FE8828 Programming Web Applications in Finance

- Session 1 -

What's Internet? What's Web?

Launch into the Cloud

R Markdown and Shiny layout

Dr. Yang Ye yy@runchee.com

Sep 10, 2019

Where does this course mean?

- Programming
- Data science
- Finance
- In the world of Internet

Programming

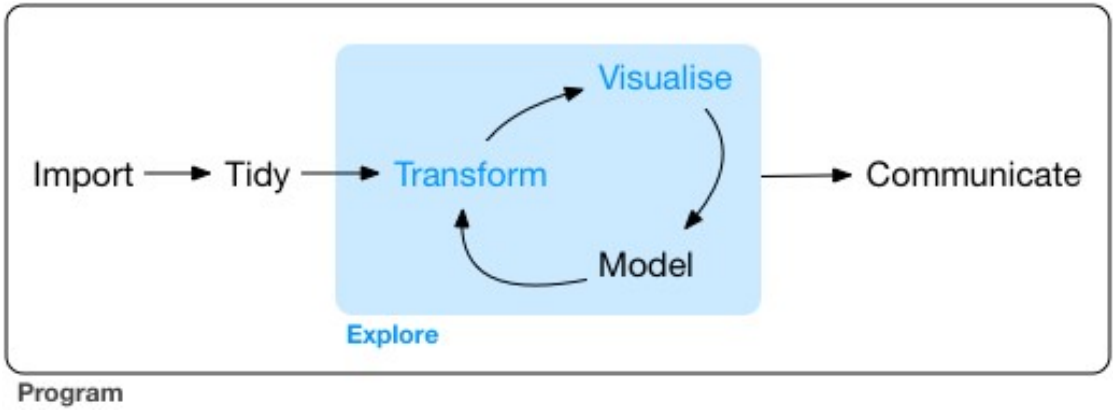
Theory <-> Programming <-> Practice

- Programming is pratice of theory.
- A well-designed software framework is a guide to pratices.
- The reason to learn programming is bi-folded:
  - *We can code our own thoughts, and*
  - *We can utilize other people's work.*
- We will build on beginner R to the imtermeidate/advanced R
- You will have ample exercise in this course - for your own learning.

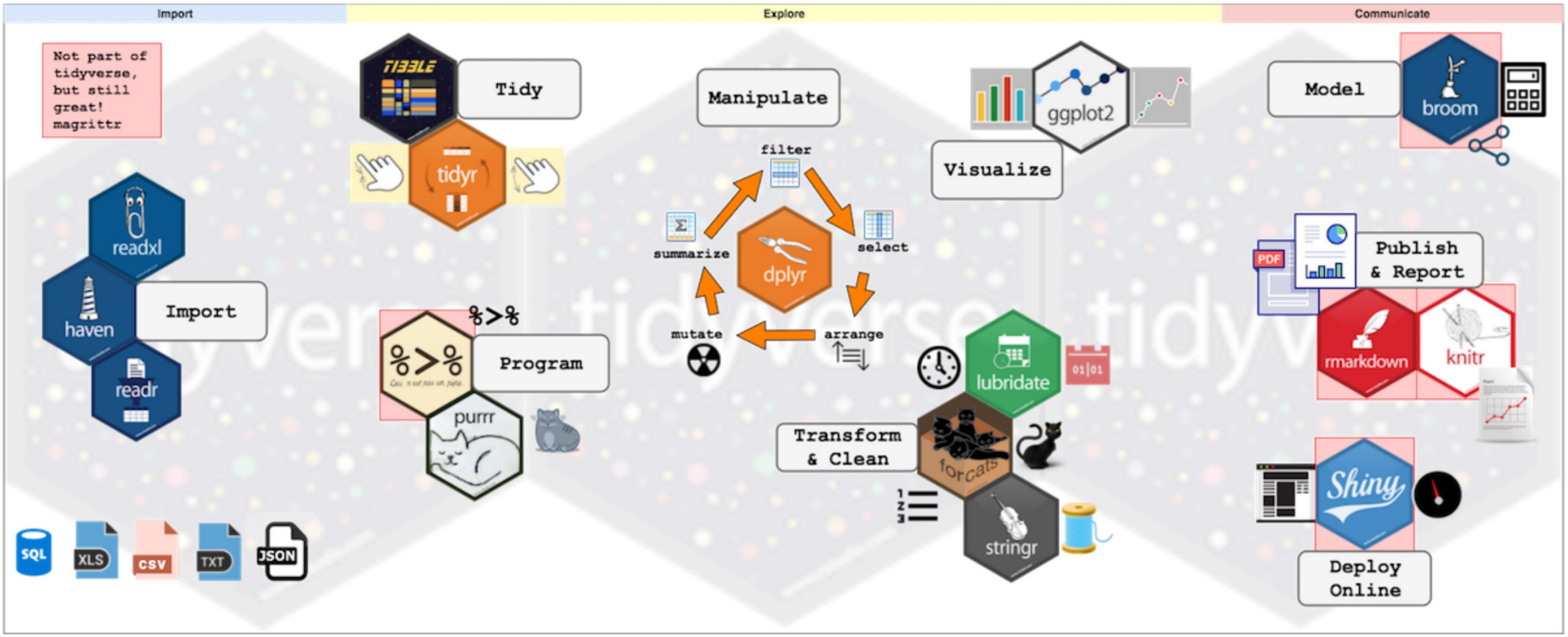
Data Science workflow

Data -> Model -> Application

- It's about "Data Science" workflow
- We get data from Web and also publish new data to web.



Data Science workflow

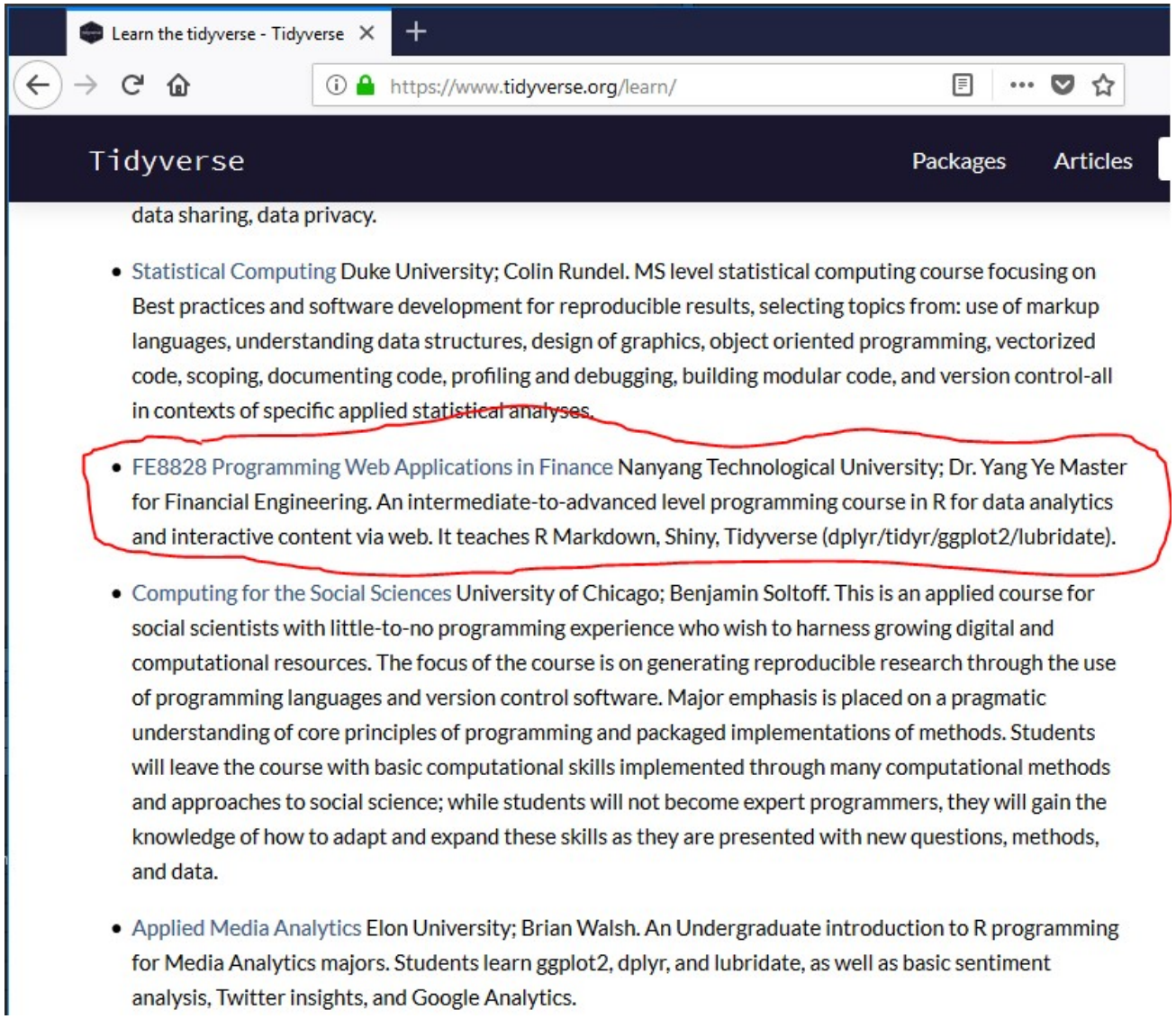


Finance

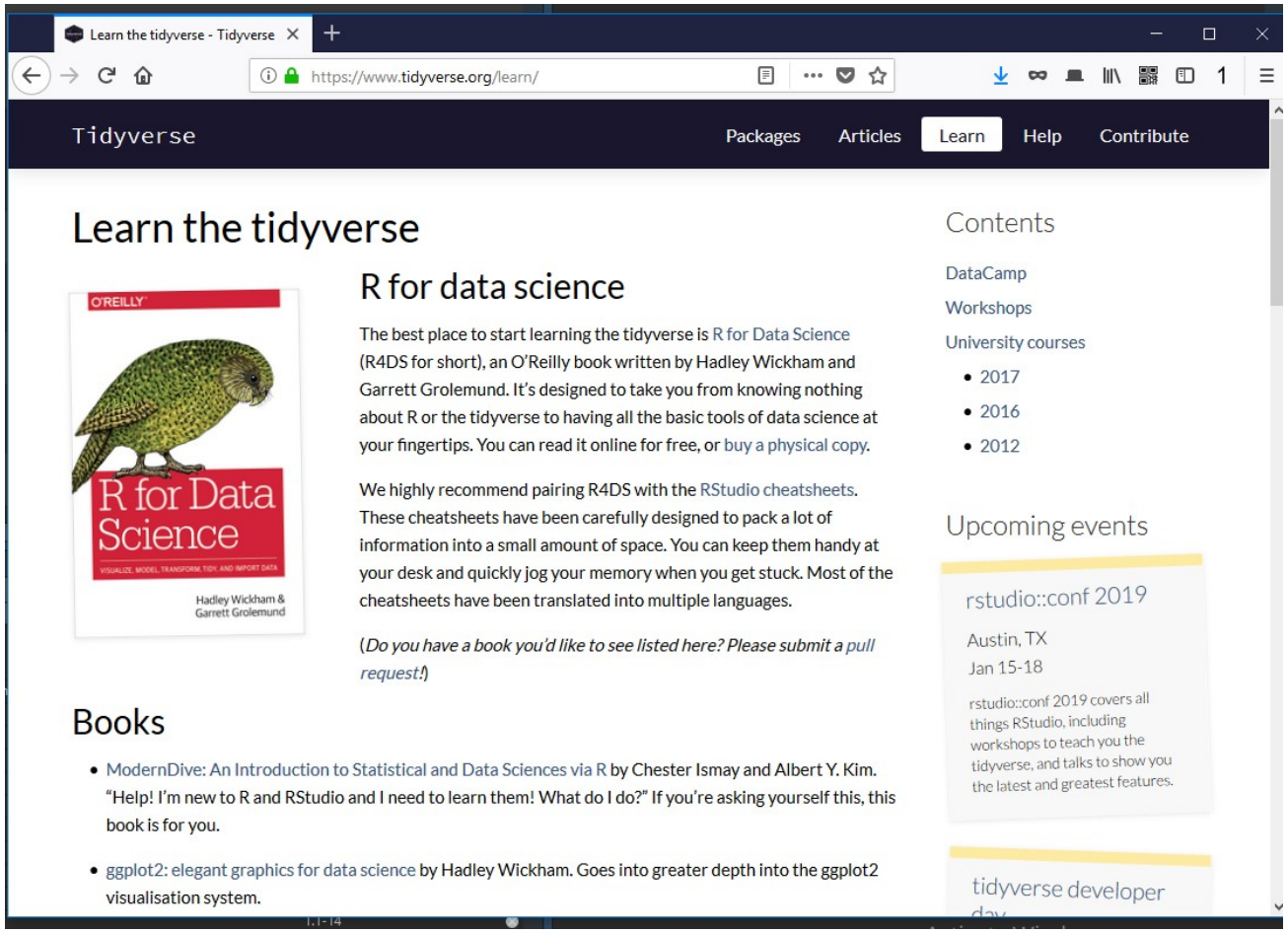
Data -> Model/Valuation Engine/Trading Engine -> Appplication

- Finance application, trading strategy and derivatives valuation.
- Tidyverse with a finance flavour

Our course


Learn the tidyverse - Tidyverse
https://www.tidyverse.org/learn/
Tidyverse
Packages
Articles
data sharing, data privacy.
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.

Reference book



Objective

1. Know the way of Internet: the network, the cloud and the application.
2. Use data manipulation and data visualization to do exploratory data analysis.
3. To do option valuation, and trading strategy performance analysis.
4. Build real-world data-driven reports and dashboard, data visualization and ~predictive model~.
5. Latest technology in cryptocurrency and payment system based on Bitcoin and Blockchain.

What does it take?

- Programming is our tool
  - *R is a system that has been designed to process data.*
  - *Intermidiate-to-Advanced level R*
  - *Use R in other MFE courses*
  - *A complete suite for data science.*
- Pick up a habit of good analyst: Use reproducible research, use notebook.
- Take a mind of data exploration
- Take a mind of analysis: answer is not fixed but open-ended. You need to draw conclusion and make suggestion.
- Take a mind of strategy thinking

Course Outline: Session 1-3

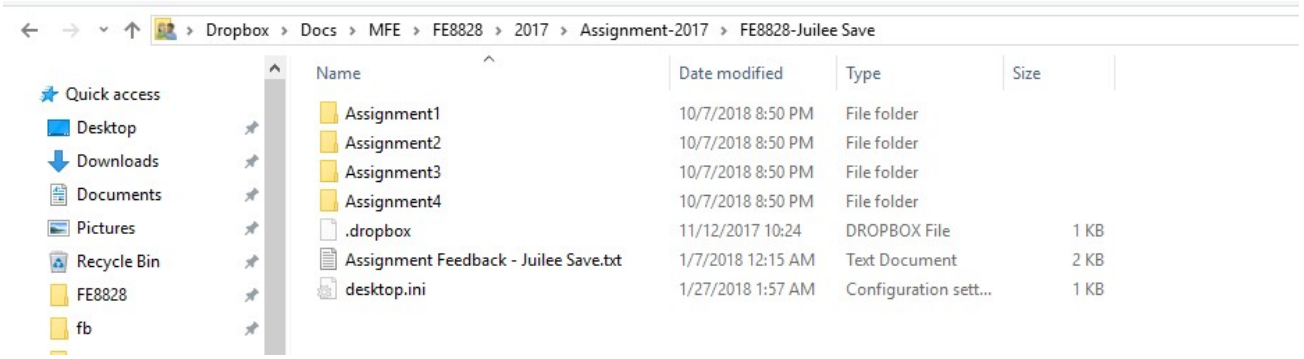
- Session 1:
  - *What's Internet? What's Web?*
  - *Launch into the Cloud: AWS*
  - *R Markdown and Shiny/1: layout*
- Session 2:
  - *Intermediate R Programming*
  - *Shiny/2: R Web Framework*
  - *Data Manipulation and EDA/1*
- Session 3:
  - *Data Manipulation and EDA22*

Course Outline: Week 4-6

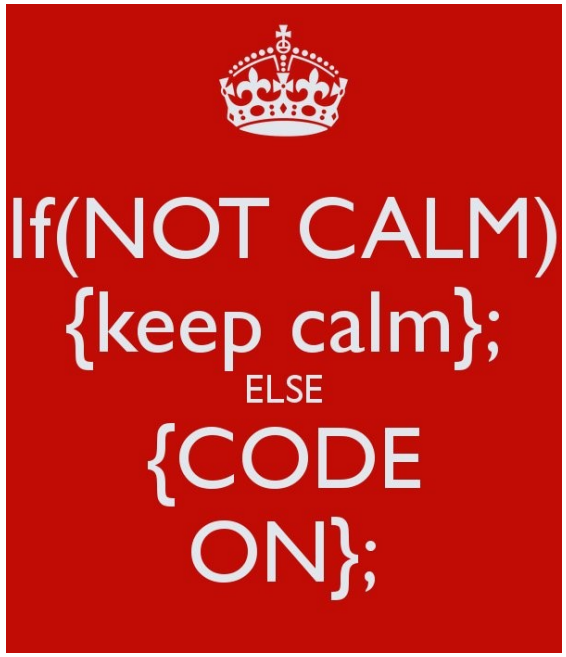
- Session 4
  - *Data Visualization and EDA*
  - *Shiny/3: Advanced*
- Session 5
  - *Building Financial Applications*
  - *Building Predictive Model*
- Session 6:
  - *Further topics: Blockchain*

Assingment

- Assignment 1: A static website: a front page, an about page and some description pages. (due by week 2)
- Assignment 2: Shiny web application. (due by week 3)
- Assignment 3: Data analysis (due by week 4)
- Assignment 4: Data visualization (due by week 5)
- Assignment 5: Group project (due the Sunday after exam, I need to submit scores by Monday)
- Submission
  - *Name your directory as FE8828-Your Name. Share the directory with leafyoung@gmail.com on Dropbox or Google drive.*
  - *Organize your assignments into directories, e.g. \Assignment 1, \Assignment 2, …*



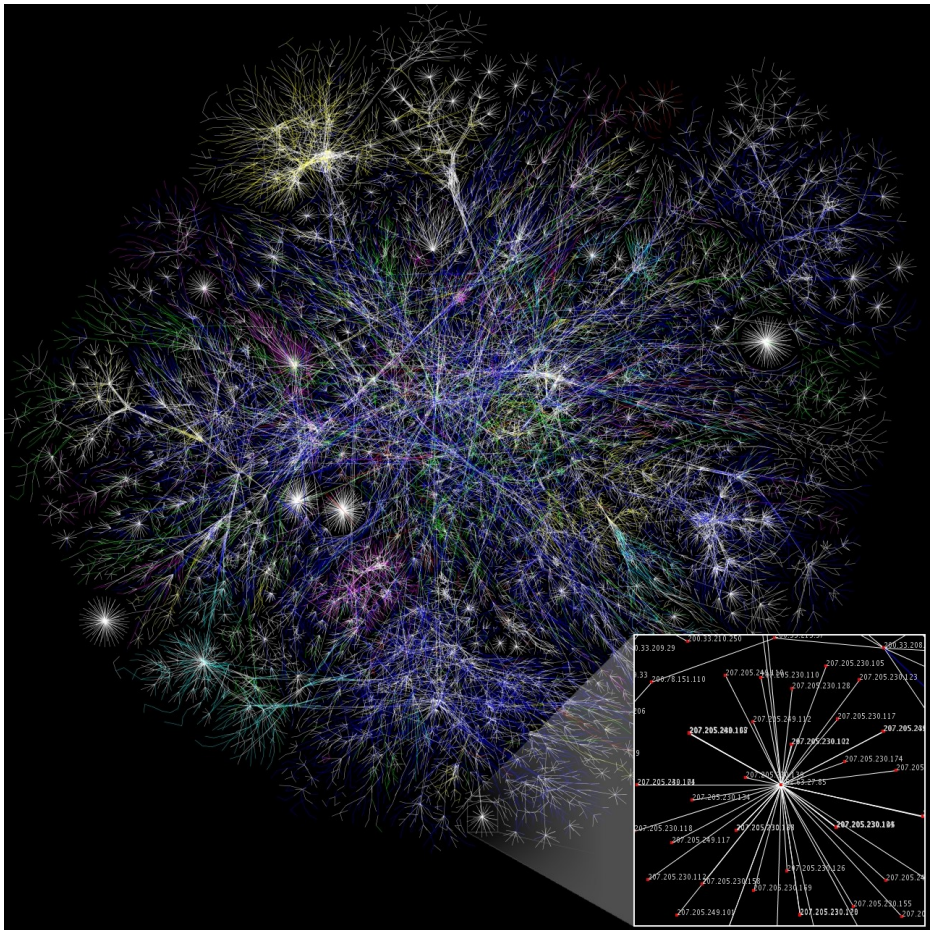
Keep calm and code on



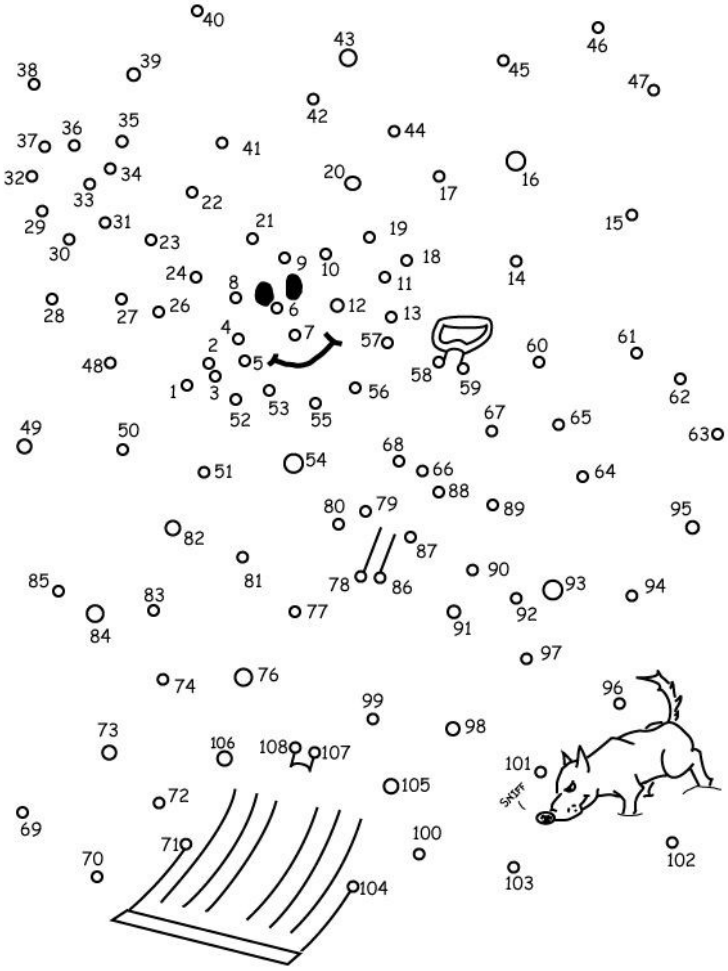
Lecture 1: What's Internet? What's Web?

- Network
- Internet
- HTTP/HTML/Web

Network

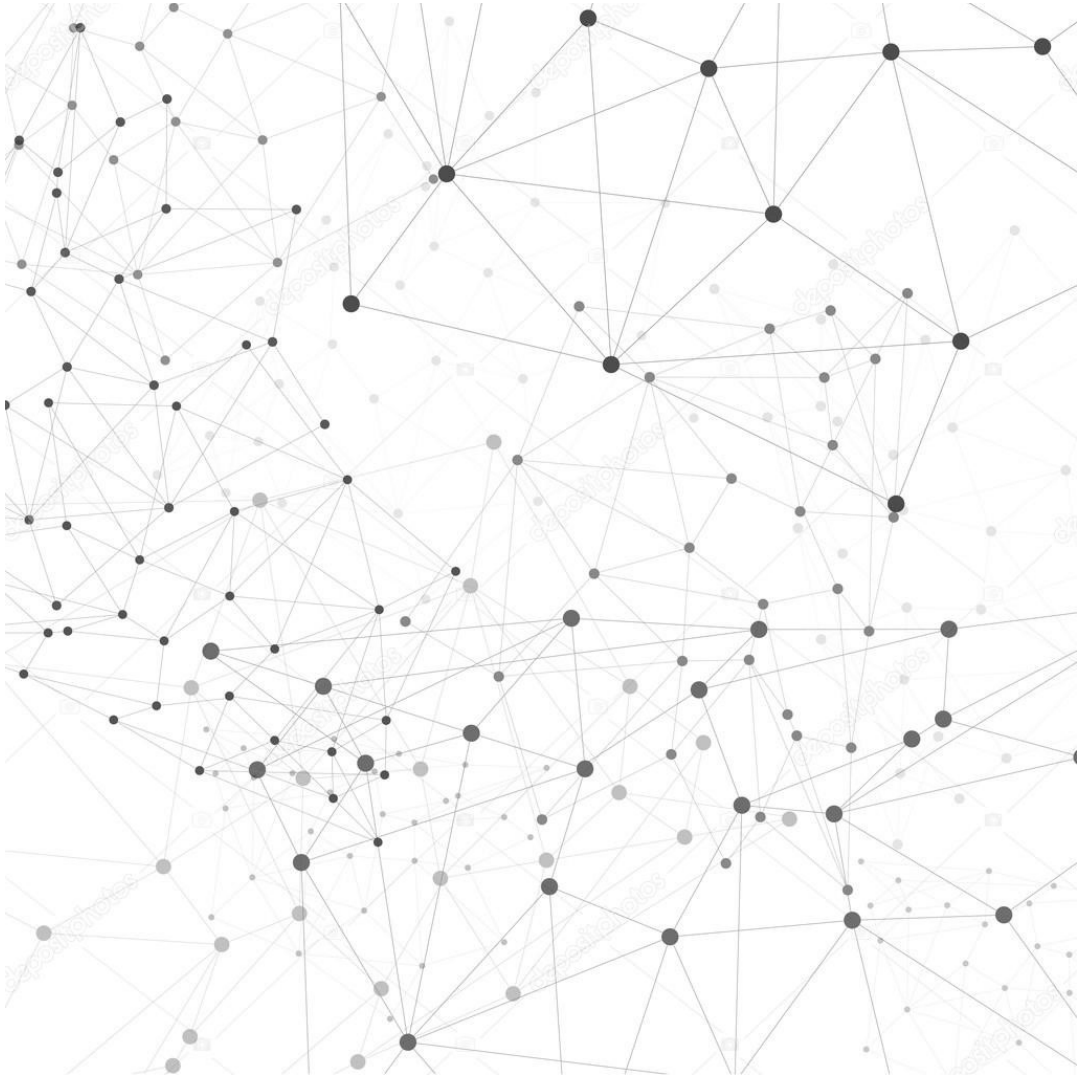


Network is to connect the dots



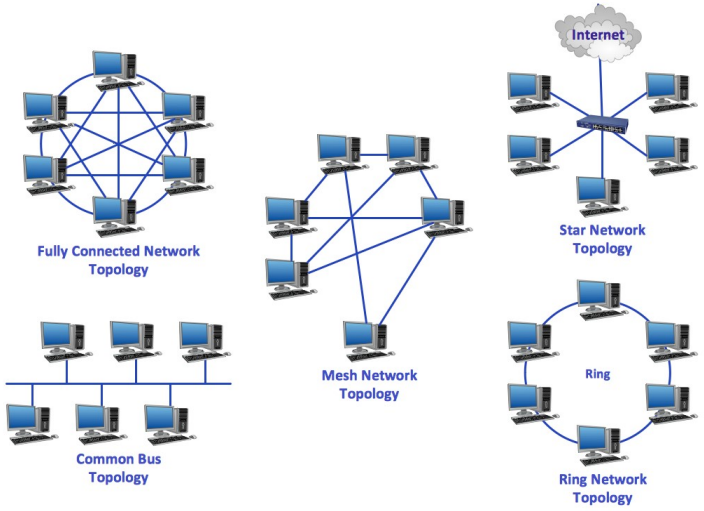
Directions (you'll need them) are available at www.ethanham.com/blog/dots/directions.pdf

Network is to connect the devices



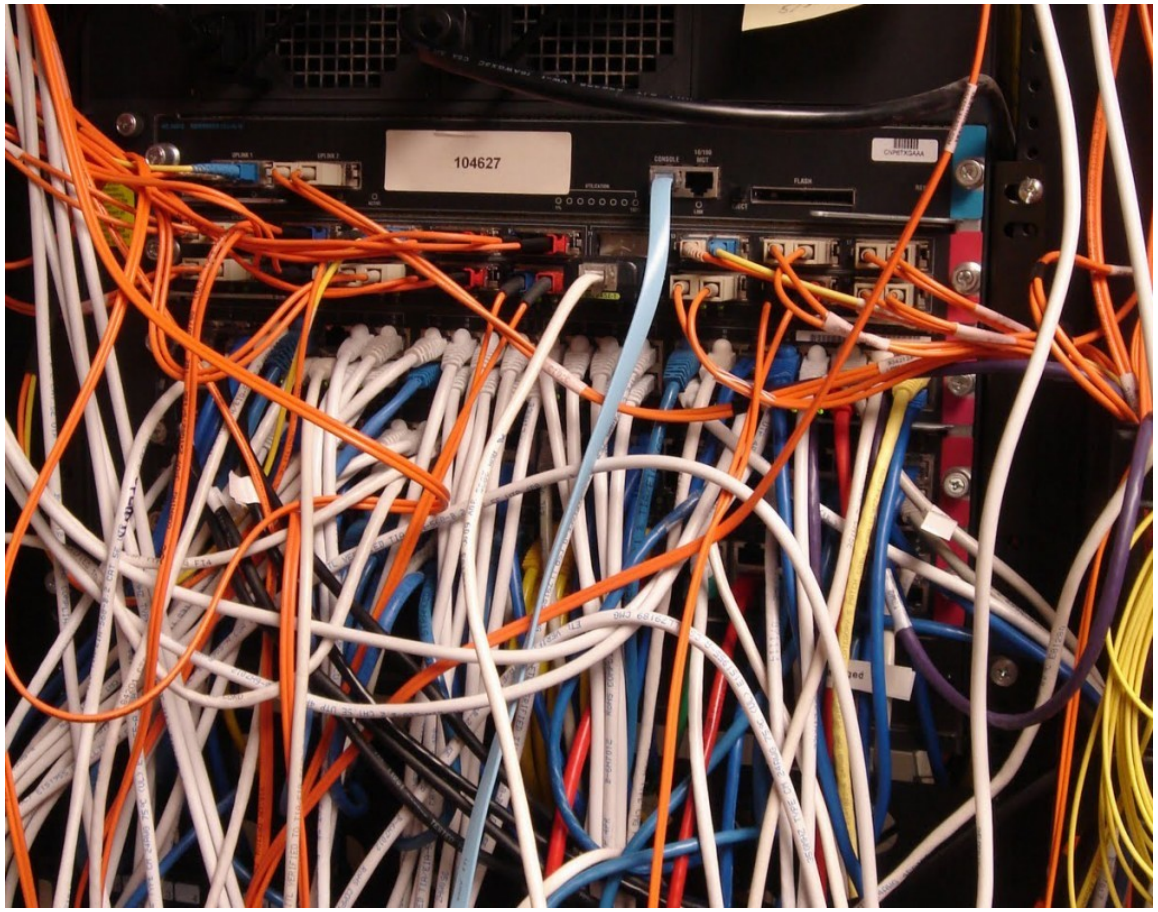
Building network

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



Network talks

What runs inside the cables?

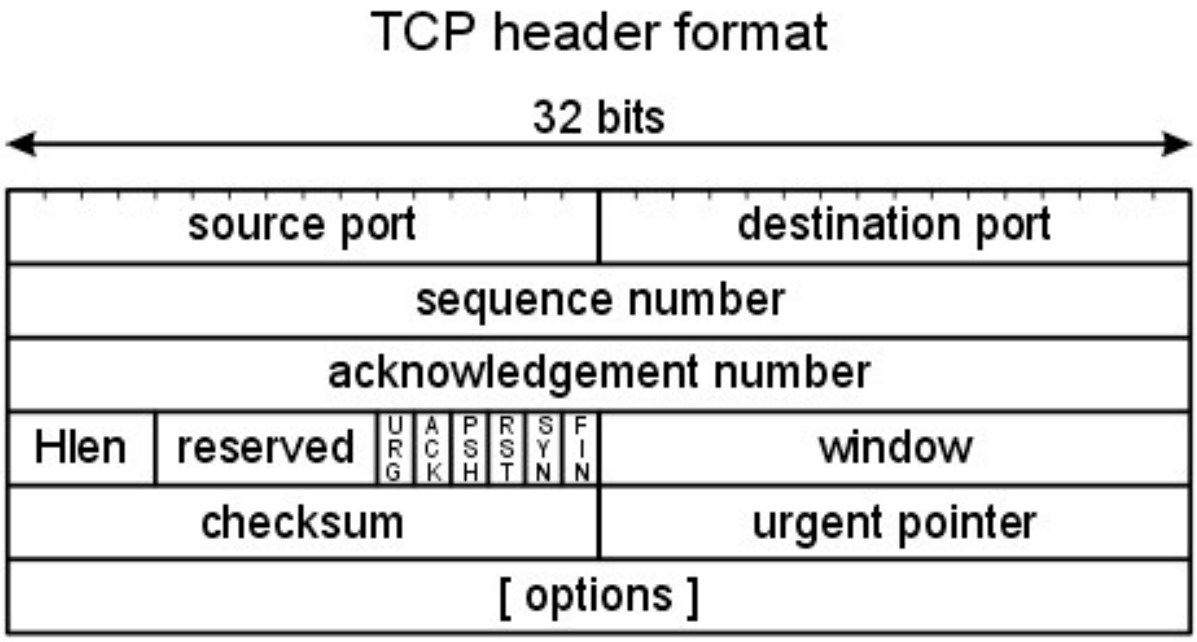


Network talks with Network Protocol

1. Information turns to *packet* according to protocol speicification
2. Protocol also specifies the process
3. Infrastructure is to route the *packets* to the destination.

Packet

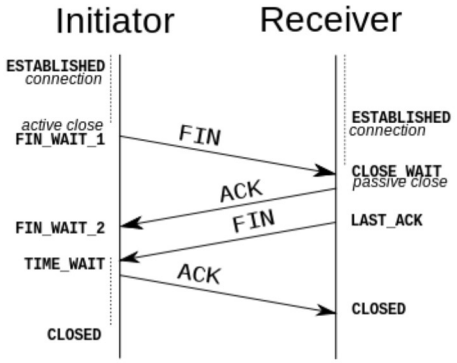
1. Information turns to packet



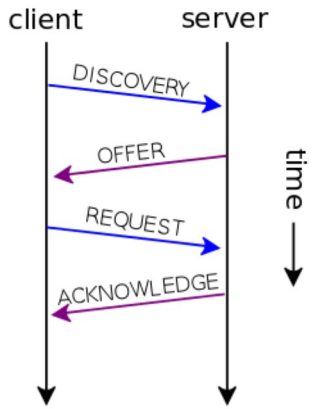
Protocol

2. Protocol designs the packet and process

TCP Session

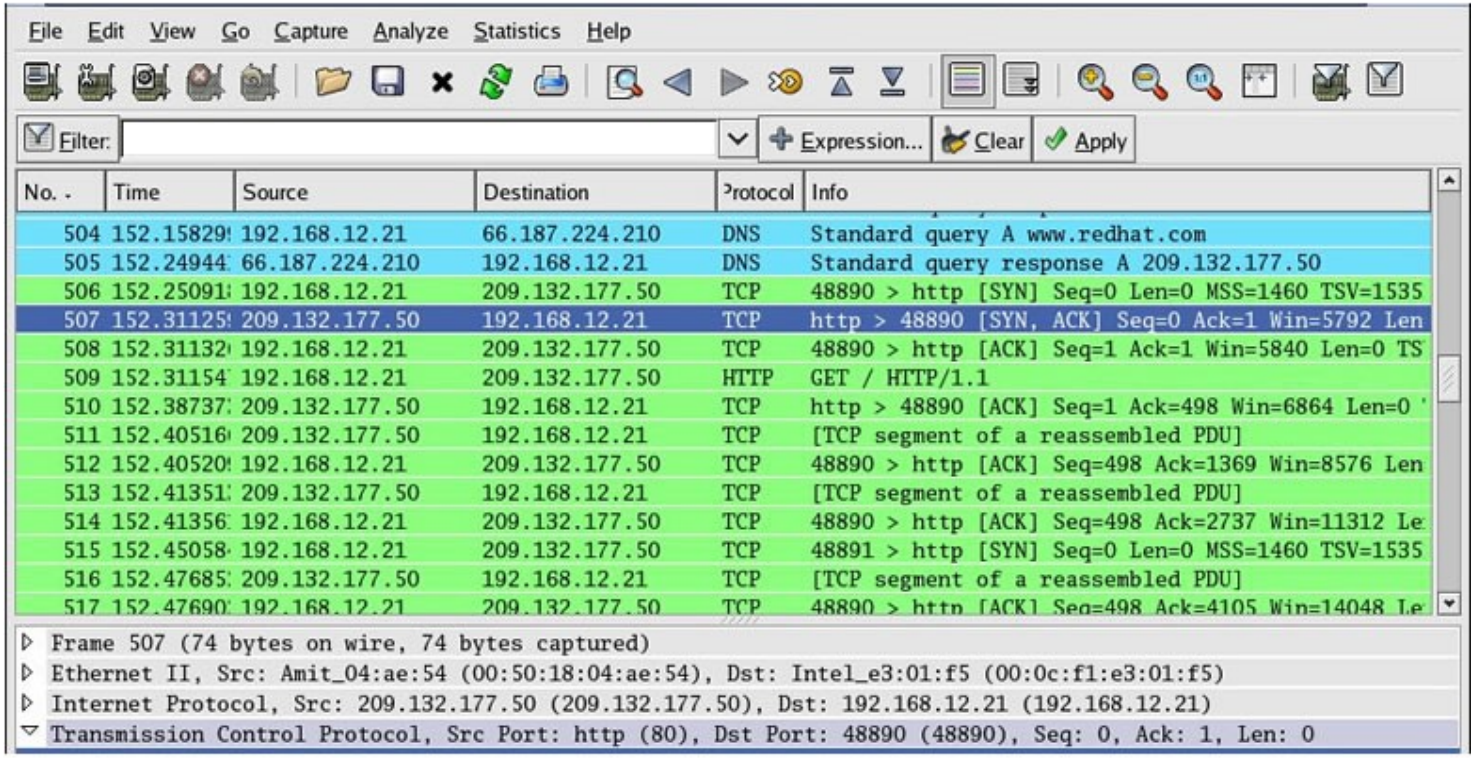


DHCP Session



Routing/Gateway

3. Infrastructure helps to transmit and route the packets to the destination.



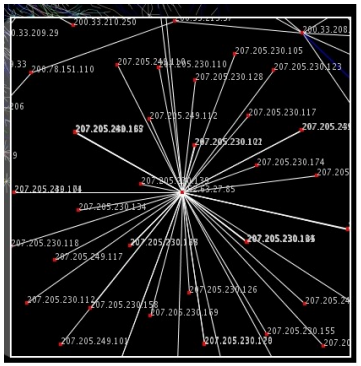
TCP/IP and Internet

- The Defense Advanced Research Projects Agency (DARPA) created the TCP/IP model in the 1970s to build ARPANET.
- ARPANET is a wide area network that preceded the internet.

What does TCP/IP gives?

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

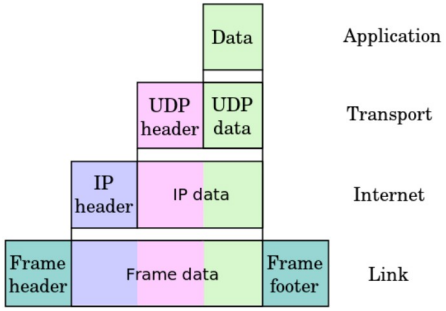
1. IP (Internet Protocol)
    - *IP address.*



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

- Four Layers



- Application layer runs many protocols

Internet protocol suite

Application layer
BGP · DHCP · DNS · FTP · HTTP · IMAP · LDAP · MGCP · NNTP · NTP · POP · ONC/RPC · RTP · RTSP · RIP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP · *more...*

Transport layer
TCP · UDP · DCCP · SCTP · RSVP · *more...*

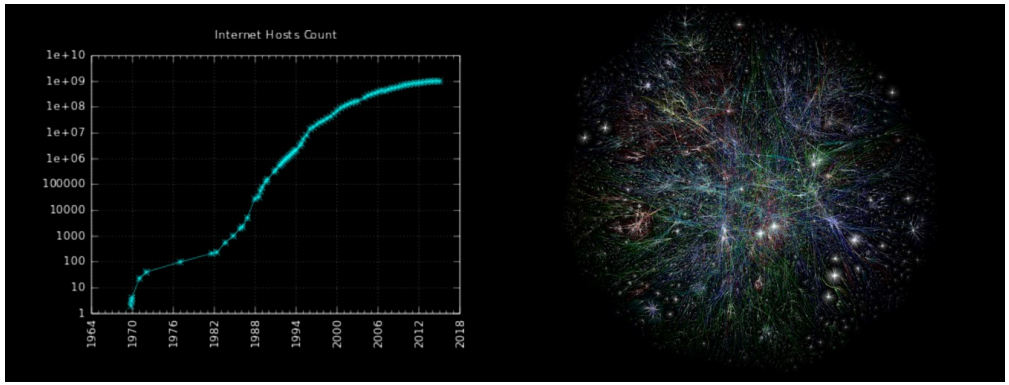
Internet layer
IP (IPv4 · IPv6) · ICMP · ICMPv6 · ECN · IGMP · IPsec · *more...*

Link layer
ARP · NDP · OSPF · Tunnels (L2TP) · PPP · MAC (Ethernet · DSL · ISDN · FDDI) · *more...*

V · T · E

This is why and how Internet growed in size

A well-defined network protocles.



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

What happens after ~~plugging cable~~, turning on Wi-Fi?

- We have talked about what runs inside the cable?
- Every network device has a hardware address
- Dynamic Host Configuration Protocol (DHCP) protocol:
  - *DHCP client, a component of the operating system sends out a DHCP request and recieves an offer from a DHCP server running on the router.*
- Device accepts the IP address and uses it to label for itself. Router also knows where to send the packet.

When it wants to visit someone on the network? 1/2

- Domain Name System:
  - *We don't use 123.456.789.012 but www.google.com.*
  - *DNS is the directory service for internet.*
  - *Your device also receives One or more DNS server addresses so the computer knows where to send DNS requests.*
  - *DNS server would return the actual IP address of the domain name.*
  - *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? 2/2

- Then, your device creates and send the packet “request”. Wait for response.
- Router and gateway will relay the packets to the receiver.

HTTP/HTML

- Now we shall have an idea of how Internet works, let's move on to Web.
- Initiated by Tim Berners-Lee at CERN (where big collison happens in "The Large Hadron Collider") in 1989.
- HTML was also invented by Tim, "HTML tags"

HTTP Request/Response

- Request

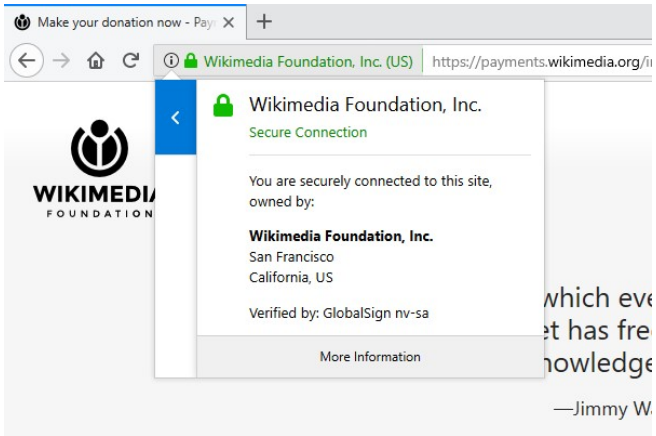
```
GET /index.html HTTP/1.1
Host: www.example.com
```

- 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/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 crytography 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">
    <img alt="" src="//upload.wikimedia.org/wikipedia/commons/thumb/6/65/Web_Index.svg/300px-Web_Index.s
        width="300" height="154" class="thumbimage"  />
    </a>
</div>
```

Web Browser

First generation

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 App and cross-platform.*

To recap

Why the Internet has succeeded?

- Information flows by packet.
  - *IP protocol sets the address for the device*
  - *TCP protocol transmits the packet reliably*
  - *UDP protocol does real-time transmission which can accept failure.*
- Local device just needs to send the packet
- Nework 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.

To recap

Web application

- Browser => HTML/CSS/JavaScript => HTTP => TCP => IP => Network physical lines
- Network physical lines => IP => TCP => HTTP => HTML/CSS/JS => Browser
- We will write in R Shiny, which subsequently output HTML/CSS/JavaScript to be run in browser.

Lecture 2: Amazon Web Services: Launch into the Cloud

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

Lecture 3: R Markdown and 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)
    }
```

```
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 lists

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 Lists

Put 4 more spaces for each level.

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

1. Chapter 1
    1. *Section 1.1*
    2. *Section 1.2*
2. Chapter 2
3. 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.

Markdown: Inline formatting

Strikthrough

This ~~is deleted text.~~ This ~~is deleted text.~~

Superscripts and subscripts

H~2~O is a liquid. 2^10^ is 1024. H_2O is a liquid. 2^{10} is 1024.

Verbatim. inline code

Use backtick `. What is the difference between `>>=` and `>>`? What is the difference between >>= and >>?

Note:

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

Markdown: Links

```
<http://google.com>
```

http://google.com

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)
```

Pizza

Formula

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

Inline with text

```
$x = {-b \pm \sqrt{b^2-4ac} \over 2a}$
```

$x = {-b \pm \sqrt{b^2-4ac} \over 2a}$

Centered

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

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

R Markdown

Reference in R Studio

- R Markdown Cheat Sheet: Help > Cheatsheets > R Markdown Cheat Sheet,
- R Markdown Reference Guide: Help > Cheatsheets > R Markdown Reference Guide.

Create it via File > New File > R Markdown.

- Document
- Presentation
- Shiny

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
- Ctrl+Shift+K or "Knitr"

Code block for R Markdown

- R Markdown is a extension to Markdown that you can execute code among the code. If you name the file as **.Rmd** and *knit* in R Studio.

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

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

- 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.

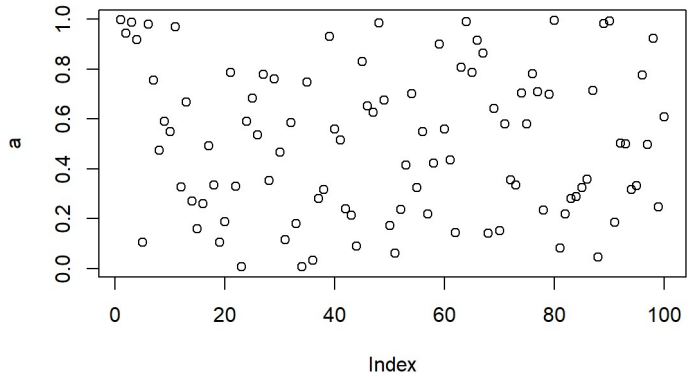
- R code can also be inline. For example, to generate a random number everytime, include this `runif(1, 0, 1)`, 0.6581207.

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)
```

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



R Markdown example: Table

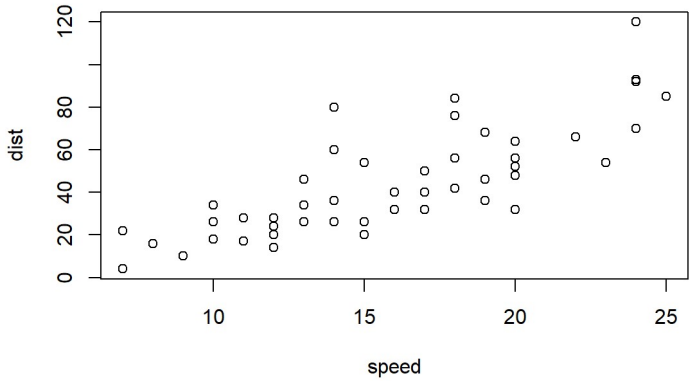
```
```{r kable}
knitr::kable(
  mtcars[1:5, ],
  caption = "A knitr kable."
)
```
```

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 Markdown: Practice

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 items

- `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(
  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
shinyApp(ui = ui, server = server)
```

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(
  fluidPage(
    navbarPage(title = "Runchee Technology",
               tabPanel("Product",
                        titlePanel("Hello!"),
                        "One more thing!"),
               tabPanel("About us",
                        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)))
        )
      )
    )
```

```
  )
)

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

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

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? 3

This is interactive document.

Assignments

- (Optional) Setup AWS and run EC2.
- Create a website with Shiny using navBar layout
  - *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!*

Runchee Technology | Product | About Us | Contact Us

Hello!

One more thing!

New breakthrough

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

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