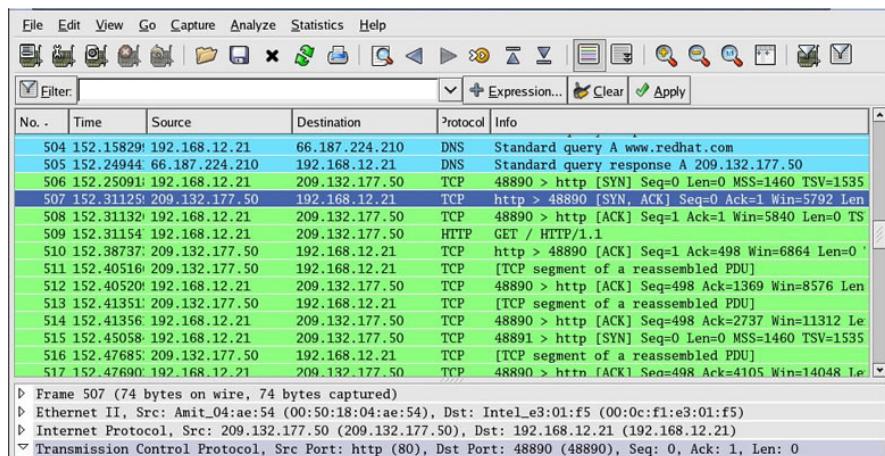


DHCP Session



Routing/Gateway

- ③ Infrastructure helps to *route* the packets to the destination.



TCP/IP and Internet

- What we learnt just now was implemented in large-scale in ARPANET in the 1970s by the Defense Advanced Research Projects Agency (DARPA)
- It's called the *TCP/IP* model that went on to build the Internet.

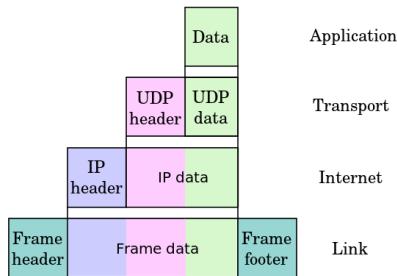
What does TCP/IP gives?

A family of protocols but what's most famous/“fundamental” is IP and TCP.

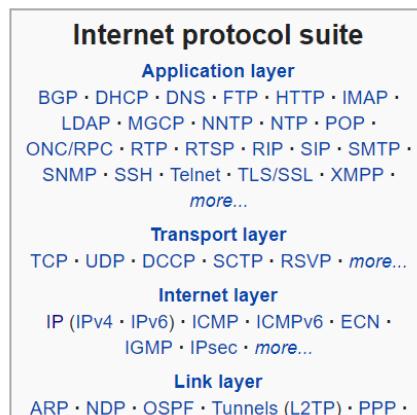
- ① IP (Internet Protocol): You shall heard of “IP address”, like 192.168.1.100.
- ② TCP (Transmission Control Protocol) / UDP (User Datagram Protocol)
 - ▶ TCP provides reliable, ordered, and error-checked delivery of a stream
 - ▶ UDP provides real-time transmission which can accept failure.

TCP/IP

③ Four Layers

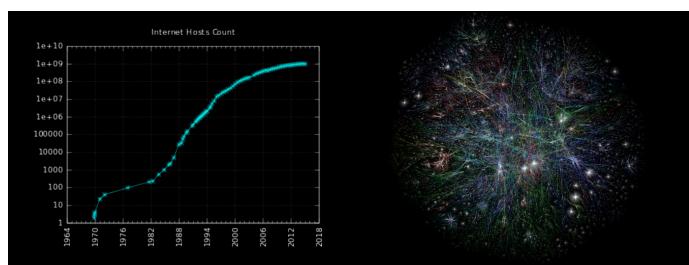


④ Application layer runs many protocols



Why Internet grewed in size?

- A family of expandable network protocols: easy join, allows failure. That's also what we want to learn to do.



- IETF (Internet Engineering Task Force) maintains and still gets new protocol approved.



What happens after plugging cable, or turning on Wi-Fi, turn on the 5G phone?

- New network device sends a join request.
- Network gateway receives the message, allocate an IP address for the new device.
- Device accepts the IP address and uses it to label for itself. Router also knows where to send the packet.
- This auto-configure process is carried out by *Dynamic Host Configuration Protocol* (DHCP) protocol.

When it wants to visit someone on the network? DNS

- DNS is the directory service for internet.
 - ▶ We don't use *123.456.789.012* but *www.google.com*.
 - ▶ One kind of attack to Internet is to hijack/brings down Root Domain Servers for Global (8 of them) or a country's root DNS.
 - ▶ (Demo with nslookup.)



When it wants to visit someone on the network? HTTP/HTML

- Now we shall have an idea of how Internet works, let's move on to Web ("World Wide Web")
- Initiated by Tim Berners-Lee at CERN (where big collision happens in "The Large Hadron Collider") in 1989.
 - ▶ Is he supposed to be a nuclear physicist?
- Tim initiated/invented both the HTTP (protocol) and HTML (content).

HTTP Request/Response

- Request

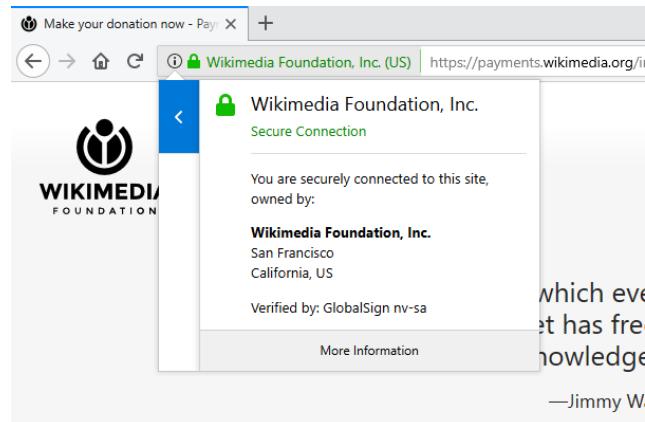
```
GET /index.html HTTP/1.1
Host: www.example.com
```

- Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    Hello World, this is a very simple HTML document.
  </body>
</html>
```

HTTP/HTTPS/HTML

- Hypertext Transfer Protocol (HTTP) but obviously, it does file, music, anything else now.
- It's a clear text protocol. That's why we need to use HTTPS (HTTP on SSL) to secure the communication.
- In browser, you can see the green lock in address bar. This is about digital certificate, associated with cryptography and authentication. Another topic.



Web

When you have HTML and URL (Uniform Resource Locator), Web is born.

- Website: <https://en.wikipedia.org/>
 - ▶ Document: https://en.wikipedia.org/wiki/World_Wide_Web
- Resource:
https://en.wikipedia.org/wiki/World_Wide_Web#/media/File:Web_Index.svg

How this resource (URL) is used in document (HTML).

```
<div class="thumbinner" style="width:302px;">
  <a href="/wiki/File:Web_Index.svg" class="image">
    
  </a>
</div>
```

Web Browser

First generation (1993)



Web application

- Static v.s. Dynamic
- Dynamic website display content based on user input.
- Supported by HTML/CSS/JavaScript. HTML 5, CSS 3 and JavaScript 7 .
- App also uses HTML/CSS/JS.



Web application

- Why it is important? Needless to say. It is not 1995 anymore.
 - ▶ Easy to deploy: no copy needed
 - ▶ Runs fast: every browser is optimized
 - ▶ Easy to develop: less effort than Mobile App and cross-platform.



What would we like to do?

Now comes the final part

- With the Internet infrastructure, we can get finance data (price, economic indicators) easily
- We can use the tools to process them, pass them through model and computing engine
- AS the result, we will have an application.
- We don't need to write HTML/CSS/JavaScript. We will write R and it assembly HTML/CSS/JavaScript for us.

Take-home

Why the Internet has succeeded?

- Information flows by packet.
- Protocol enables easy join, allows failure.
 - Local device just needs to send the packet
 - Network routers/gateways does the route/transmission to the destination.
 - Scalable and Efficient

Why Web has succeeded?

- Web helps to locate universal resource.
- Web helps to organize them in one place.

Take-home

Finance application

- Any finance application is a data application.
- Internet gave us the access to data and computing power.

Inspiration

- Internet is a recent invention that's just 30-something years old.
- Quant finance is also young: (Bachelier 1900s), Markowitz 50s, CAPM 60s, APT/Black-Scholes 70s, Quant finance 80s, Factor research 90s, High-Frequency trading 00s, GFC/QE 08, Robo-invest/ML/DL 10s, post-COVID 20s (hope not too long) ...
- Both are journeys of inventions and mavericks that inspire you to live to the future of quant research.
- We are learning/using recent inventions and have the opportunities to bring them forward.

Section 3

Lecture 2: Launch into the Cloud: Amazon Web Services

Lecture 2: Launch into the Cloud: Amazon Web Services

- Sign-up for AWS Account
- Setup AWS for EC2
- Launch EC2
- Running R

Disclaimer:

- ① I am not owning Amazon shares directly or indirectly.
- ② I don't plan to long AMZN during the course of this course.
- ③ I am not working for Amazon and I don't get paid by doing this except receiving AWS educate credit.

Amazon Web Services

History:

While Amazon standardized its internal infrastructure for all teams, it found the opportunity of open it up to the public and make computing infrastructure available to all people. Amazon thus becomes the new utility company for the Internet age, supplying crucial infrastructure like electricity and water.

We can almost attribute the inventor of “Cloud”, “Cloud services”, “Cloud computing” to Amazon.

Now the “cloud” market has many competitor, with AWS still holds about 33% (Apr 2018)

Sign-up for AWS as a student

1. Sign-up with an AWS account at <https://aws.amazon.com>
 - AWS account needs to be fully activated by completing phone verification steps and adding a valid credit card.
2. Sign-up AWS Educate account at
<https://aws.amazon.com/education/awseducate/>
 - They will need to select the AWS Account option and enter their 12 digit AWS Account ID number when applying to AWS Educate.
3. (Depends) Because NTU is not (yet) listed as a institutude, you may need to apply credit from AWS Educate separately
<https://aws.amazon.com/education/awseducate/contact-us/>
4. Claim credit from AWS according according to the instruction of AWS

AWS Free Tier for New Account

AWS Free Tier (12 Month Introductory Period):

These free tier offers are only available to new AWS customers, and are available for 12 months following your AWS sign-up date. When your 12 month free usage term expires or if your application use exceeds the tiers, you simply pay standard, pay-as-you-go service rates (see each service page for full pricing details). Restrictions apply; see [offer terms](#) for more details.

Elastic Compute Cloud (EC2)

- 750 hours of [Amazon EC2](#) Linux t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month*
- 750 hours of [Amazon EC2](#) Microsoft Windows Server t2.micro instance usage (1 GiB of memory and 32-bit and 64-bit platform support) – enough hours to run continuously each month*
- 750 hours of an [Elastic Load Balancer](#) shared between Classic and Application load balancers, 15 GB data processing for Classic load balancers, and 15 LCUs for Application load balancers*
- 30 GB of [Amazon Elastic Block Storage](#) in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with EBS Magnetic) and 1 GB of snapshot storage*
- 500 MB-month of [Amazon EC2 Container Registry](#) storage for new customers*

Amazon Simple Storage Service (S3)

- 5 GB of [Amazon S3](#) standard storage, 20,000 Get Requests, and 2,000 Put Requests*

Data Transfer

- 15 GB of data transfer out and 1GB of regional data transfer aggregated across all AWS services*

Amazon Data Pipeline

- 3 low frequency preconditions running on AWS per month*
- 5 low frequency activities running on AWS per month*

Amazon ElastiCache

- 750 hours of [Amazon ElastiCache](#) cache.t2micro Node usage - enough hours to run continuously each month.*

Amazon CloudFront

- 50 GB Data Transfer Out, 2,000,000 HTTP and HTTPS Requests of [Amazon CloudFront](#)*

Amazon API Gateway

- 1 Million API Calls per month*

Yang Ye | FE8828 | 2020

6

Register for AWS Educate account

<https://www.awseducate.com/registration>

The screenshot shows the AWS Educate registration process. At the top, there's a logo for 'aws educate' and a button to 'Apply to join AWS Educate'. Below this, a blue header bar says 'Step 1/3: Choose your role'. There are five categories listed in rows of two:

Role	Action
Student	
Educator	
US Veteran	
Institution	
Company/Recruiter	

7

Apply AWS Educate Credit

Name Yang Ye
Account 457734848212

Regarding* Account and Billing Support
 Service Limit Increase
 Technical Support
Unavailable under the Basic Support Plan

Service* Educate

Category* Credit Inquiry

Subject* Applying credit

Description Guidance
AWS Educate provides educators and students with grant-based access to AWS, training, and content, while also providing educators and students with a forum for collaboration. If you would like additional information about AWS Educate, please visit <https://aws.amazon.com/education/awseducate/>.

Description* I am doing a course in NTU Singapore. I would like to request for education credit for my account snid

Yang Ye | FE8828 | 2020

<https://aws.amazon.com/education/awseducate/contact-us/>

8

2. Setup AWS

After you login to AWS

1. Change your region to **Singapore**.
2. Add shortcuts in the navigation bar
 - a. EC2
 - b. VPC



Yang Ye | FE8828 | 2020

9

2. Create VPC Network

Search for VPC

The screenshot shows the AWS Management Console home page. At the top, there is a search bar with the text "vpc". Below the search bar, the "AWS services" section is visible, with "VPC" highlighted under "Isolated Cloud Resources". Other services listed include EC2, IAM, Compute, Developer Tools, Internet of Things, and various AWS tools like CodeStar, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, X-Ray, AWS IoT, AWS Greengrass, Contact Center, and Amazon Connect.

Yang Ye | FE8828 | 2020

10

2.1 VPC: Click “Start VPC Wizard”

The screenshot shows the VPC Dashboard. On the left, there is a sidebar with options: "Virtual Private Cloud", "Your VPCs", "Subnets", "Route Tables", and "Internet Gateways". On the right, the main area is titled "Resources" and contains two buttons: "Start VPC Wizard" and "Launch EC2 Instances". A note below the buttons states: "Note: Your Instances will launch in the Asia Pacific (Singapore) region". Below this, it says: "You are using the following Amazon VPC resources in (Singapore) region:" followed by a table of resource counts.

2 VPCs	2 Internet Gateways
0 Egress-only Internet Gateways	3 Subnets
3 Route Tables	2 Network ACLs
0 Elastic IPs	0 VPC Peering Co

Yang Ye | FE8828 | 2020

11

2.2 VPC two steps - follow defaults, add a name

Step 1: Select a VPC Configuration

VPC with a Single Public Subnet

Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet. Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.

Creates:

A /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.

Select

Step 2: VPC with a Single Public Subnet

IPv4 CIDR block: * 10.0.0.0/16 (65531 IP addresses available)

IPv6 CIDR block: No IPv6 CIDR Block Amazon provided IPv6 CIDR block

VPC name: mfe

Public subnet's IPv4 CIDR: * 10.0.0.0/24 (251 IP addresses available)

Availability Zone: * No Preference

Subnet name: Public subnet

You can add more subnets after AWS creates the VPC.

Service endpoints

Add Endpoint

Enable DNS hostnames: * Yes No

Hardware tenancy: * Default

Yang Ye | FE8828 | 2020

12

2.3 Subnet: Change settings. Tick auto-assign public IP4 address

88 | ap-southeast-1.console.aws.amazon.com/vpc/home

AWS Services Resource Groups EC2 VPC Route Tables

VPC Dashboard Filter by VPC: Select a VPC

Virtual Private Cloud Your VPCs Subnets Route Tables

Create Subnet Subnet Actions

Search Subnet

Name State VP

Public subnet subnet-834b82ca available vpc

Subnet Actions

Delete Subnet Edit IPv6 CIDRs Create Flow Log Modify auto-assign IP settings

Modify auto-assign IP settings

Enable auto-assign public IPv4 or IPv6 addresses to automatically request an IP address for instances launched into this subnet.

Auto-assign IPs Enable auto-assign public IPv4 address

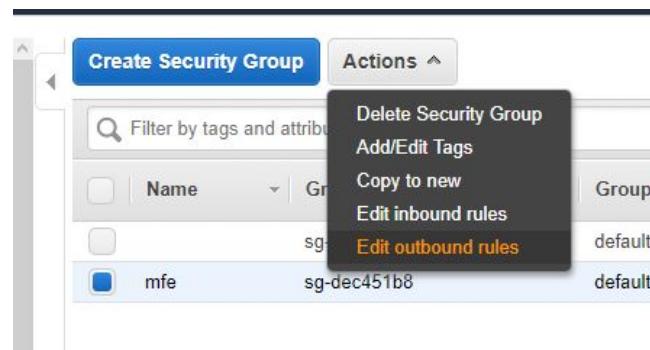
Note: You can override the auto-assign IP settings for each individual instance at launch time for IPv4 or IPv6. Regardless of how you've configured the auto-assign public IP feature, you can assign a public IP address to an instance that has a single, new network interface with a device index of eth0.

Cancel Save

Yang Ye | FE8828 | 2020

2.4 In VPC wizard, it created new security group.

Selected the newly created security group.
Edit Inbound rules and edit outbound rules.



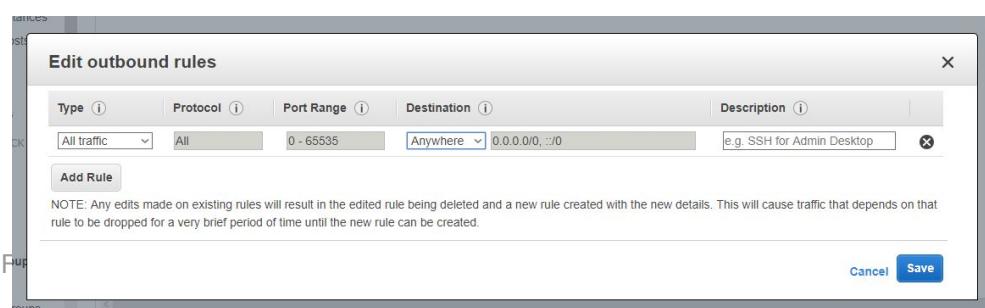
Yang Ye | FE8828 | 2020

14

2.4 Inbound/Outbound: choose All traffic/Anywhere

An inbound firewall protects the network against incoming traffic from the internet or other network segments, namely disallowed connection from outside.

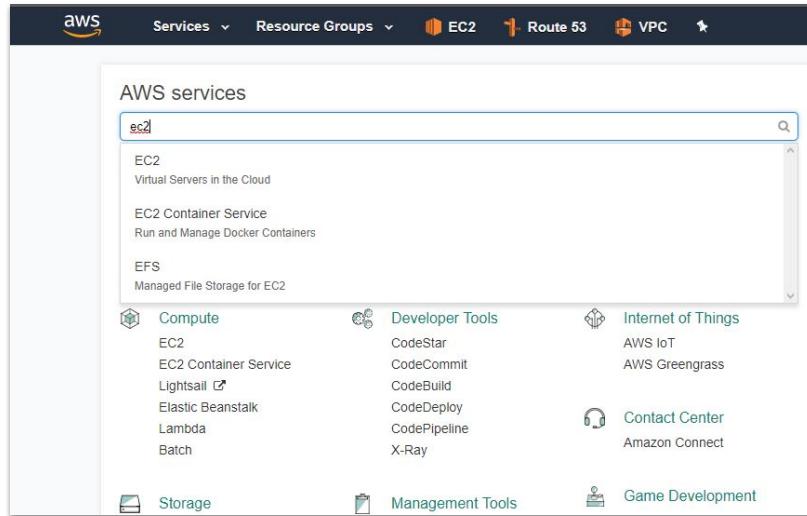
Outbound rules determines what application can connect to the outside.



Yang Ye | F

15

3. Go to EC2



Yang Ye | FE8828 | 2020

16

3.1 Ready to Launch

A screenshot of the EC2 Dashboard. On the left is a sidebar with navigation links: 'EC2 Dashboard', 'Events', 'Tags', 'Reports', 'Limits', 'INSTANCES', 'Instances', 'Spot Requests', 'Reserved Instances', 'IMAGES', 'AMIs', 'Bundle Tasks', 'ELASTIC BLOCK STORE', 'Volumes', 'Snapshots', 'NETWORK & SECURITY', 'Security Groups', 'Elastic IPs', 'Placement Groups', 'Key Pairs', 'Network Interfaces', 'LOAD BALANCING', 'Load Balancers', and 'AUTO SCALING', 'Launch Configurations', 'Auto Scaling Groups'. The main content area has a heading 'Resources' with a note: 'You are using the following Amazon EC2 resources in the US East (N. Virginia) region: 0 Running Instances, 0 Volumes, 0 Key Pairs, 0 Placement Groups, 0 Elastic IPs, 0 Snapshots, 0 Load Balancers, 1 Security Groups'. Below this is a 'Create Instance' section with a red box around the 'Launch Instance' button. A note says: 'To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.' At the bottom of this section is a note: 'Note: Your instances will launch in the US East (N. Virginia) region'. To the right of this is a 'Service Health' section with 'Service Status' (US East (N. Virginia): The service is operating normally) and 'Availability Zone Status' (us-east-1a: Availability zone is operating normally, us-east-1b: Availability zone is operating normally, us-east-1c: Availability zone is operating normally, us-east-1e: Availability zone is operating normally). There's also a 'Scheduled Events' section with 'US East (N. Virginia): No events'.

Yang Ye | FE8828 | 2020

17

EC2 Step 1: Community AMI: search for “RStudio”

RStudio-1.1.456_R-3.5.1_CUDA-9.0_cuDNN-7.2.1_ubuntu-16.04-LTS-64bit

Id: ami-07a45f81350cb6584 in Singapore region
created by http://www.louisaslett.com/RStudio_AMI/

Newer than below

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

The screenshot shows the AWS Lambda console with a search bar at the top containing 'rstudio'. Below the search bar is a table of search results. The first result is highlighted with a blue circle and labeled 'Selected'. The table includes columns for the AMI name, ID, description, root device type, virtualization type, and a 'Select' button. There are 9 AMIs listed in total, with the first one being the selected RStudio AMI.

AMIs	Name	ID	Description	Root Device Type	Virtualization Type	Select
1	RStudio-1.1.456_R-3.5.1_CUDA-9.0_cuDNN-7.2.1_ubuntu-16.04-LTS-64bit	ami-07a45f81350cb6584	Ready to run RStudio sever for statistical computation (www.louisaslett.com). Connect to instance public DNS in web brower (standard port 80), username rstudio and password is instance ID	ebs	hvm	Selected
2	RStudio-1.1.383_R-3.4.2_Julia-0.6.0_CUDA-8_cuDNN-6_ubuntu-16.04-LTS-64bit	ami-5b9bde38	Ready to run RStudio + Julia/Python server for statistical computation (www.louisaslett.com). Connect to instance public DNS in web brower (standard port 80), username rstudio and password rstudio	ebs	hvm	Select
3	OMPI_R_RStudioServer	ami-9797204	[Copied ami-9fb77587 from us-east-1] OMPI_R_RStudioServer	ebs	hvm	Select
4	RStudio-1.0.153_R-3.4.1_Julia-0.6.0_ubuntu-16.04-LTS-64bit	ami-a13b59c2	Ready to run RStudio + (experimental) Julia server for statistical computation (www.louisaslett.com). Connect to instance public DNS in web brower (standard port 80), username rstudio and password rstudio	ebs	hvm	Select
5	SATT Analytics Platform - Base-a98c7627-bb8b-4cc2-89bd-a69e52b8431b-ami-65772400.2	ami-ac091afe	An Advanced Analytics Platform, based on R Foundation. The Platform is equipped with RStudio server, R web application framework, Interactive visualization and dynamic report generation packages.	ebs	hvm	Select
6	RStudio-1.0.143_R-3.4.0_Julia-0.5.2_ubuntu-16.04-LTS-64bit	ami-c6db5da5	Ready to run RStudio + (experimental) Julia server for statistical computation (www.louisaslett.com). Connect to instance public DNS in web brower (standard port 80), username rstudio and password rstudio	ebs	hvm	Select
7	RStudio-1.0.143_R-3.4.0_Julia-0.5.2_ubuntu-16.04-LTS-64bit	ami-c6db5da5	Ready to run RStudio + (experimental) Julia server for statistical computation (www.louisaslett.com). Connect to instance public DNS in web brower (standard port 80), username rstudio and password rstudio	ebs	hvm	Select

Yang Ye | FE8828 | 2020

18

EC2 Step 2: Choose instance type. You can enjoy having one instance of **t2.micro** during 12-month free-tier.

- Upgrade to higher instance type when you have received your AWS educate credit.
- T2.medium/t2.large should be good enough. There are more expensive ones.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

The screenshot shows the AWS Lambda console with a search bar at the top containing 'rstudio'. Below the search bar is a table of search results. The first result is highlighted with a blue circle and labeled 'Selected'. The table includes columns for the AMI name, ID, description, root device type, virtualization type, and a 'Select' button. There are 9 AMIs listed in total, with the first one being the selected RStudio AMI.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium \$	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large \$	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes

19

EC2 Step 3: Make sure Auto-assign Public IP is ticked. Click “Next”.

The screenshot shows the 'Step 3: Configure Instance Details' page of the AWS EC2 instance creation wizard. The top navigation bar includes tabs for '1. Choose AMI', '2. Choose Instance Type', '3. Configure Instance' (which is selected and highlighted in orange), '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review'. The main section is titled 'Step 3: Configure Instance Details' with the sub-instruction 'Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of instance, and more.' Below this, there are several configuration fields:

- Number of instances:** Set to 1, with an option to 'Launch into Auto Scaling Group'.
- Purchasing option:** An unchecked checkbox for 'Request Spot instances'.
- Network:** Set to 'vpc-d3ebe1b7 | mfe' with a 'Create new VPC' button.
- Subnet:** Set to 'subnet-834b82ca | Public subnet | ap-southeast-1b' with a 'Create new subnet' button and a note '251 IP Addresses available'.
- Auto-assign Public IP:** A dropdown menu set to 'Use subnet setting (Enable)', which is circled in blue.
- IAM role:** Set to 'None' with a 'Create new IAM role' button.
- Shutdown behavior:** Set to 'Stop'.
- Enable termination protection:** An unchecked checkbox.
- Monitoring:** An unchecked checkbox for 'Enable CloudWatch detailed monitoring' with a note 'Additional charges apply'.
- Tenancy:** Set to 'Shared - Run a shared hardware instance' with a note 'Additional charges will apply for dedicated tenancy'.

On the left side of the form, the name 'Yang' is visible. On the right side, the number '20' is present.

EC 2 Step 4: Add Storage, use the default setting.

- One AMI image can be launched into many instances so its content is static.
- The author of AMI image configures a persistent storage so it saves any changes we did.
- Most importantly, we don't need to do anything now.

EC Step 6: Click “6. Configure Security Group”

Select an existing security group. Your previous changes are loaded.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more about Amazon EC2 security groups.](#)

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-dec451b8	default	default VPC security group	Copy to new

Inbound rules for sg-dec451b8 (Selected security groups: sg-dec451b8)

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

haha!!!

Yang Ye | FE8828 | 2020

22

EC 2 Step 7: Review and Launch

Click the blue button.

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details [Edit AMI](#)

Studio-1.0.153_R-3.4.1_Julia-0.6.0_ubuntu-16.04-LTS-64bit - ami-a13b59c2
Ready to run RStudio + (experimental) Julia server for statistical computation (www.louisaslett.com). Connect to Instance public DNS in web brower (standard port 80), username rstudio and password rstudio.
Root Device Type: ebs Virtualization type: hvm

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups [Edit security groups](#)

Security Group ID	Name	Description
sg-dec451b8	default	default VPC security group

All selected security groups inbound rules

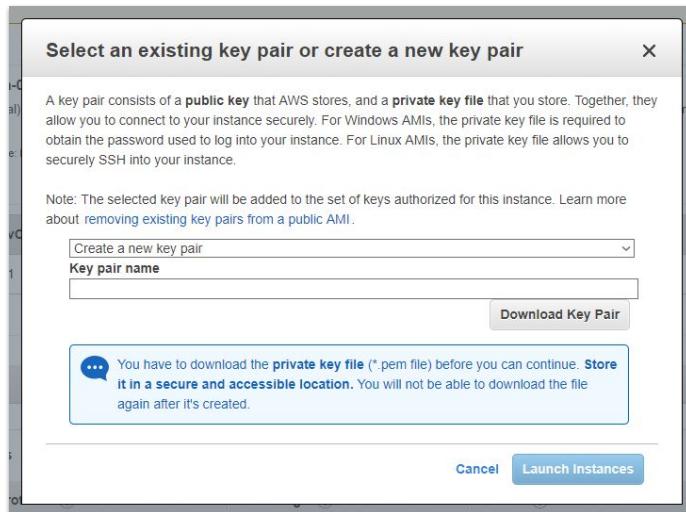
Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

Launch

Yang Ye | FE8828 | 2020

23

One last thing: Key pair
Create if you didn't have an existing one or lost the previous download.



Yang Ye | FE8828 | 2020

24

If you know SSH

You can connect to the server by

- Windows: Download Git for Windows from <https://git-scm.com/download/>.

```
ssh -i 'c:\Users\yourusername\Downloads\MyKeyPair.pem' ubuntu@{IP_Address}
```

- Mac: skip download Git. Go straight

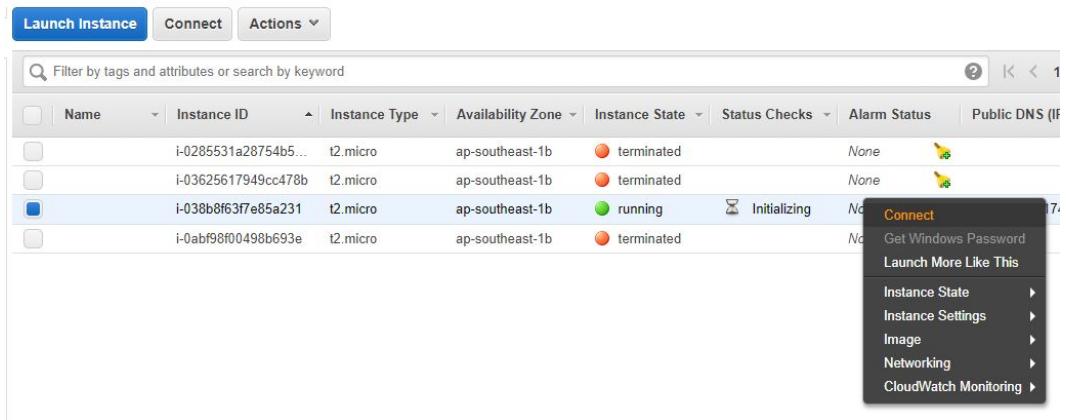
```
ssh -i ~/Downloads/MyKeyPair.pem ubuntu@{IP_Address}
```

Yang Ye | FE8828 | 2020

25

Control the instance.

- Connect gives information.
- Stop but not to terminate, which deletes all data from EBS.
- A running instance charges.
- A stopped instance will charge a small fee for the storage. Our configured size is 10G, within the free-tier for new account within 12-month.
- Use AWS calculator

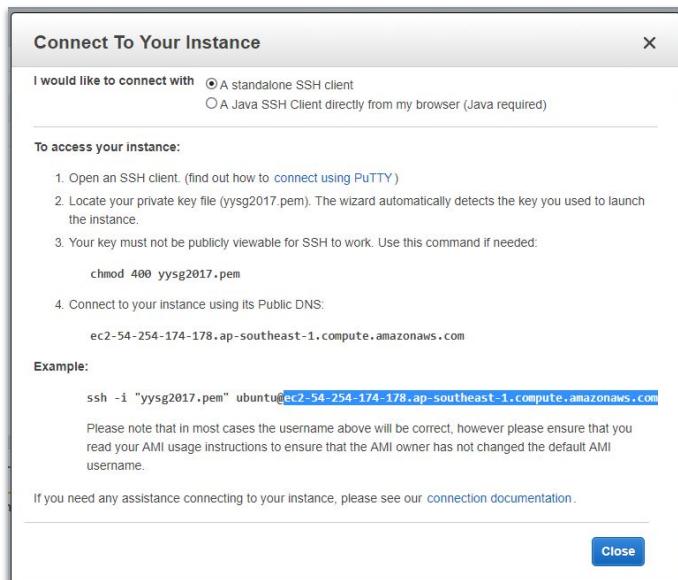


Yang Ye | FE8828 | 2020

26

Access it

- Copy this address



Yang Ye | FE8828 | 2020

27

<http://ec2-54-254-174-178.ap-southeast-1.compute.amazonaws.com/>

This would change every time when you launch it.

Initial password: rstudio/Instance ID.

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. A search bar is followed by a table header: 'Name' (dropdown), 'Instance ID', 'Instance Type', 'Availability Zone', 'Instance State', and 'Status'. Below the table, there is a list of EC2 instances:

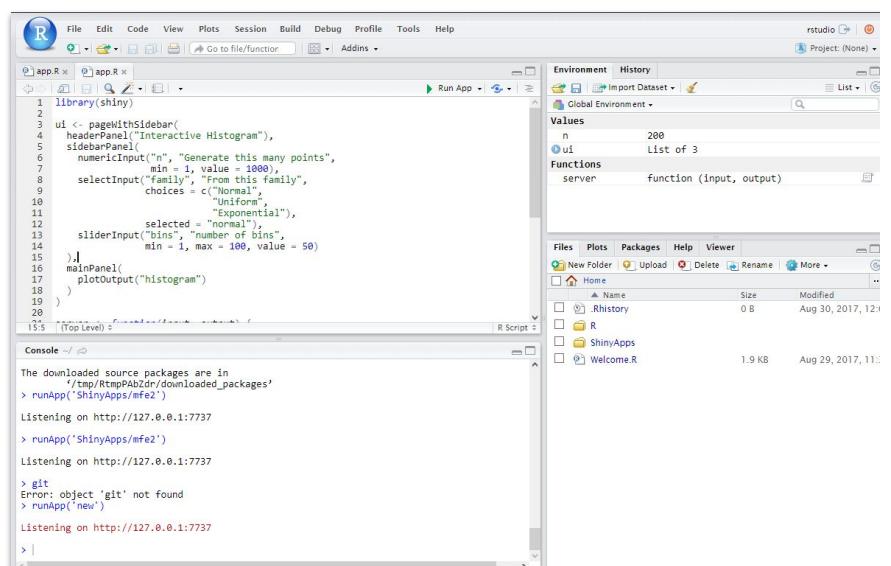
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status
	i-0285531a28754b5...	t2.micro	ap-southeast-1b	terminated	
	i-03625617949cc478b	t2.micro	ap-southeast-1b	terminated	
RStudio	i-038b8f637e85a231	t2.micro	ap-southeast-1b	running	Init
	i-0abf98f00498b693e	t2.micro	ap-southeast-1b	terminated	

A red box highlights the instance with the ID 'i-038b8f637e85a231'. Below the table, a modal window titled 'Sign in to RStudio' is displayed, showing fields for 'Username' (rstudio) and 'Password' (redacted). There is also a 'Stay signed in' checkbox and a 'Sign In' button.

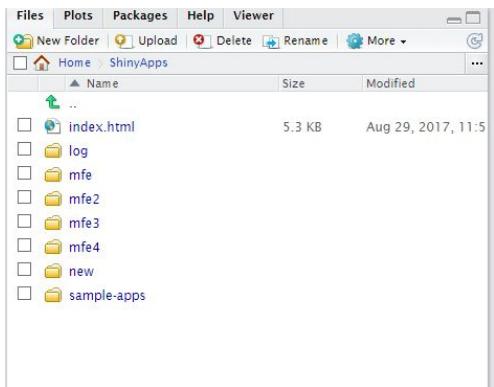
28

RStudio Setup

- Install packages
 - tidyverse
 - tidyquant
 - install.packages
- Tools -> Shell
 - passwd
- Shiny
 - Directory ShinyApps
 - Create sub-directories
- Files
 - Manage upload



Shiny Server



/home/rstudio/ShinyApps/new/

/ShinyApps/new/

<http://ec2-13-229-181-28.ap-southeast-1.compute.amazonaws.com/shiny/rstudio/sample-apps/hello/>

Yang Ye | FE8828 | 2020

30

AWS Recap

- AWS (Amazon Web Services) is an utility company for the internet like electricity/water.
 - We can launch many computing/storage resources as we need.
 - We use Spot instance which is expensive.
 - AWS provides other pricing plan for long-term usage.
 - It's cool to have something running in the cloud. You can show people to impress.
-
- AWS setup is optional in this course.
 - You can run R Studio and application on your laptop
 - If you finishing working on it, leave the instance as Stopped. Be aware of how much you spend on AWS.

Yang Ye | FE8828 | 2020

31

Section 4

Lecture 3: R Markdown and R Shiny/1: layout

Introduction

- **Markdown** is a format that is easy to read and can be converted to other formats, HTML, PDF, Word, Slides.
- R Studio extends it further to create R notebook, interactive document and web application, which is R Markdown.
- **Shiny** is a web programming framework in R. We use it extensively in this course. We begin with the layout part.

Markup and Markdown

- Document stores information.
- Web is a superset of interlinked documents.
- HTML is a markup language, built for machines.
- Markdown is for humans to write doc, with minimal added to decorate it, created by John Gruber in collaboration with Aaron Swartz in 2004.

A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.

- John Gruber



Markdown example

```
---  
title: My first bitcoin  
subtitle: and how I bought a pizza!  
author: "Gru"  
date: "Jul 9, 2010"  
---  
  
# How I bought it  
  
I found someone was selling _10000_ on ebay for __$30__.  
I think that's  
  
- cool  
- fun  
- hacker  
  
# How I used it  
  
I forgot to bring my wallet the other day.  
So I used **the bitcoins** to buy some pizza.  
  
![Pizza](../notes/imgs/bitcoin-pizza.png)
```

Markdown output example

title	subtitle	author	date
My first bitcoin	and how I bought a pizza!	Despicable me	Jul 9, 2010

How I bought it

I found someone was selling 10000 on ebay for \$30. I think that's

- cool
- fun
- hacker

How I used it

I forgot to bring my wallet the other day. I was hungry so I used **the bitcoins** to buy some pizza.



Markdown: Header and Code

Headers

More hashtag, deeper level.

```
# Header1  
## Header2  
### Header3
```

Code

Give four spaces before it

```
if (a > 0) {  
    print(a)  
}
```

Markdown: List

* First paragraph.

Continued.

* Second paragraph. With a code block, which must be indented eight spaces:

```
{ code }
```



- First paragraph. Continued.
- Second paragraph. With a code block, which must be indented eight spaces:

Markdown: Multi-level list

Put four more spaces for each level.

```
* fruits
    + apples
        - macintosh
        - red delicious
    + pears
* vegetables
    + broccoli

● fruits
    ▶ apples
        ★ macintosh
        ★ red delicious
    ▶ pears
● vegetables
    ▶ broccoli
```

Markdown: Ordered List

Put 4 more spaces for each level.

```
#. Chapter 1
  #. Section 1.1
  #. Section 1.2
#. Chapter 2
#. Chapter 3
```



- ➊ Chapter 1
 - ➊ Section 1.1
 - ➋ Section 1.2
- ➋ Chapter 2
- ➌ Chapter 3

Table

Tables	Are	Cool
-----	-----:	----:
col 3 is	right-aligned	\$1600
col 2 is	centered	\$12
zebra stripes	are neat	\$1



Tables	Are	Cool
col 3 is	right-aligned	\$1600
col 2 is	centered	\$12
zebra stripes	are neat	\$1

Markdown: Inline formatting

Emphasis

To emphasize some text, surround it with *'s or _, like this:

This text is emphasized with underscores, and this is **emphasized with asterisks**.

Double * or _ produces strong emphasis:

This is ***strong emphasis*** and with underscores.



This text is *emphasized with underscores*, and this is **emphasized with asterisks**.

Double * or _ produces strong emphasis.

This is **strong emphasis** and **with underscores**. A * or _ character surrounded by spaces, or backslash-escaped, will not trigger emphasis.

Strikethrough

This ~~is deleted text~~

This ~~is deleted text~~.

Markdown: formatting

Superscripts and subscripts

H₂O is a liquid. 2¹⁰ is 1024.

H₂O is a liquid. 2¹⁰ is 1024.

Verbatim, inline code

Use backtick ` (the key to the left of 1)

What is the difference between 'a = 1' and 'a <- 1'?

What is the difference between a = 1 and a <- 1?

Note:

- If the verbatim text includes a backtick, use two backticks.
- Use \\ to turn off \

Links

<http://google.com>



http://google.com

Markdown: Images

A link immediately preceded by a ! will be treated as an image. The link text will be used as the image's alt text:

```
! [Pizza] (imgs/bitcoin-pizza.png)
```



Figure 1: Pizza

Markdown: Formula

MathJax. Use LaTex syntax. There are many online references.

- Inline with text, $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$,
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Or, Centered, $\sum_{i=1}^n X_i$

$$\sum_{i=1}^n X_i$$

Markdown Support in R Studio

- R Markdown is a extension to Markdown that you can execute code among the code.
- It's with suffix **.Rmd**. R Studio has good support for it.
- Reference
 - ▶ R Markdown Cheat Sheet: Help > Cheatsheets > R Markdown Cheat Sheet,
 - ▶ R Markdown Reference Guide: Help > Cheatsheets > R Markdown Reference Guide.
- Create via File > New File > R Markdown.

R Markdown Document example

```
---
```

```
title: "Data Analysis Report"
author: "Yang Ye"
date: "October 23, 2018"
output: html_document
---
```

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## Report
```{r cars}
summary(cars)
```

## Including Plots
```{r pressure, echo=FALSE}
plot(pressure)
```
```

R Markdown Document Output

In the header, you can change the output to other types:

- html_document
- pdf_document
- word_document

After all, press **Ctrl+Shift+K** or Click Knitr

Code block for R Markdown

- Embed code (R, Python, etc.) in R Markdown
- R code can be inline. For example, to generate a random number everytime, include this `r runif(1, 0, 1)`, 0.4369457.
- R code can exist as block, run and show its results. `Calculate_7` is the chunk name. It's optional to give a chunk name. If included, each code chunk needs a distinct name. It's usually best to give each code chunk a name, for easier debug.

Code chunk example:

```
```{r Calculate_7}
a <- 3
b <- 4
print(a + b)
```
```

Result

```
## [1] 7
```

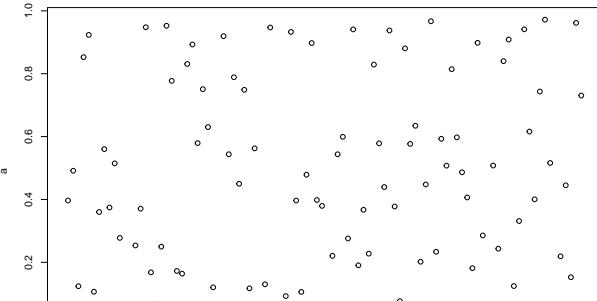
Chunk options

- *echo* is to decide whether to display code, default is FALSE.
- *results* is to decide whether to display result, default is “markup”, set to “hide” to hide.
- *include* is to hide both code and result, default is FALSE.

```
```{r cars, echo = TRUE}
a <- runif(100, 0, 1)
````
```

```
a <- runif(100, 0, 1)
```

```
```{r plot}
plot(a)
````
```



R Markdown: Table

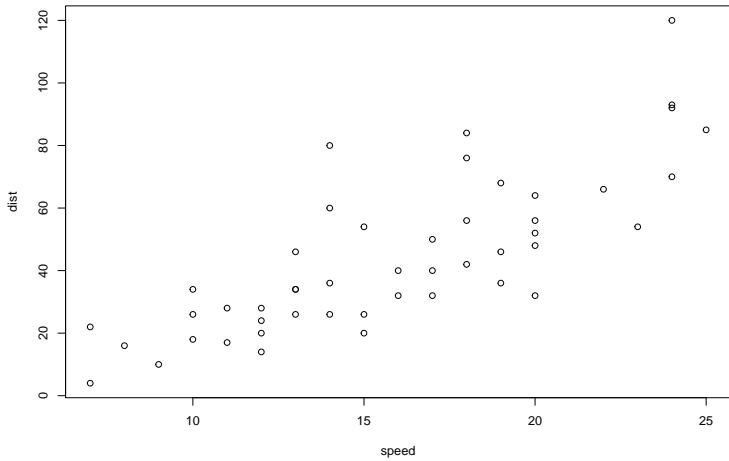
```
```{r kable}
knitr::kable(
 mtcars[1:5,],
 caption = "A knitr kable."
)
````
```

Table 2: A knitr kable.

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |

R Markdown example: Plot

```
```{r plot1, echo = FALSE}
a <- filter(cars, speed > 4)
plot(a)
```
```



R Shiny

- To start, use R Studio.
- File > New File > Shiny Web App...
- Choose single file
- Give a name and folder
- Ctrl+Shift+S or “Run App”

UI First

I removed everything in functions `server` and `ui`. This is the minimal Shiny.
(shiny-1-empty.R)

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Sidebar Layout

Let's add a minimal sidebarLayout (shiny-2-sidebar.R)

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
    fluidPage(sideBarLayout(
        sideBarPanel("This is a panel on the side"),
        mainPanel("This is the main panel")
    )),
    fluidPage(sideBarLayout(
        sideBarPanel("This is a panel on the side"),
        mainPanel("This is the main panel")
    )))
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

fluidPage

- fluidPage means to place the controls from left-right, top-down order.
- fluidPage function can take any number of input parameters.

```
fluidPage(sidebarLayout(  
    sidebarPanel(),  
    mainPanel()  
)
```

Add some tags

- titlePanel("Hello Shiny!"), h1("Introduction to Layout"), h2("Sidebar Layout") (shiny-3-sidebar-min.R)

```
library(shiny)  
  
# Define UI for application that draws a histogram  
ui <- fluidPage(  
  fluidPage(  
    titlePanel("Hello Shiny!"),  
    sidebarLayout(  
      sidebarPanel(  
        h1("Introduction to Layout"),  
        h2("Sidebar Layout")  
      ),  
      mainPanel(  
        img(src = "p19-Hero-Image-796x398.jpg")  
      )  
    )  
  )  
  
# Define server logic required to draw a histogram  
server <- function(input, output) {  
}  
  
# Run the application  
shinyApp(ui = ui, server = server)
```

Sidebar layout with bar on the right

```
fluidPage(  
  sidebarLayout(position = "right",  
    sidebarPanel(),  
    mainPanel()  
  )  
)
```

More tags: Sidebar with more tags (shiny-3-sidebar.R)

```
library(shiny)  
  
# Define UI for application that draws a histogram  
ui <- fluidPage(  
  fluidPage(  
    titlePanel("Hello Shiny!"),  
    sidebarLayout(  
      sidebarPanel(  
        h1("Introduction to Layout"),  
        h2("Sidebar Layout"),  
        a("A link to Google", href="http://www.google.com"),  
        # unordered list  
        tags$ul("About",  
          tags$li("Who are we"),  
          tags$li("What we do"))),  
      # ordered list  
      tags$ol("Steps",  
        tags$li("Write"),  
        tags$li("Run"))),  
    mainPanel(  
      img(src = "p19-Hero-Image-796x398.jpg"))))  
)  
)  
)  
)  
  
# Define server logic required to draw a histogram  
server <- function(input, output) {}  
  
# Run the application  
shinyApp(ui = ui, server = server)
```

Each tag is a function.

```
h1("A header")
p("some text as a paragraph")
a("A link to Google", href="http://www.google.com")
img(src = "p19-Hero-Image-796x398.jpg", width = "100%")
tags$ul("title", tags$li("Item 1"), tags$li("Item 2"))
tags$ol("Step", tags$li("Item 1"), tags$li("Item 2"))
```

Note:

- For image, you need to create a sub-directory `www` together with the R source file. Place the file under it.
- `tags` is a list of functions. To avoid name conflict, I prefer to use `tags$img()`, even `img()` is available to use.

Panels

`titlePanel()` and `wellPanel()` (`shiny-4-wellPanel.R`)

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      h1("Well 1"),
      wellPanel(
        h2("Well 1.1"),
        actionButton("goButton", "Go!")
      ),
      h1("Well 2"),
      wellPanel(
        h2("Well 2.1"),
        actionButton("goButton2", "Go!")
      )
    ),
    mainPanel(
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

}

# Run the application
```

Navlist panel (shiny-5-navPanel.R)

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    navlistPanel(
      "Header A",
      tabPanel("Section 1",
        h1("Section 1"),
        p("This is section 1. First lecture in FE8828."),
      ),
      tabPanel("Section 2",
        h1("Section 2")),
      "Header B",
      tabPanel("Section 3",
        h1("Section 3")),
      "-----",
      tabPanel("Component 5")
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

}

# Run the application
shinyApp(ui = ui, server = server)
```

tabPanel (shiny-6-tabPanel.R)

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(
  fluidPage(
    titlePanel("Hello Shiny!"),
    tabsetPanel(
      tabPanel("Plot", h1("plot")),
      tabPanel("Summary", h1("summary")),
      tabPanel("Image", img(src = "p19-Hero-Image-796x398.jpg"))
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

}

# Run the application
shinyApp(ui = ui, server = server)
```

navBar (shiny-7-navbar.R)

```
library(shiny)

ui <- fluidPage(
  navbarPage(title = "Runchee Technology",
    tabPanel("Product",
      titlePanel("Hello!"),
      "One more thing!",
      p("another paragragh")),
    tabPanel("About us",
      fluidPage(titlePanel("Hello!"),
        "Exordinary people"))

  ,
  navbarMenu(title = "Contact Us",
    tabPanel("Address", "3/4 platform"),
    tabPanel("Phone", "+123.456"))
)
)
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Column-based layout (shiny-8-column.R)

- Caveat: There is fluidRow, but no fluidColumn.
- Column counts always add up to $12 = 4 + 6 + 2$; otherwise, it will appear in the next line.

```
library(shiny)

ui <- fluidPage(
  fluidPage(
    fluidPage(
      titlePanel("Hello Shiny!"),
      fluidRow(
        column(4,
          wellPanel(
            dateInput("date", "How's weather today?")
          )
        ),
        column(6,
          h3("Plot"),
          wellPanel(plotOutput("distPlot"))
        ),
        column(2, h3("Extra"),
          wellPanel(plotOutput("extraPlot"))
        )
      )
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {
}

# Run the application
shinyApp(ui = ui, server = server)
```

Composition layout: Top and Down (shiny-10-composite.R)

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  fluidPage(
    fluidPage(
      title = "Diamonds Explorer",
      fluidRow(
        column(12,
          img(src = "p19-Hero-Image-796x398.jpg", width = "100%")
        )
      ),
      hr(),
      fluidRow(
        column(3,
          h4("Diamonds Explorer"),
          sliderInput('sampleSize', 'Sample Size',
                     min=1, max=nrow(diamonds), value=min(1000, nrow(diamonds)),
                     step=500, round=0),
          br(),
          checkboxInput('jitter', 'Jitter'),
          checkboxInput('smooth', 'Smooth')
        ),
        column(4, offset = 1,
               selectInput('x', 'X', names(diamonds)),
               selectInput('y', 'Y', names(diamonds), names(diamonds)[[2]]),
               selectInput('color', 'Color', c('None', names(diamonds))))
        ),
        column(4,
               selectInput('facet_row', 'Facet Row', c(None='.', names(diamonds))),
               selectInput('facet_col', 'Facet Column', c(None='.', names(diamonds)))
        )
      )
    )
  )
)
```

R Markdown can also contain Shiny (shiny-mfe-example.Rmd)

```
---
title: "MFE FE8828 Assignment 1"
date: 2018-11-03
output: html_document
runtime: shiny
---

```{r setup, include = FALSE}
```

# Use echo = TRUE for assignment is an exception, so code is visible.
```{r, echo = TRUE}
wellPanel("Inputs",
 numericInput("fav_num", "What's your favorite number?", 3))
```
```

Inputs

What's your favorite number?

This is interactive document. ->

Take-home

- Markdown is a simplified way to write formated documents.
- R Markdown can embed code and results. It can mix code and documentation, speed up research progress.
- R Shiny is a web framework. One tag is one function. Easy to compose web page.

Assignment

- Create a website (perhaps a single *.R* file) with Shiny using one of the layouts
 - ▶ You are starting a company to offer.
 - ▶ Decide what you want to do
 - ▶ Create three pages. Name the pages depending on what you want to do.
e.g. Product, About Us and Contact Us
 - ▶ Use different layouts for the pages: sideBar, column-based layout, Navlist.
 - ▶ Be creative!

Demo website with navBar style.

Hello!

One more thing!



The illustration depicts a superhero in a red and blue suit flying over a cityscape at night. The city features buildings with glowing windows and silhouettes of other buildings against a dark sky. The superhero is captured in mid-flight, with one arm extended forward. In the foreground, there are three white rectangular boxes containing text:

- New breakthrough**
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
- To be Released**
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
- Talk to Us!**
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Section 5

Lecture 4: R Programming

R programming

Let's review some R basics and progress to intermediate level. We have following topics:

- Vector/Matrix/String/Date/Time
- Data Frame
- List
- Load/Save
- Anonymous function ...



R: Vector

```
# Create a vector from number
v <- c(1, 3)
v[1] <- 3
v
## [1] 3 3
```

```
# repeat 100 for 10 times.
rep(100, 10)
## [1] 100 100 100 100 100 100 100 100 100 100
```

R: Matrix

```
# create matrix of 3x4
mat <- matrix(2, 3, 4)
mat
##      [,1] [,2] [,3] [,4]
## [1,]     2     2     2     2
## [2,]     2     2     2     2
## [3,]     2     2     2     2
# set first row to 4
mat[1, ] <- 4
# set element (2, 2) to 6
mat[2, 2] <- 6
```

Find element(s) in Vector

- which()
- match()
- %in%

```
data <- 10:1
match(c(1, 3), data)
## [1] 10 8
data[match(c(1, 3), data)]
## [1] 1 3
# Equivalently, ...
which(1 == data | 3 == data)
## [1] 8 10
data[which(1 == data | 3 == data)]
## [1] 3 1
```

Check whether element exists

- FALSE case when element doesn't exist

```
match(c(11, 31), 10:1)
## [1] NA NA
which(11 == 10:1 | 31 == 10:1)
## integer(0)
length(which(11 == 10:1 | 31 == 10:1)) # When length(result) == 0, :
## [1] 0
```

- `any()` and `all()`

```
if (all(c(1, 33) %in% 1:3)) {
  cat("Found all\n")
}

if (any(c(1, 33) %in% 1:3)) {
  cat("Found one/some.\n")
}
## Found one/some.
```

Random

```
# Normal distribution random number
rnorm(3, mean = 10, sd = 3)
## [1] 11.532142 9.143481 7.909362
```

```
# Uniform distribution random number
runif(3)
## [1] 0.6981622 0.2276544 0.3544616
```

```
# Sample
sample(1:10, 10, replace = FALSE)
## [1] 6 7 5 8 4 3 9 2 1 10
# To Be/Not to Be
sample(c(T, F), 10, replace = TRUE)
## [1] FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE
# Throw a dice
sample(1:6, 10, replace = TRUE)
## [1] 3 4 5 4 4 4 5 6 1 4
```

Print

- There is print() but I use most cat(paste0(..., "\n")).
- "\n" is appended to the end to create a line break.
- paste0()/paste() can use to create new strings from any data types.
- paste0() combines any thing without space. paste() uses space, by default.
- paste0()/paste() with collapse helps with vector to print them in one line.
- paste0()/paste() works with all types of data.

```
x <- c(Sys.Date(), Sys.Date(), Sys.Date())
cat(paste0("Current dates is ", x, ".\n"))
## Current dates is 2020-09-17.
## Current dates is 2020-09-17.
## Current dates is 2020-09-17.
cat(paste0("Current dates is ", paste0(x, collapse = " "), ".\n"))
## Current dates is 2020-09-17, 2020-09-17, 2020-09-17.
```

String

```
# sub-string
# substr(x, start, stop)
substr("The fox jumps.", 6, 6 + 5 - 1)
## [1] "ox ju"
```

```
# paste0/paste to concatenate string/convert to string
new_string <- paste0("This is ", "cat")
new_string <- paste0("This is ", "cat", sep = "a")
new_string <- paste0(1:3, sep = "a")
```

```
# toupper/tolower
toupper("big")
## [1] "BIG"
tolower("LOWER")
## [1] "lower"
```

Find/Replace in String

```
# grep: Find, returns T or F
grep("A", "ABC", fixed = TRUE)
## [1] TRUE
grep("D", "ABC", fixed = TRUE)
## [1] FALSE

# sub: replace for one time
# sub(pattern, replace, string,...)
# fixed = TRUE means use fixed string. Not regular expression
sub("D", "ABC", "DDD", fixed = TRUE)
## [1] "ABCDD"
# gsub: replace for all
gsub("D", "ABC", "DDD", fixed = TRUE)
## [1] "ABCABCABC"
```

Find/Replace String with Regular Expression (RE)

If you start to use *regular expression*, sub/grep becomes super powerful. They are default with RE turned on with default value for fixed = FALSE.

```
# If we need to find `Start` appearing the beginning of the string
grep("^Start", "Start with me")
## [1] TRUE
grep("^Start", "me Start")
## [1] FALSE

# To find something in the end
sub("X$", "Z", "XYZ ends with X")
## [1] "XYZ ends with Z"
```

Match with Regular Expression (RE)

```
sub("[^\\_]+\\_.*", "", "USDCNY_M1")
## [1] ""
```

- `[^_]`: Character not containing `_`. Because `_` is a special character, we quote it with two backslashes.
- `+`: One or more
- `.`: Any character
- `*`: none or more.

Extraction with Regular Expression (RE)

```
# Rough cut
sub("(^[^\\_]+)\\_.*", "\\1", "USDCNY_M1")
## [1] "USDCNY"
```

```
# Nice cut
sub("(^[^\\_]+)\\_(.*)", "\\1 \\2", "USDCNY_M1")
## [1] "USDCNY M1"
```

```
# Wonderful cut
sub("(^[^\\_]+)\\_([:alpha:]):([[:digit:]])", "\\1 \\2 \\3", "USDCNY_M1")
## [1] "USDCNY M 1"
```

Regular Expression's Cheatsheet <https://www.rstudio.com/resources/cheatsheets/>

Date

```
# Create date
dt1 <- as.Date("2021-11-03")
dt1
## [1] "2021-11-03"
dt2 <- Sys.Date()
dt2
## [1] "2020-09-17"

library(lubridate)

# Date is such a central role in finance.
# More function about date can be found in package `lubridate`
# Create date with lubridate, a package which provides lots of date functions.
ymd(20210910)
## [1] "2021-09-10"
ymd("20210910")
## [1] "2021-09-10"
```

Date: format code

We can use codes for convert date from/to string.

- %Y/%y: four-digit year/two-digit year
- %m: month in number
- %b/%B: month in abbreviation/full, i.e. Jan/January.
- %d: day

```
format(Sys.Date(), format = "%Y/%m/%d")
## [1] "2020/09/17"

as.Date("2021-11-03", format = "%Y-%m-%d") # %m for number month
## [1] "2021-11-03"
as.Date("2021-Nov-03", format = "%Y-%b-%d") # %b for the 3-letter month
## [1] "2021-11-03"
as.Date("03Nov2021", format = "%d%b%Y")
## [1] "2021-11-03"
```

Other functions from lubridate

```
library(lubridate)
# Change a date
x <- as.Date("2021-10-10")
month(x) <- 1
x
## [1] "2021-01-10"

# Set to the end of the month
day(x) <- days_in_month(x)
```

Business days

Use package `bizdays`

```
# install.packages("bizdays")
library(bizdays)

# 'weekends' is a calendar of weekdays
bizdays("2021-10-01", "2021-10-31", "weekends")
## [1] 20

# add bizdays
add.bizdays("2021-10-01", 5, "weekends")
## [1] "2021-10-08"

# Generate all business days between two dates.
# You will find this useful for financial application.
bizseq("2021-10-01", "2021-10-31", cal = "weekends")
## [1] "2021-10-01" "2021-10-04" "2021-10-05" "2021-10-06" "2021-10-07"
## [6] "2021-10-08" "2021-10-11" "2021-10-12" "2021-10-13" "2021-10-14"
## [11] "2021-10-15" "2021-10-18" "2021-10-19" "2021-10-20" "2021-10-21"
## [16] "2021-10-22" "2021-10-25" "2021-10-26" "2021-10-27" "2021-10-28"
## [21] "2021-10-29"
# bizdays excludes starting day, so one day less than bizseq.
length(bizseq("2021-10-01", "2021-10-31", cal = "weekends"))
## [1] 21
```

Calendar

If not provided, `start.date` is by default the first holiday and `end.date` is the last holiday, so we provide them here.

```
# Create a holiday calendar for this mini term.
create.calendar(name="MFE_Mini_2", holidays = c(as.Date("2021-10-28")),
                 start.date = as.Date("2021-09-10"), end.date = as.Date("2021-10-31"),
                 weekdays = c("saturday", "sunday"))

bizdays("2021-10-01", "2021-10-31", cal = "weekends")
## [1] 20
# One day less
bizdays("2021-10-01", "2021-10-31", cal = "MFE_Mini_2")
## [1] 19
```

Time

Convert time to character/string

- %H: hour
- %M: minute
- %S: second

```
format(Sys.time(), format = "%H%M")
## [1] "0147"
format(Sys.time(), format = "%H:%M:%S")
## [1] "01:47:05"
format(Sys.time(), format = "%H:%M:%S")
## [1] "01:47:05"
library(lubridate)
ymd_hms("2021-12-31 12:59:59")
## [1] "2021-12-31 12:59:59 UTC"
```

Time

Change time, lubridate provides hour, minute

```
x <- Sys.time()
x
## [1] "2020-09-17 01:47:05 +08"
hour(x) <- 12
x
## [1] "2020-09-17 12:47:05 +08"
minute(x) <- 3
x
## [1] "2020-09-17 12:03:05 +08"
minute(x) <- 123 # what will happen?
x
## [1] "2020-09-17 14:03:05 +08"
```

List - Basic

```
# Create a list with list() function
# Nameless list
# list[_n_] => item by order
a <- list(3, 4)
a[[1]]
## [1] 3
a[[2]]
## [1] 4

# Named list, you can use $ and [ operators
# list[]: gives back a value
# list$name => list[["name"]]
a <- list(a = 3, b = 4)
a[[1]]
## [1] 3
a[[2]]
## [1] 4
a[["a"]]
## [1] 3
a$a
## [1] 3
```

List - Create

```
# When you want to use a number as key, use backtick
list_of_strikes <- list()
list_of_strikes$`65` <- 3
list_of_strikes$`60` <- 4

# if a name doesn't exist in the list
a$c
## NULL
# Use `is.null()` to check
if (is.null(a$c)) {
  cat("c doesn't exist in list a\n")
}
## c doesn't exist in list a
```

List - Create

```
ll <- list(elem = 1, c1 = "a", c2 = "b")

# access the list
ll[[1]]
ll$elem

# add new member to the list
ll$new_elem <- 3
# update member in the list
ll$c1 <- 3

# set c1 to NULL would delete c1 from the list
ll$c1 <- NULL

ll
## $elem
## [1] 1
##
## $c2
## [1] "b"
##
## $new_elem
## [1] 3
```

List - Usage 1

```
# List - Basic
# Map
basket <- sample(c("Apple", "Orange", "Pear"), 100, replace = TRUE)
fruit_count <- list()
for (b in basket) {
  if (is.null(fruit_count[[b]])) {
    fruit_count[[b]] <- 1
  } else {
    fruit_count[[b]] <- fruit_count[[b]] + 1
  }
}
fruit_count
## $Pear
## [1] 37
##
## $Orange
## [1] 32
##
## $Apple
## [1] 31
```

List - Usage 2

```
# Let's write a generic function to do this
add_to_map <- function(map, key, value) {
  if (is.null(map[[key]])) {
    map[[key]] <- value
  } else {
    map[[key]] <- map[[key]] + value
  }
  map
}

# You may copy function add_to_map to every file that you want to use this kind of d
fruit_count <- add_to_map(fruit_count, "Pomelo", 12)
fruit_count
## $Pear
## [1] 37
##
## $Orange
## [1] 32
##
## $Apple
## [1] 31
##
## $Pomelo
## [1] 12
```

List - Usage 3

```
# Use case 1: Use list to pass data in or out.  
# pass in  
do_lots_of_work <- function(lst) {  
  lst$a + lst$b  
}  
# pass out  
ret_lots_of_work <- function() {  
  return(list(a = a, b = b))  
}  
  
res <- ret_lots_of_work()  
res$a  
## $a  
## [1] 3  
##  
## $b  
## [1] 4  
res$b  
## [1] "Pear"
```

List - Usage 4

```
# Case 2: configuration  
app_config <- list(MAX = 10, MIN = 10, DISPLAY_RESULT = TRUE)  
  
do_lots_of_work <- function(app_config) {  
  app_config$MAX  
}
```

R: data frame

Common functions for data frame

```
View()  
head()  
tail()  
str()  
nrow()  
ncol()  
dim() # returns both nrow and ncol  
colnames() / rownames()
```

R: data frame

The basic structure of a data frame:

- There is one observation per row and
- Each column represents a variable, a measure, feature, or characteristic of that observation.
- In summary, **2D table**

```
df <- tibble(  
  date = seq(as.Date("2021-01-01"), as.Date("2021-01-10"), by = "day"),  
  stock = replicate(10, paste0(sample(LETTERS, 3, replace = TRUE), collapse = "")),  
  quantity = round(runif(10) * 10000 ,0))  
# df["date"]: gives a data frame  
# df[["date"]]: gives value  
# df$date: same as [[{"date"}]]  
  
# Get three rows  
df[c(3, 6, 9), , drop = F]
```

Data frame extraction with drop = TRUE or drop = FALSE

```
# Get three columns
df[, 1, drop = FALSE]
# This would return a vector
df[, 1, drop = TRUE]

# Use column names
df[, c("date", "quantity"), drop = FALSE]
```

Functions

Input parameters

```
func1 <- function() { }

func2 <- function(input1, input2) { }

# Param input1 is default to 1
func3 <- function(input1 = 1, input2) { }

func4 <- function(input1, input_type = c("int", "char")) {
  # This would check whether input_type is set to one of the pre-set values.
  input_type = match.arg(input_type)
}

func5 <- function(in1, in2) {
  if (in1 < 0) {
    return(0)
  } else {
    return(in1 + in2)
  }
}
```

Functions

```
# The last value before function finishes will be returned automatically.  
# No need to use return.  
func5 <- function(in1, in2) {  
  if (in1 < 0) {  
    0          # return this  
  } else {  
    in1 + in2 # or, return this  
  }  
}  
  
# Unless there is extra steps before  
func6 <- function(in1, in2) {  
  if (in1 < 0) {  
    return(0) # explicit-force return.  
  } else {  
    res <- in1 + in2  
  }  
  
  res <- res * 3  
  res # res is the last value to be returned.  
}
```

Anonymous Function

```
# Function that's defined in-place, which doesn't need to have a name.  
(function(x) { print(x) })(3)  
## [1] 3  
# if there is only one line, you can skip {}  
(function(x) print(x))(3)  
## [1] 3  
  
# For longer functions, you can make it multi-lines.  
(function(x) {  
  if (x > 3) {  
    print(x)  
  } else {  
    print(x - 3)  
  }  
})(3)  
## [1] 0
```

purrr::map and sapply() Function - 1

```
library(purrr) # if not installed, run this line: install.packages("purrr").  
  
# These two are equivalent.  
res1 <- purrr::map(1:10, function(x) { rnorm(x, n = 10) })  
# function(x) func(x) can be simplified as func.  
res2 <- purrr::map(1:10, rnorm, n = 10)  
head(res1, n = 1)  
## [[1]]  
## [1] 1.824955643 1.384339487 1.028139356 1.739618010 0.135943228  
## [6] 2.150535246 -0.018378860 1.732545165 1.166419950 0.009825557  
# purrrr::map returns a list()
```

purrr::map and sapply() Function - 2

```
# This is what we really want to do.  
# Generate ten normal distribution and get their mean.  
# rnorm(n, mean = 0, sd = 1). Where does the input go to?  
res <- purrr::map(1:10, rnorm, n = 1000)  
map_dbl(res, mean)  
## [1] 1.019905 2.048421 3.012500 3.967535 5.031277 5.985041 6.986432  
## [8] 7.981051 8.990855 10.091227  
  
# sapply() achieves the same as purrrr::map(), a bit slower.  
# Package purrr succeeds original R base.  
sapply(1:10, function(x) x ^ 2 )  
## [1] 1 4 9 16 25 36 49 64 81 100  
sapply(1:10, function(x) `^` (x, 2) )  
## [1] 1 4 9 16 25 36 49 64 81 100  
sapply(1:10, function(x) `^` (2, x) )  
## [1] 2 4 8 16 32 64 128 256 512 1024
```

Exercise

Write functions to do

- Determine leap year?
- Print the list of month names in abbreviation or full
- Write a function to count how many working days in a year, given 1) the year 2) list of holidays?