Perl for Pipeline
Part I

L1110@BUMC

2/6/2018

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Tutorial Resource

Before we start, please take a note - all the code scripts and supporting documents are accessible through:

- http://rcs.bu.edu/examples/perl/tutorials/

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Sign In Sheet

We prepared sign-in sheet for each one to sign
We do this for internal management and quality control
So please SIGN IN if you haven't done so

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Research Computing Services (RCS)

- RCS is a group within Information Services & Technology (IS&T) at Boston University provides computing, storage, and visualization resources and services to support research that has specialized or highly intensive computation, storage, bandwidth, or graphics requirements.

- **Three Primary Services:**

    1. Research Computation
    2. Research Visualization
    3. Research Consulting and Training

- More Info: http://www.bu.edu/tech/about/research/

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Research Computing Services (RCS) Tutorials

RCS offers three times a year tutorials

- Spring – in January/Feburary
- Summer – in May/June
- Fall – in September/October

This tutorial is part I of a set (Part II come Thursday)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

About Me

- long time programmer, dated back in 1987
- Proficient in C/C++/Perl
- Domain knowledge: Network/Communication, Databases, Bioinformatics, System Integration.
- Contact: yshen16@bu.edu, 617-638-5851
- Main Office: 801 Mass Ave. 4th Floor (Crosstown Building)

Tell Me A bit about You

- Name
- Experience in programming? If so, which specific lauguage? Self rating?
- Experience in Perl?
- Account on SCC?
- Motivation (Expectation) to attend this tutorial
- Any other questions/fun facts you would like the class to know?

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Evaluation

One last piece of information before we start:

- DON'T FORGET TO GO TO:
  - http://rcs.bu.edu/survey/tutorial_evaluation.html

Leave your feedback for this tutorial (both good and bad as long as it is honest are welcome. Thank you)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Topics for today

HuRI - A Bioinformatical Pipeline Example

Get Back to Fundamentals

- Perl Environment
- Using Perl
- Code Examples

Advanced Features

- Packages, Modules and OO Methodology
- Perl Regular Expression
- Debugger

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI - A Real Bioinformatical Pipeline Example

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

Project Summary:

map high-quality binary protein-protein interactions (PPIs) is based on using yeast two-hybrid (Y2H) as the primary screening method followed by validation of subsets of PPIs in multiple orthogonal assays for binary PPI detection.

Three Stages:
HI-I-05: space of ~7,000 human genes, ~2,700 PPIs
HI-II-14: space of ~13,000 human genes , ~14,000 PPIs
HI-III: space of ~ 18,000 human genes, ~50,000+ PPIs up to 2015

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

The HI-III space is huge, AD 18k x DB 18k = ~320m binary pairs
Way to tackle : divide and conquer –>
divided entire space to 9 AD groups and 9 DB groups, that gives
9 x 9 = 81 matrices

What is the computational challenge by this design:
- more demanding in experiment design
- more complicated in algorithms
- more detail-oriented data storage and maintenance
- …

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

HuRI – Human Reference Interactome Map

Project Scope:

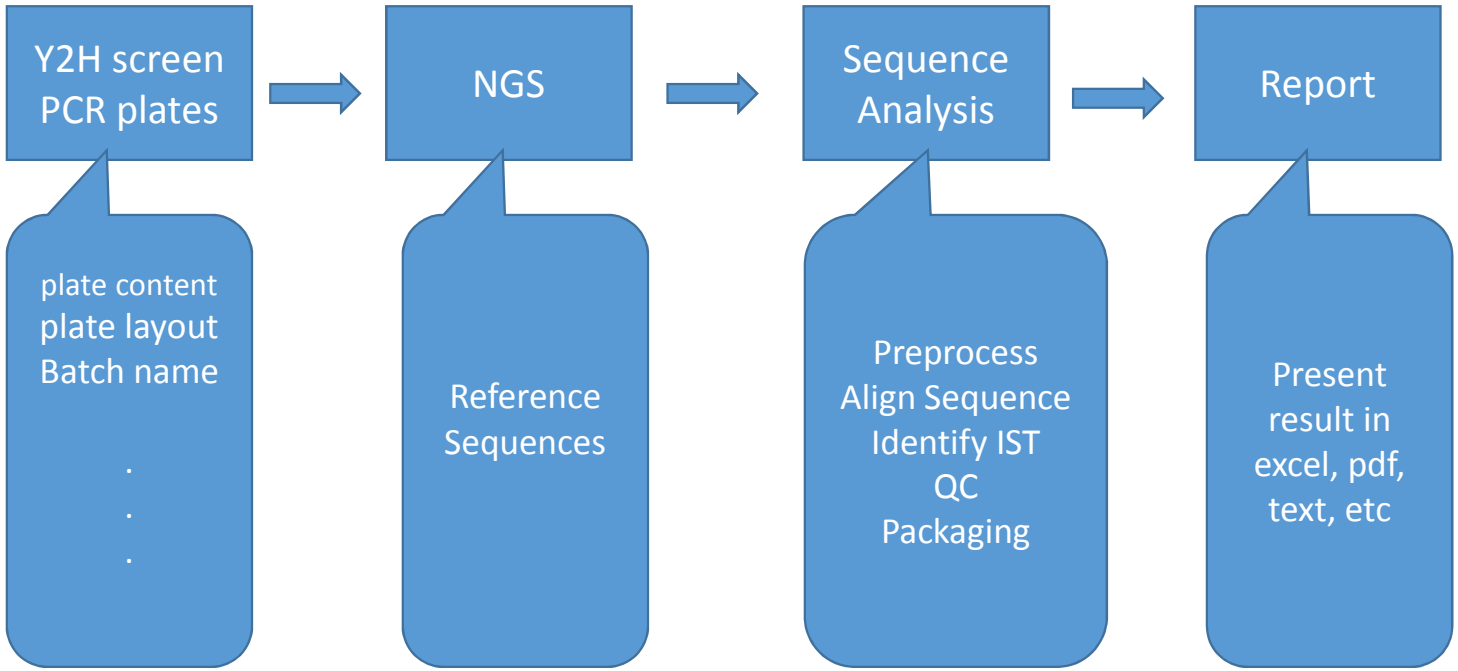
Total sequence batches: 35
Total PCR plates processed: 6528
Total Read count: ~1.3×10^9
Total Sequence File Size: ~3.5×10^{11}

With each plate be the result of colony pick of PCR product of thousands of AD and DB mating

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

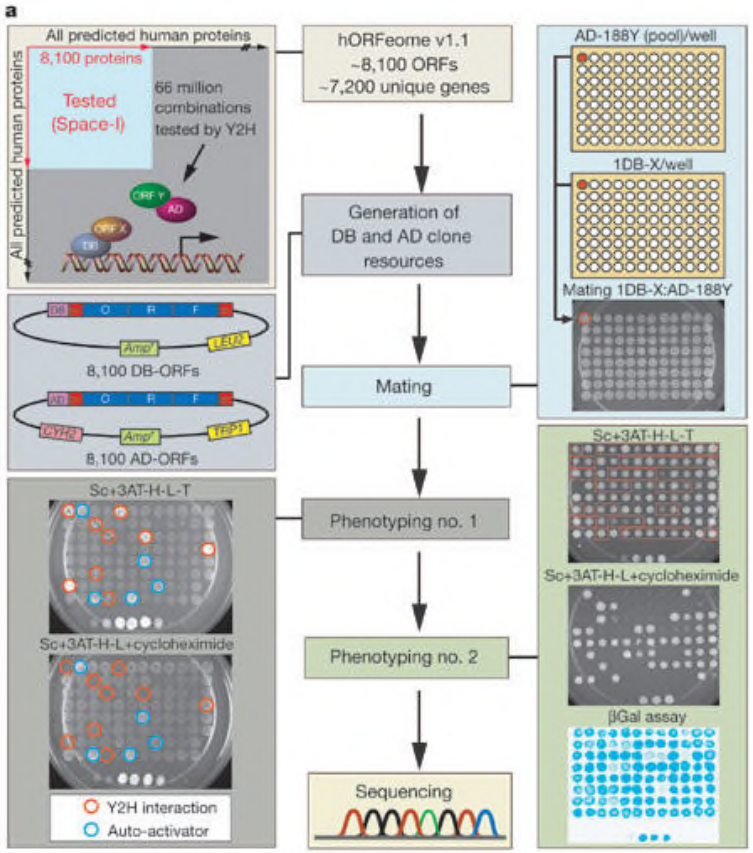
HuRI – Human Reference Interactome Map

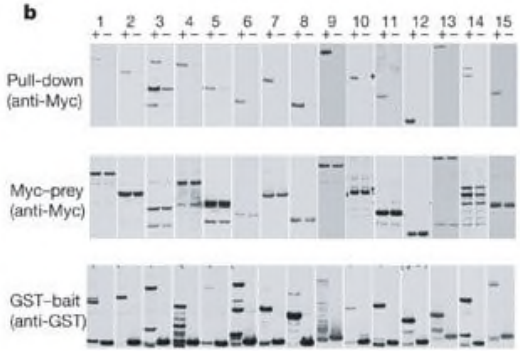


BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map





	Y2H-only	LCI-only	Y2H and LCI	Y2H/LCI-negative
Total	117	8	8	9
co-AP+	100	6	7	3
co-AP-	17	2	1	6
Success rate	85.5%	75%	87.5%	33.3%
Adjusted success rate	78.2%	62.5%	81.3%	NA

(source: https://www.ncbi.nlm.nih.gov/pubmed/16189514)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

You replied on 11/24/2014 10:34 AM.

From: [redacted]
To: [redacted]
Cc: [redacted]
Subject: Re: HuRI_r009

Message | HuRI_R009_samplesheet.xlsx (91 KB)

Hi All

Here is the data download link for HuRI_R009:

https://s3.amazonaws.com/SEQWELL/DATA/SWIM_20141121.tgz

Sample sheet is also attached. Run looked good on this end, but of couse please let me know if you have any issues.

Have a good weekend,
Joe

On Fri, Nov 21, 2014 at 9:23 AM, [redacted] wrote:
Great thanks Joe

Amanda, do you have everything you need ready to process this? We're hoping to include these results in the quarterly report due Dec 1st.

Cheers
m

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

HuRI – Human Reference Interactome Map

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	384_dest	SWIM_sample_i	SWIM_pla	fastq_name	384_ro	384_coll	96_src_l	96_src_p	96_src_i	96_src_c	i7 in	i5 in	% READS	FCID	La	SampleID	Sample	Index	Descript	Cont	Rec	Opera	SampleProj
197	L01		HuRI_r009	L01-HuRI_r009-D01	L	01	3	D01	D	01	TAAGGCG	CTGCTTC	0	A220V	1	L01-HuRI_r009-D01	hg18	TAAGGCGA-CTGCTTCG	L01	N		Joe	SWIM
198	M01		HuRI_r009	M01-HuRI_r009-E01	M	01	3	E01	E	01	TAAGGCG	GAGTCAG	0	A220V	1	M01-HuRI_r009-E01	hg18	TAAGGCGA-GAGTCAGT	M01	N		Joe	SWIM
199	N01		HuRI_r009	N01-HuRI_r009-F01	N	01	3	F01	F	01	TAAGGCG	GCTGATC	0	A220V	1	N01-HuRI_r009-F01	hg18	TAAGGCGA-GCTGATCG	N01	N		Joe	SWIM
200	O01		HuRI_r009	O01-HuRI_r009-G01	O	01	3	G01	G	01	TAAGGCG	TATGGAG	0	A220V	1	O01-HuRI_r009-G01	hg18	TAAGGCGA-TATGGAGG	O01	N		Joe	SWIM
201	P01		HuRI_r009	P01-HuRI_r009-H01	P	01	3	H01	H	01	TAAGGCG	TGATACA	0	A220V	1	P01-HuRI_r009-H01	hg18	TAAGGCGA-TGATACAT	P01	N		Joe	SWIM
202	I02		HuRI_r009	I02-HuRI_r009-A02	I	02	3	A02	A	02	CGTACTA	ACACAGC	0	A220V	1	I02-HuRI_r009-A02	hg18	CGTACTAG-ACACAGCT	I02	N		Joe	SWIM
203	J02		HuRI_r009	J02-HuRI_r009-B02	J	02	3	B02	B	02	CGTACTA	AGATCGT	0	A220V	1	J02-HuRI_r009-B02	hg18	CGTACTAG-AGATCGTC	J02	N		Joe	SWIM
204	K02		HuRI_r009	K02-HuRI_r009-C02	K	02	3	C02	C	02	CGTACTA	CCTATTG	0	A220V	1	K02-HuRI_r009-C02	hg18	CGTACTAG-CCTATTGA	K02	N		Joe	SWIM
205	L02		HuRI_r009	L02-HuRI_r009-D02	L	02	3	D02	D	02	CGTACTA	CTGCTTC	0	A220V	1	L02-HuRI_r009-D02	hg18	CGTACTAG-CTGCTTCG	L02	N		Joe	SWIM
206	M02		HuRI_r009	M02-HuRI_r009-E02	M	02	3	E02	E	02	CGTACTA	GAGTCAG	0	A220V	1	M02-HuRI_r009-E02	hg18	CGTACTAG-GAGTCAGT	M02	N		Joe	SWIM
207	N02		HuRI_r009	N02-HuRI_r009-F02	N	02	3	F02	F	02	CGTACTA	GCTGATC	0	A220V	1	N02-HuRI_r009-F02	hg18	CGTACTAG-GCTGATCG	N02	N		Joe	SWIM
208	O02		HuRI_r009	O02-HuRI_r009-G02	O	02	3	G02	G	02	CGTACTA	TATGGAG	0	A220V	1	O02-HuRI_r009-G02	hg18	CGTACTAG-TATGGAGG	O02	N		Joe	SWIM
209	P02		HuRI_r009	P02-HuRI_r009-H02	P	02	3	H02	H	02	CGTACTA	TGATACA	0	A220V	1	P02-HuRI_r009-H02	hg18	CGTACTAG-TGATACAT	P02	N		Joe	SWIM
210	I03		HuRI_r009	I03-HuRI_r009-A03	I	03	3	A03	A	03	AGGCAGA	ACACAGC	0	A220V	1	I03-HuRI_r009-A03	hg18	AGGCAGAA-ACACAGCT	I03	N		Joe	SWIM
211	J03		HuRI_r009	J03-HuRI_r009-B03	J	03	3	B03	B	03	AGGCAGA	AGATCGT	0	A220V	1	J03-HuRI_r009-B03	hg18	AGGCAGAA-AGATCGTC	J03	N		Joe	SWIM
212	K03		HuRI_r009	K03-HuRI_r009-C03	K	03	3	C03	C	03	AGGCAGA	CCTATTG	0	A220V	1	K03-HuRI_r009-C03	hg18	AGGCAGAA-CCTATTGA	K03	N		Joe	SWIM
213	L03		HuRI_r009	L03-HuRI_r009-D03	L	03	3	D03	D	03	AGGCAGA	CTGCTTC	0	A220V	1	L03-HuRI_r009-D03	hg18	AGGCAGAA-CTGCTTCG	L03	N		Joe	SWIM
214	M03		HuRI_r009	M03-HuRI_r009-E03	M	03	3	E03	E	03	AGGCAGA	GAGTCAG	0	A220V	1	M03-HuRI_r009-E03	hg18	AGGCAGAA-GAGTCAGT	M03	N		Joe	SWIM
215	N03		HuRI_r009	N03-HuRI_r009-F03	N	03	3	F03	F	03	AGGCAGA	GCTGATC	0	A220V	1	N03-HuRI_r009-F03	hg18	AGGCAGAA-GCTGATCG	N03	N		Joe	SWIM
216	O03		HuRI_r009	O03-HuRI_r009-G03	O	03	3	G03	G	03	AGGCAGA	TATGGAG	0	A220V	1	O03-HuRI_r009-G03	hg18	AGGCAGAA-TATGGAGG	O03	N		Joe	SWIM
217	P03		HuRI_r009	P03-HuRI_r009-H03	P	03	3	H03	H	03	AGGCAGA	TGATACA	0	A220V	1	P03-HuRI_r009-H03	hg18	AGGCAGAA-TGATACAT	P03	N		Joe	SWIM
218	I04		HuRI_r009	I04-HuRI_r009-A04	I	04	3	A04	A	04	TCCTGAG	ACACAGC	0	A220V	1	I04-HuRI_r009-A04	hg18	TCCTGAGC-ACACAGCT	I04	N		Joe	SWIM
219	J04		HuRI_r009	J04-HuRI_r009-B04	J	04	3	B04	B	04	TCCTGAG	AGATCGT	0	A220V	1	J04-HuRI_r009-B04	hg18	TCCTGAGC-AGATCGTC	J04	N		Joe	SWIM
220	K04		HuRI_r009	K04-HuRI_r009-C04	K	04	3	C04	C	04	TCCTGAG	CCTATTG	0	A220V	1	K04-HuRI_r009-C04	hg18	TCCTGAGC-CCTATTGA	K04	N		Joe	SWIM
221	L04		HuRI_r009	L04-HuRI_r009-D04	L	04	3	D04	D	04	TCCTGAG	CTGCTTC	0	A220V	1	L04-HuRI_r009-D04	hg18	TCCTGAGC-CTGCTTCG	L04	N		Joe	SWIM
222	M04		HuRI_r009	M04-HuRI_r009-E04	M	04	3	E04	E	04	TCCTGAG	GAGTCAG	0	A220V	1	M04-HuRI_r009-E04	hg18	TCCTGAGC-GAGTCAGT	M04	N		Joe	SWIM
223	N04		HuRI_r009	N04-HuRI_r009-F04	N	04	3	F04	F	04	TCCTGAG	GCTGATC	0	A220V	1	N04-HuRI_r009-F04	hg18	TCCTGAGC-GCTGATCG	N04	N		Joe	SWIM
224	O04		HuRI_r009	O04-HuRI_r009-G04	O	04	3	G04	G	04	TCCTGAG	TATGGAG	0	A220V	1	O04-HuRI_r009-G04	hg18	TCCTGAGC-TATGGAGG	O04	N		Joe	SWIM
225	P04		HuRI_r009	P04-HuRI_r009-H04	P	04	3	H04	H	04	TCCTGAG	TGATACA	0	A220V	1	P04-HuRI_r009-H04	hg18	TCCTGAGC-TGATACAT	P04	N		Joe	SWIM
226	I05		HuRI_r009	I05-HuRI_r009-A05	I	05	3	A05	A	05	GGACTCC	ACACAGC	0	A220V	1	I05-HuRI_r009-A05	hg18	GGACTCCT-ACACAGCT	I05	N		Joe	SWIM
227	J05		HuRI_r009	J05-HuRI_r009-B05	J	05	3	B05	B	05	GGACTCC	AGATCGT	0	A220V	1	J05-HuRI_r009-B05	hg18	GGACTCCT-AGATCGTC	J05	N		Joe	SWIM
228	K05		HuRI_r009	K05-HuRI_r009-C05	K	05	3	C05	C	05	GGACTCC	CCTATTG	0	A220V	1	K05-HuRI_r009-C05	hg18	GGACTCCT-CCTATTGA	K05	N		Joe	SWIM
229	L05		HuRI_r009	L05-HuRI_r009-D05	L	05	3	D05	D	05	GGACTCC	CTGCTTC	0	A220V	1	L05-HuRI_r009-D05	hg18	GGACTCCT-CTGCTTCG	L05	N		Joe	SWIM
230	M05		HuRI_r009	M05-HuRI_r009-E05	M	05	3	E05	E	05	GGACTCC	GAGTCAG	0	A220V	1	M05-HuRI_r009-E05	hg18	GGACTCCT-GAGTCAGT	M05	N		Joe	SWIM
231	N05		HuRI_r009	N05-HuRI_r009-F05	N	05	3	F05	F	05	GGACTCC	GCTGATC	0	A220V	1	N05-HuRI_r009-F05	hg18	GGACTCCT-GCTGATCG	N05	N		Joe	SWIM
232	O05		HuRI_r009	O05-HuRI_r009-G05	O	05	3	G05	G	05	GGACTCC	TATGGAG	0	A220V	1	O05-HuRI_r009-G05	hg18	GGACTCCT-TATGGAGG	O05	N		Joe	SWIM
233	P05		HuRI_r009	P05-HuRI_r009-H05	P	05	3	H05	H	05	GGACTCC	TGATACA	0	A220V	1	P05-HuRI_r009-H05	hg18	GGACTCCT-TGATACAT	P05	N		Joe	SWIM
234	I06		HuRI_r009	I06-HuRI_r009-A06	I	06	3	A06	A	06	TAGGCAT	ACACAGC	0	A220V	1	I06-HuRI_r009-A06	hg18	TAGGCATG-ACACAGCT	I06	N		Joe	SWIM
235	J06		HuRI_r009	J06-HuRI_r009-B06	J	06	3	B06	B	06	TAGGCAT	AGATCGT	0	A220V	1	J06-HuRI_r009-B06	hg18	TAGGCATG-AGATCGTC	J06	N		Joe	SWIM
236	K06		HuRI_r009	K06-HuRI_r009-C06	K	06	3	C06	C	06	TAGGCAT	CCTATTG	0	A220V	1	K06-HuRI_r009-C06	hg18	TAGGCATG-CCTATTGA	K06	N		Joe	SWIM
237	L06		HuRI_r009	L06-HuRI_r009-D06	L	06	3	D06	D	06	TAGGCAT	CTGCTTC	0	A220V	1	L06-HuRI_r009-D06	hg18	TAGGCATG-CTGCTTCG	L06	N		Joe	SWIM
238	M06		HuRI_r009	M06-HuRI_r009-E06	M	06	3	E06	E	06	TAGGCAT	GAGTCAG	0	A220V	1	M06-HuRI_r009-E06	hg18	TAGGCATG-GAGTCAGT	M06	N		Joe	SWIM
239	N06		HuRI_r009	N06-HuRI_r009-F06	N	06	3	F06	F	06	TAGGCAT	GCTGATC	0	A220V	1	N06-HuRI_r009-F06	hg18	TAGGCATG-GCTGATCG	N06	N		Joe	SWIM

sample sheet

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

yun@north:/North/bioinfo/sp3/SWIM_20141121_v2_huri_r009/Sample_I01-HuRI_...

```
[yun@north sp3]$ cd SWIM_20141121_v2_huri_r009/
[yun@north SWIM_20141121_v2_huri_r009]$ ls
Sample_I01-HuRI_r009-A01  Sample_K07-HuRI_r009-C07  Sample_N04-HuRI_r009-F04
Sample_I02-HuRI_r009-A02  Sample_K08-HuRI_r009-C08  Sample_N05-HuRI_r009-F05
Sample_I03-HuRI_r009-A03  Sample_K09-HuRI_r009-C09  Sample_N06-HuRI_r009-F06
Sample_I04-HuRI_r009-A04  Sample_K10-HuRI_r009-C10  Sample_N07-HuRI_r009-F07
Sample_I05-HuRI_r009-A05  Sample_L01-HuRI_r009-D01  Sample_N08-HuRI_r009-F08
Sample_I06-HuRI_r009-A06  Sample_L02-HuRI_r009-D02  Sample_N09-HuRI_r009-F09
Sample_I07-HuRI_r009-A07  Sample_L03-HuRI_r009-D03  Sample_O01-HuRI_r009-G01
Sample_I08-HuRI_r009-A08  Sample_L04-HuRI_r009-D04  Sample_O02-HuRI_r009-G02
Sample_I09-HuRI_r009-A09  Sample_L05-HuRI_r009-D05  Sample_O03-HuRI_r009-G03
Sample_I10-HuRI_r009-A10  Sample_L06-HuRI_r009-D06  Sample_O04-HuRI_r009-G04
Sample_J01-HuRI_r009-B01  Sample_L07-HuRI_r009-D07  Sample_O05-HuRI_r009-G05
Sample_J02-HuRI_r009-B02  Sample_L08-HuRI_r009-D08  Sample_O06-HuRI_r009-G06
Sample_J03-HuRI_r009-B03  Sample_L09-HuRI_r009-D09  Sample_O07-HuRI_r009-G07
Sample_J04-HuRI_r009-B04  Sample_L10-HuRI_r009-D10  Sample_O08-HuRI_r009-G08
Sample_J05-HuRI_r009-B05  Sample_M01-HuRI_r009-E01  Sample_O09-HuRI_r009-G09
Sample_J06-HuRI_r009-B06  Sample_M02-HuRI_r009-E02  Sample_P01-HuRI_r009-H01
Sample_J07-HuRI_r009-B07  Sample_M03-HuRI_r009-E03  Sample_P02-HuRI_r009-H02
Sample_J08-HuRI_r009-B08  Sample_M04-HuRI_r009-E04  Sample_P03-HuRI_r009-H03
Sample_J09-HuRI_r009-B09  Sample_M05-HuRI_r009-E05  Sample_P04-HuRI_r009-H04
Sample_J10-HuRI_r009-B10  Sample_M06-HuRI_r009-E06  Sample_P05-HuRI_r009-H05
Sample_K01-HuRI_r009-C01  Sample_M07-HuRI_r009-E07  Sample_P06-HuRI_r009-H06
Sample_K02-HuRI_r009-C02  Sample_M08-HuRI_r009-E08  Sample_P07-HuRI_r009-H07
Sample_K03-HuRI_r009-C03  Sample_M09-HuRI_r009-E09  Sample_P08-HuRI_r009-H08
Sample_K04-HuRI_r009-C04  Sample_N01-HuRI_r009-F01  Sample_P09-HuRI_r009-H09
Sample_K05-HuRI_r009-C05  Sample_N02-HuRI_r009-F02  tmp
Sample_K06-HuRI_r009-C06  Sample_N03-HuRI_r009-F03
[yun@north SWIM_20141121_v2_huri_r009]$ cd Sample_I01-HuRI_r009-A01
[yun@north Sample_I01-HuRI_r009-A01]$ ls
I01-HuRI_r009-A01_TAAGGCGA-ACACAGCT_L001_R1_001.fastq  SampleSheet.csv
I01-HuRI_r009-A01_TAAGGCGA-ACACAGCT_L001_R2_001.fastq
[yun@north Sample_I01-HuRI_r009-A01]$
```

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

Output-Summary :

	A	B	C	D	E
1	POOL_ID	136			
2	POOL_NAME	ds20141122_S4_HuRI_r009_Hs04a2			
3	TOTAL_PLATES	30			
4	AVG_RAWREAD_PER_PLATE	104184.0333			
5	TOTAL_FAIL_PLATE	5			
6	TOTAL_IST@swim0.2	1749			
7	UNIQ_IST@swim0.2	1405			
8	UNIQ_AD@swim0.2	369			
9	UNIQ_DB@swim0.2	791			
10	UNIQ_WELL@swim0.2	1478			
11					
12					
13	****BELOW ARE FAILED PLATES AT SWIM_CUTOFF=0.2**********				
14	POOL_ID	PLATE_ID	PLATE_STD_NAME	DS_ID	
15	136	4	p1E07_Hs04a2d1_004	4	
16	136	14	p1E17_Hs04a2d4_003	14	
17	136	15	p1E18_Hs04a2d4_004	15	
18	136	21	p1E24_Hs04a2d6_003	21	
19	136	28	p1E31_Hs04a2d9_001	28	
20					
21	********** AD HUB with degree>30 **********				
22	AD_ORF_ID	DEGREE			
23	54982	54			
24	70073	44			
25	5231	43			
26	100016209	42			
27	100016119	37			
28	70282	33			
29	71924	31			
30	6490	31			
31					
32					

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

HuRI – Human Reference Interactome Map

Output-Detail :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	DS_ID	PLATE_ID	PLATE_STD_NAME	PLATE_ALIAS	PRIEMR_SET	RAWREAD	LAST_WELL	WELL_COUNT	RAWREAD_PER_WELL	AD_GROUP_EXP	DB_GROUP_EXP	AD_GROUP_BEST	DB_GROUP_BEST	IST_PER_USED_WELL@0.2
2	1	1	p1E04_Hs04a2d1_001	p1E04_Hs04a2d1_001	E	107886	H11	95	1135.642105	2	1	2	1	0.8
3	2	2	p1E05_Hs04a2d1_002	p1E05_Hs04a2d1_002	E	120356	H11	95	1266.905263	2	1	2	1	1.21
4	3	3	p1E06_Hs04a2d1_003	p1E06_Hs04a2d1_003	E	112378	H11	95	1182.926316	2	1	2	1	0.78
5	4	4	p1E07_Hs04a2d1_004	p1E07_Hs04a2d1_004	E	29147	A3	3	9715.666667	2	1		1	0.33
6	5	5	p1E08_Hs04a2d2_001	p1E08_Hs04a2d2_001	E	116250	H11	95	1223.684211	2	2	2	2	0.78
7	6	6	p1E09_Hs04a2d2_002	p1E09_Hs04a2d2_002	E	106552	H11	95	1121.6	2	2	2	2	0.77
8	7	7	p1E10_Hs04a2d2_003	p1E10_Hs04a2d2_003	E	111521	H8	92	1212.184783	2	2	2	2	0.73
9	8	8	p1E11_Hs04a2d3_001	p1E11_Hs04a2d3_001	E	129438	H11	95	1362.505263	2	3	2	3	0.87
10	9	9	p1E12_Hs04a2d3_002	p1E12_Hs04a2d3_002	E	111573	H11	95	1174.452632	2	3	2	3	1.04
11	10	10	p1E13_Hs04a2d3_003	p1E13_Hs04a2d3_003	E	123710	H11	95	1302.210526	2	3	2	3	0.88
12	11	11	p1E14_Hs04a2d3_004	p1E14_Hs04a2d3_004	E	91926	B9	21	4377.428571	2	3	2	3	0.9
13	12	12	p1E15_Hs04a2d4_001	p1E15_Hs04a2d4_001	E	147682	H11	95	1554.547368	2	4	2	4	0.84
14	13	13	p1E16_Hs04a2d4_002	p1E16_Hs04a2d4_002	E	16624	H11	95	174.9894737	2	4	2	4	0.58
15	14	14	p1E17_Hs04a2d4_003	p1E17_Hs04a2d4_003	E	103013	H11	95	1084.347368	2	4	2	4	0.47
16	15	15	p1E18_Hs04a2d4_004	p1E18_Hs04a2d4_004	E	64292	A9	9	7143.555556	2	4	2	4	0.22
17	16	16	p1E19_Hs04a2d5_001	p1E19_Hs04a2d5_001	E	106564	H11	95	1121.726316	2	5	2	5	0.65
18	17	17	p1E20_Hs04a2d5_002	p1E20_Hs04a2d5_002	E	97392	H11	95	1025.178947	2	5	2	5	0.79
19	18	18	p1E21_Hs04a2d5_003	p1E21_Hs04a2d5_003	E	106015	E6	54	1963.240741	2	5	2	5	0.81
20	19	19	p1E22_Hs04a2d6_001	p1E22_Hs04a2d6_001	E	123082	H11	95	1295.6	2	6	2	6	0.68
21	20	20	p1E23_Hs04a2d6_002	p1E23_Hs04a2d6_002	E	50257	H11	95	529.0210526	2	6	2	6	0.59
22	21	21	p1E24_Hs04a2d6_003	p1E24_Hs04a2d6_003	E	89892	D10	46	1954.173913	2	6	2	6	0.3
23	22	22	p1E25_Hs04a2d7_001	p1E25_Hs04a2d7_001	E	138814	H11	95	1461.2	2	7	2	7	0.62
24	23	23	p1E26_Hs04a2d7_002	p1E26_Hs04a2d7_002	E	114693	H11	95	1207.294737	2	7	2	7	0.67
25	24	24	p1E27_Hs04a2d7_003	p1E27_Hs04a2d7_003	E	132609	H11	95	1395.884211	2	7	2	7	0.73
26	25	25	p1E28_Hs04a2d8_001	p1E28_Hs04a2d8_001	E	131739	H11	95	1386.726316	2	8	2	8	0.69
27	26	26	p1E29_Hs04a2d8_002	p1E29_Hs04a2d8_002	E	121707	H11	95	1281.126316	2	8	2	8	0.55
28	27	27	p1E30_Hs04a2d8_003	p1E30_Hs04a2d8_003	E	95845	E10	58	1652.5	2	8	2	8	0.6
29	28	28	p1E31_Hs04a2d9_001	p1E31_Hs04a2d9_001	E	96808	H11	95	1019.031579	2	9	2	9	0.42
30	29	29	p1E32_Hs04a2d9_002	p1E32_Hs04a2d9_002	E	124382	H11	95	1309.284211	2	9	2	9	0.86
31	30	30	p1E33_Hs04a2d9_003	p1E33_Hs04a2d9_003	E	103374	C1	25	4134.96	2	9	2	9	0.76

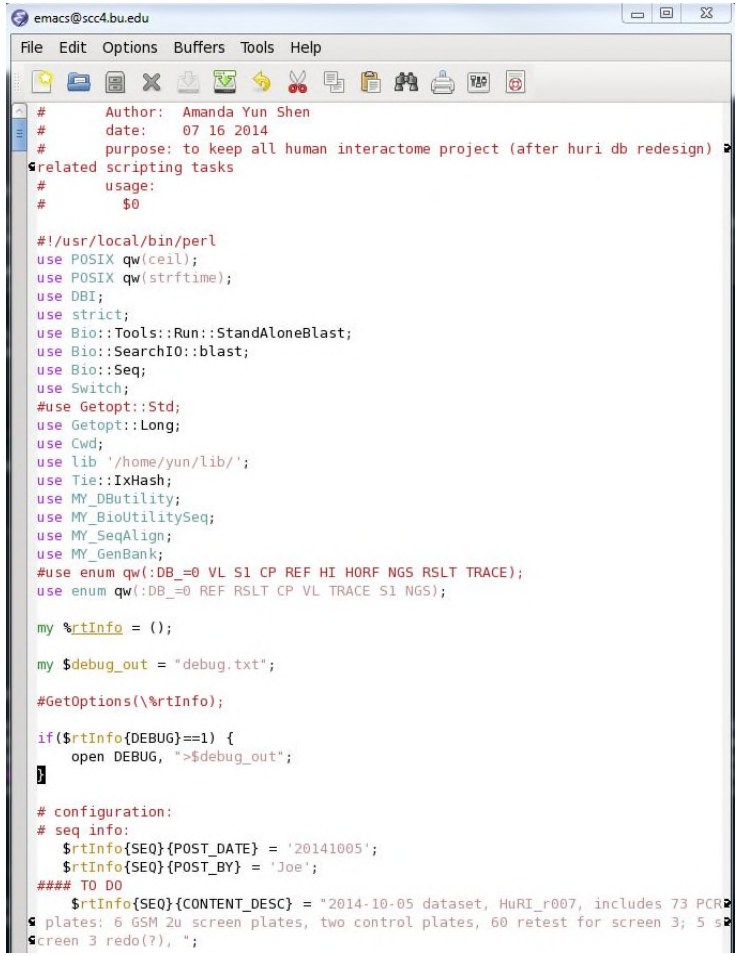
BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

So how do we achieve this ??

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Pipeline code:
Huri_pipeline.pl




```
#       Author:  Amanda Yun Shen
#       date:    07 16 2014
#       purpose: to keep all human interactome project (after huri db redesign)
related scripting tasks
#       usage:
#         $0

#!/usr/local/bin/perl
use POSIX qw(ceil);
use POSIX qw(strftime);
use DBI;
use strict;
use Bio::Tools::Run::StandAloneBlast;
use Bio::SearchIO::blast;
use Bio::Seq;
use Switch;
#use Getopt::Std;
use Getopt::Long;
use Cwd;
use lib '/home/yun/lib/';
use Tie::IxHash;
use MY_DButility;
use MY_BioUtilitySeq;
use MY_SeqAlign;
use MY_GenBank;
#use enum qw(:DB_=0 VL S1 CP REF HI HORF NGS RSLT TRACE);
use enum qw(:DB_=0 REF RSLT CP VL TRACE S1 NGS);

my %rtInfo = ();

my $debug_out = "debug.txt";

#GetOptions(\%rtInfo);

if($rtInfo{DEBUG}==1) {
    open DEBUG, ">$debug_out";
}

# configuration:
# seq info:
   $rtInfo{SEQ}{POST_DATE} = '20141005';
   $rtInfo{SEQ}{POST_BY} = 'Joe';
#### TO DO
    $rtInfo{SEQ}{CONTENT_DESC} = "2014-10-05 dataset, HuRI_r007, includes 73 PCR
 plates: 6 GSM 2u screen plates, two control plates, 60 retest for screen 3; 5 s
creen 3 redo(?), ";
```

We will come back later

Perl Language Fundamentals

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Language Design Philosophy

- "There's more than one way to do it" design philosophy and multi-paradigm, dynamically typed language features leads to great degree of flexibility in program design.

- CPAN and Perl Modules (191,032 available modules in CPAN in 35,637 distributions, written by 13,218 authors, mirrored on 250 servers over 60 countries)

- CPAN is honored to be called Perl's 'killer app' (see https://en.wikipedia.org/wiki/CPAN for more)

Perl Classification

Perl 5 and 6 are considered a family of high-level, general-purpose, interpreted, dynamic programming languages.

- High-level – syntax/semantics close to natural language
- General purpose – not limited to specific tasks in a particular application domain
- Interpreted – relative to compiled language (prepared/checked vs real-time/interactive)
- Dynamic – not strict in predefined data type constraints, etc.

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Borrowed Features

Perl Borrows many features from other programming languages

- From C: procedural, variables, expression, assignment (=), brace-delimited blocks ({}, ;), control flow (if, while, for, do, etc), subroutine
- From shell: '$' sign, system command
- From Lisp: lists data structure; implicit return value
- From AWK: hash
- From sed: regular expression

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Authentic Features

Perl's most authentic features of its own:

- auto data-typing
- auto memory management
- It's all handled by Perl interpreter

These are very powerful features and contribute a lot to the wide adoption of Perl language

more details on Perl5 feature summary: https://www.perl.org/about.html

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Where Perl is used

- System administration
- Configuration management
- Web sites/web application
- Small scripts
- Bioinformatics
- Scientific calculations
- Test automation
- … (the riches lie in CPAN)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Swiss Army Chainsaw or Duct Tape of Internet?

Perl gained its nickname of 'Swiss army chainsaw' for its flexibility and power; its 'Duct Tape of Internet' for its ability and often 'ugly', quick, easy fixes for solutions to various problems. Commonly referred applications:

- Powerful text processing without data length limitation
- Regular expression and string parsing capability
- CGI (duct tape, glue language for Internet)
- DBI
- BioPerl

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Major versions

- Perl 5 – almost rewrite of Perl interpreter, adding object-oriented (OO) feature, complex data structure, module and CGI support. Among them, module support plays critical role to CPAN's establishment, and nowadays a great resource and strength for Perl community
- Perl 6 – fundamentally different from Perl 5, dedicated to Larry's birthday, goal is to fix all the warts in Perl 5; it's said to be good at all that Perl 5 is good at, and a lot more.

Language Scope

- Perl is highly extensive language
- Open source framework – CPAN model
- CPAN and Perl Module
  - 191,032 available modules
  - 35, 637 distributions
  - written by 13,218 authors
  - mirrored on 250 servers

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Language Elements

- Data Types
  - scalar, array, hash, reference
- Control Structures
  - for, while, if, next, last, goto (yes, there is a Goto)
- Regular Expressions
- User Defined Extensions (Subroutines and functions)
- Objects/modules/packages

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Advantage Over C

- Perl runs on all platforms and is far more portable than C.
- Perl and a huge collection of Perl Modules are free software (either GNU General Public License or Artistic License).
- Perl is very efficient in TEXT and STRING manipulation i.e. REGEXP.
- It is a language that combines the best features from many other languages and is very easy to learn.
- Dynamic memory allocation is very easy in PERL, at any point of time we can increase or decrease the size of the array (i.e. splice(), push())

Disadvantage Over C

- You cannot easily create a binary image ("exe") from a Perl file. It's not a serious problem on Unix, but it might be a problem on Windows.
- Moreover, if you write a script which uses modules from CPAN, and want to run it on another computer, you need to install all the modules on that other computer, which can be a drag.
- Perl is an interpretative language, so its comparatively slower to other compiling language like C. So, it’s not feasible to use in Real time environment like in flight simulation system.

Some famous applications

- Web CGI (EBay, Craigslist, BBC, Amazon, …)
- 1000 Genome Project
- Financial analysis (ease of use, speed for integration, rapid prototyping) - BarclaysCapital
- Summarizing system logs/deal with Windows registry or Unix Passwd or groups file

Get To Know Environment

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Connecting to SCC

- Option 1: You are able to keep everything you generate
  Use your Shared Computing Cluster account if you have one.

- Option 2: all that you do in the tutorial may be wiped out after tutorial ends unless you move the contents to somewhere belong to you.

  Tutorial accounts if you need one (will be offered in class).
  - Username: TBD
  - Password: TBD

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Download source code

Follow these steps to download the code:

ssh user@sccN.bu.edu ('user' is an account on SCC, 'N' can be 1-4)

mkdir perlThruEx

cd perlThruEx

wget http://scv.bu.edu/examples/perl/tutorials/src/perlThruExamples.zip

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 1 - Where is My Perl

Two commands to use:

'which perl'

and

'perl -v'

Do the experiment on next page to help understand the concept and discover more

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 1a - Where is My Perl

Type ‘which perl’ in terminal

```
[yshen16@scc4 beginner_perl]$ which perl
/usr/local/bin/perl
```

Now type ‘perl -v’

```
[yshen16@scc4 beginner_perl]$ perl -v

This is perl, v5.10.1 (*) built for x86_64-linux-thread-multi

Copyright 1987-2009, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 1b - Where is My Perl

Type ‘module load perl’, then type ‘which perl’ in terminal

```
[yshen16@scc4 beginner_perl]$ module load perl
[yshen16@scc4 beginner_perl]$ which perl
/share/pkg/perl/5.24.0/install/bin/perl
```

Now type ‘perl -v’

```
[yshen16@scc4 beginner_perl]$ perl -v

This is perl 5, version 24, subversion 0 (v5.24.0) built for x86_64-linux

Copyright 1987-2016, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```


Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 1 - Observation

What's the difference between Exercise 1a and 1b?

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

What do we learn from Exercise 1

- Perl is an environment – means
  it can be changed by pointing to different installations.

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 2 – Perl Program Structure

Open code examples in gedit and browse the content:

codeEx_simplest.pl and codeEx_simplest.pl.nofirst

Try to run the following commands:

```
./codeEx_simplest.pl
./codeEx_simplest.pl.nofirst
```

What happened?

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 2 – Perl Program Structure (2)

Here is what would be:

```
[yshen16@scc4 code]$ ./codeEx_simplest.pl
Hello World!
[yshen16@scc4 code]$ ./codeEx_simplest.pl.nofirst
./codeEx_simplest.pl.nofirst: line 3: print: command not found
[yshen16@scc4 code]$
```

Now try to run the following command:

perl ./codeEx_simplest.pl.nofirst

What happened?

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 2 – Perl Program Structure (3)

Here is what would be this time:

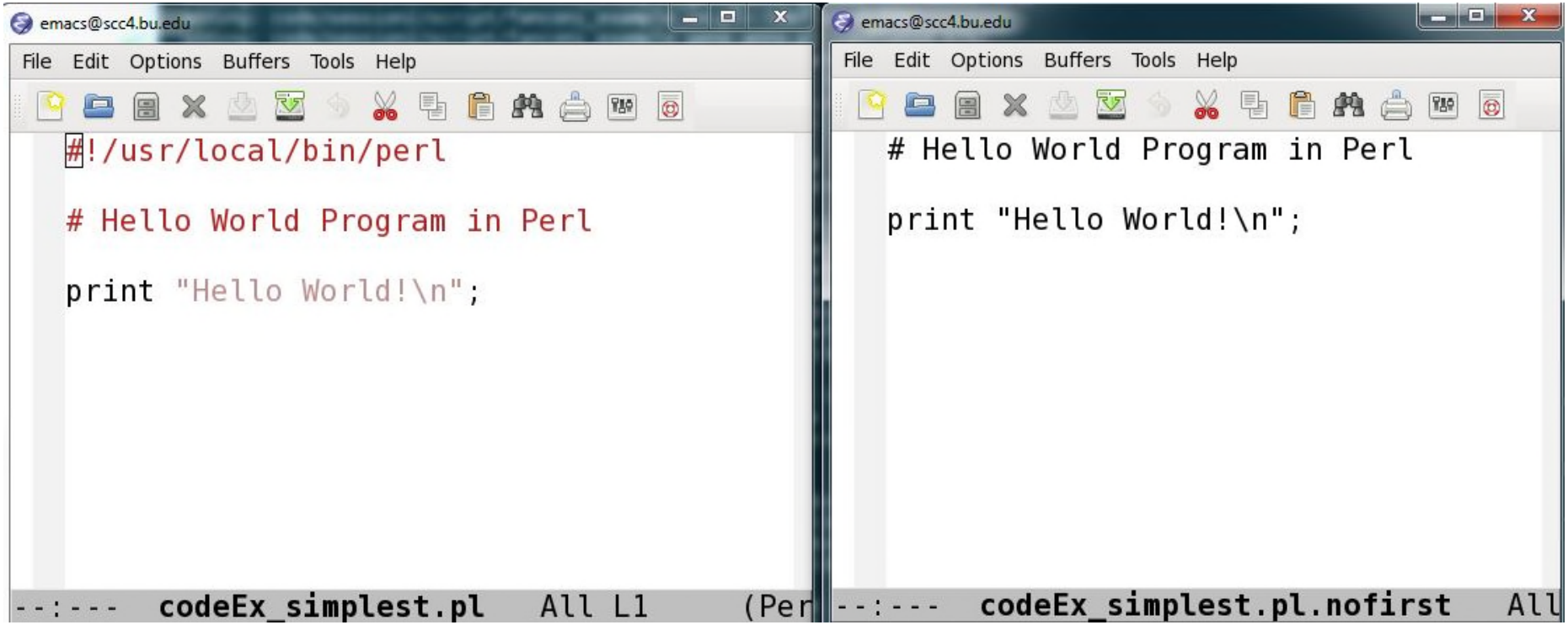
```
[yshen16@scc4 code]$ ./codeEx_simplest.pl.nofirst
./codeEx_simplest.pl.nofirst: line 3: print: command not found
[yshen16@scc4 code]$ perl ./codeEx_simplest.pl.nofirst
Hello World!
[yshen16@scc4 code]$
```

So why? Why is 'perl' in the command so critical to the 2nd code example?

Topic: Perl program and OS

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Exercise 2 – Check Source Code



BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Comments on Exercise 2

Comment#1: file name doesn't matter (.pl is just a convention)
Comment#2: file permission doesn't matter (the file can be in plain readable text permission)

Reason: in the first command, **./codeEx_simplest.pl**, the file functions as an executable (in this case, the executable permission is a must), and inside the script, it must contains the location for the perl interpreter (which is what the first line of the code does)

But in the second form with perl leading the command: the file functions as mere an input parameter to feed 'perl' command. The true executable from OS point is 'perl' program itself.

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

What do we learn from Exercise 2

- Importance of the first line of almost every Perl script (Perl Interpreter is mandatory to be present)
- This is why the path has to be specified in each Perl script to let the system know where to start (this is called ‘Entry Point’)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Using Perl

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Command line Option Explained

- Command format:
  perl -[v|p|e|i] “perl statement/expression” input

- Options: (type “perl -h” for more options)
  -e # tell perl to execute some statements in what is quoted following
  -v # check current perl version
  -i[extension] # edit input files in place (makes backup if extension supplied)
  -n # assume "while (<>) { ... }" loop around program
  -p # assume loop like -n but print line also

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Command line Examples

- perl -e 'print "Hello World\n"'
  - same result as run ‘codeEx_simplest.pl’
- perl -n -e 'print "$. - $_"' codeEx_simplest.pl
  - implicit loop, print code with line number
- perl -p -e '$_="$. - $_"' codeEx_simplest.pl
  - implicit loop, implicit print, , using $_ new assignment
- perl -ne 'print "$. - $_" unless /^#/' codeEx_simplest.pl
  - implicit loop, print code with line number
- perl -ne 'print "$. - $_" if /^#/' codeEx_simplest.pl
  - print all lines that are starting with ‘#’, that is, all comment lines
- perl -ne 'print "$. - $_" if $.<=5' codeEx_simplest.pl
  - Print the first 5 lines

Good Programming Practices

- Always starts with hash-bang line
  #!/usr/local/bin/perl
- Using template/framework to standardize and simplify code tasks (see MyFramework.pl for explanation)
- Learn to using Perl debugger tool rather than use 'print'
- Start with minimum code required (isolate code)
- Reduce interference by defining good interfaces through subroutines
- Pay attention to format (especially with statement across multiple lines)
- Many more … (refer to 'Perl Best Practice')

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Good Programming Practices Code Example

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Variable Scope

- What is scope? The space that something is seen/valid
- Two types of scope: Global vs. Lexical
  - Global variable – visible in the entire package, ‘our’ keyword
  - lexical variable – only visible in the context, with ‘my’ keyword
- Override: Inside variable overrides(hides) the outside variable
- Package independence - same variable name can be used in different packages, they are totally independent and won’t affect each other
- Use namespace to provide specificity – use “package::variable” qualifier

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Variable Scope Example 1

Variable scope: enclosing block

```
#!/usr/bin/perl
use strict;
use warnings;

{
    my $email = 'foo@bar.com';
    print "$email\n";     # foo@bar.com
}
# print $email;
# $email does not exists
# Global symbol "$email" requires explicit package name at ...
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Variable Scope Example 2

Variable hidden by other declaration

```
#!/usr/bin/perl
use strict;
use warnings;

my $fname = "Foo";
print "$fname\n";        # Foo

{
    print "$fname\n";    # Foo

    my $fname  = "Other";
    print "$fname\n";    # Other
}
print "$fname\n";        # Foo
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Variable Scope Example 3

```
[yshen16@scc4 session1]$ more codeEx_varScope_namespace.pl
#!/usr/local/bin/perl
use strict;
use warnings;

package Calc;
use strict;
use warnings;

our $total = 100;

sub add {
    my $total=0;
    $total += $_ for (@_);
    return $total;
}


package main;

my $total = Calc::add(3, 4);
print "\$total in Main: $total\n";
print "\$total in Calc: ${Calc::total}\n";
[yshen16@scc4 session1]$
[yshen16@scc4 session1]$
[yshen16@scc4 session1]$ perl codeEx_varScope_namespace.pl
"my" variable $total masks earlier declaration in same scope at codeEx_varScope_namespace.pl line 20.
$total in Main: 7
$total in Calc: 100
[yshen16@scc4 session1]$
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Variable Scope Good Practice

To avoid ambiguity –

- avoid using same name for different variables unless you are sure they are meant to be same thing ;
- use meaningful names for each variable

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Special Symbols

- Also called ‘pre-defined variables’ in perldoc
- Can be divided into five categories:
  - General Variables
  - Regular Expression Variables
  - Filehandle Variables
  - Error Variables
  - State Variables
- Perl programming depends highly on using these special symbols (variables, more officially). So it is good to know about them.
- Use ‘perldoc perlvar’ to read the help documentation

Special Symbols - General

$ARG/$_ – default input space
@ARG/@_ – parameter array for subroutine

```
sub max {
    my $max = shift(@_);
    foreach $foo (@_) {
        $max = $foo if $max < $foo;
    }
    return $max;
}
$bestday = max($mon,$tue,$wed,$thu,$fri);
```

$a – small number in sort(); $b – large number in sort()

```
@all = sort { $b <=> $a } 4, 19, 8, 3;

@ordered = sort { $a->name cmp $b->name } @employees;
```

%ENV – environment variables
%INC – the paths to be searched
…

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Special Symbols – Regular Expression

$1, $2, … - matching groups in the parentheses in pattern

```
my $outer = 'Wallace and Grommit';
my $inner = 'Mutt and Jeff';

my $pattern = qr/(\S+) and (\S+)/;

sub show_n { print "\$1 is $1; \$2 is $2\n" }

{
OUTER:
    show_n() if $outer =~ m/$pattern/;

    INNER: {
        show_n() if $inner =~ m/$pattern/;
        }

    show_n();
}
```

Output:

```
$1 is Wallace; $2 is Grommit
$1 is Mutt; $2 is Jeff
$1 is Wallace; $2 is Grommit
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Special Symbols – Regular Expression (2)

- $&/${^MATCH} – last successful matching string
- $`/${^PREMATCH} – the string preceding the last matching string
- $'/${^POSTMATCH} – the string following the last matching string

```
local $_ = 'abcdefghi';
/def/;
print "$`:$&:$'\n";          # prints abc:def:ghi
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Special Symbols – File handlers

- $AGRV – name of current file
- @ARGV – command line arguments
- ARGV – special file handle for command line filenames
- $. – current line number
- $/ - input line delimiter
- $\ - output line delimiter
- $% - current page number

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Special Symbols – File handlers

- $@ Perl error string
- $! Error number from C, ‘errno’
- $^E Extended OS error info, such as ‘CDROM tray not closed’
- $? Exit status from last process

```
eval q{
            open my $pipe, "/cdrom/install |" or die $!;
            my @res = <$pipe>;
            close $pipe or die "bad pipe: $?, $!";
        };
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Code Examples

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Walk Through Code Examples

Examples To walk through: (code examples are in ./code/session1/)

1. bio_nts_trans.pl - example in real world to show regular expression in use

2. bio_prot_trans.pl – example in real world to show hash structure in use

Let's go to the terminal to go through these examples now.

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Packages and Modules

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Purpose of Packages/Modules

- To address the complicity of software functionality, when single script is not sufficient and clear to provide the service.

- It's a way to organize code

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

What is Package

- ‘package’ – the term used for functionality, means a division of global namespace; can be spread across several files (modules);
- It’s a logical unit for code functionality;
- Declares the BLOCK or the rest of the compilation unit as being in the given namespace (Perldoc definition)
- Package = Namespace (simplified)
- Way Perl uses to implement ‘class’ (object-oriented)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

What is Module

- ‘module’ – a library file consists of a set of related methods;
- It can be used as ‘class’ definition or class implementation , or both (for example: Bio::SeqIO)
- modules are actual physical libraries stored in file system to implement desired functioning system
- the common practice is to organize them by their logical namespaces (package)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Package vs Module - relationship

- Modern design of perl modules – one module one package
- object-oriented
  - hierarchically origanized, so outer namespace could cover the inner namespace, to provide modularity
  - Module file directory reflects namespace hierarchy
  - well defined interfaces between modules (namespaces);
  - Two Examples, Bio::DB and Bio::SeqIO

    Bio::DB – no common interface; every sub namespace is self-referenced

    Bio::SeqIO – has common abstract interface defined (implemented), while inside every sub namespace related to certain SeqIO may refer to this common interface

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

BioPerl on SCC

This is the first level file structure of BioPerl installed on SCC:

```
[yshen16@scc4 Bio]$ ls
Align                    CodonUsage           LiveSeq              Perl.pm          SearchIO.pm              Symbol
AlignIO                  Coordinate           LocatableSeq.pm      Phenotype        Seq                      Taxon.pm
AlignIO.pm               DB                   Location             PhyloNetwork     Seq.pm                   Taxonomy
AnalysisI.pm             DBLinkContainerI.pm  LocationI.pm         PhyloNetwork.pm  SeqAnalysisParserI.pm    Taxonomy.pm
AnalysisParserI.pm       Das                  Map                  PopGen           SeqEvolution             Tools
AnalysisResultI.pm       DasI.pm              MapIO                PrimarySeq.pm    SeqFeature               Tree
AnnotatableI.pm          DescribableI.pm      MapIO.pm             PrimarySeqI.pm   SeqFeatureI.pm           TreeIO
Annotation               Draw                 Matrix               PullParserI.pm   SeqI.pm                  TreeIO.pm
AnnotationCollectionI.pm Event                MolEvol              Range.pm         SeqIO                    UpdateableSeqI.pm
AnnotationI.pm           Factory              Nexml                RangeI.pm        SeqIO.pm                 Variation
Assembly                 FeatureHolderI.pm    NexmlIO.pm           Restriction      SeqUtils.pm              WebAgent.pm
Cluster                  HandlerBaseI.pm      Ontology             Root             SimpleAlign.pm
ClusterI.pm              IdCollectionI.pm     OntologyIO           Search           SimpleAnalysisI.pm
ClusterIO                IdentifiableI.pm     OntologyIO.pm        SearchDist.pm    Species.pm
ClusterIO.pm             Index                ParameterBaseI.pm    SearchIO         Structure
[yshen16@scc4 Bio]$
```

for full library structure, refer to : doc/bioperl_structure.txt

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Perl help system

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

Perl Language Reference

- This is the ultimate resource of authority – BLUEPRINT of a language;
- Access entrance:
  - http://perldoc.perl.org/index-language.html
- May be found too difficult to be understood for beginners

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

‘perldoc’ utility

- Embedded Perl documentation system in ‘POD’ (Plain Old Documentation) format

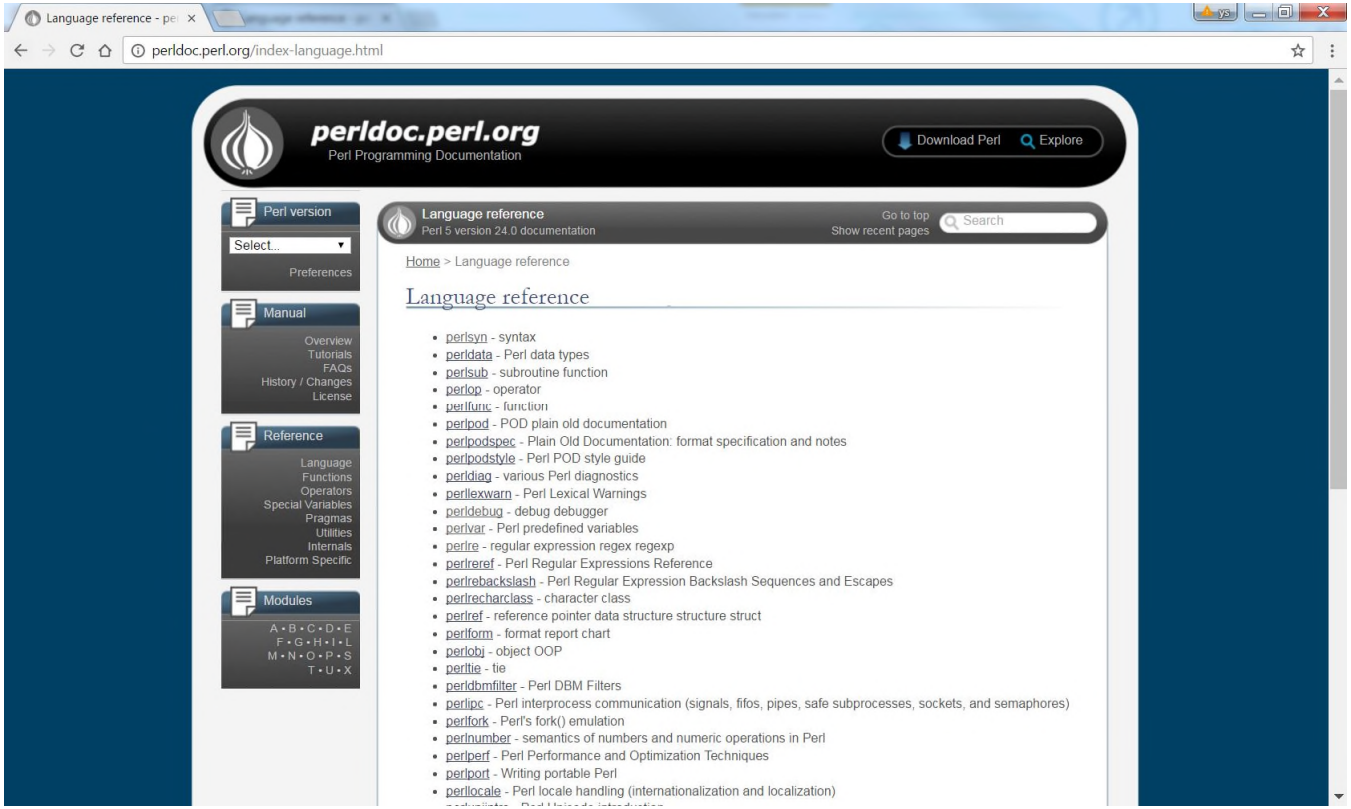
- Mostly written for Perl library modules:

```
perldoc perldoc # how to use perldoc
perldoc perlintro # perl introduction for beginners
perldoc perltoc # Perl table of contents
perldoc perl   # overview of Perl
perldoc perlfunc  # Full list of Perl functions
perldoc -f print  # help on built-in function called ‘print’
perldoc perlop # full list of perl operators
```

many more … (http://perldoc.perl.org/perl.html)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

http://perldoc.perl.org/index-language.html



BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

‘man’ command

- Linux ‘man’ command can be used to access perl module help, for example:

  man perl
  man perldoc
  man perltoc
  man perlre
  …

- ‘perldoc’ is recommended over ‘man’ – ‘man’ depends on if the man pages are installed for certain Perl Modules or not

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Get Help – online resources

Websites:
https://learn.perl.org/tutorials/
https://perlmaven.com/
http://perlmonks.org/
https://www.tutorialspoint.com/perl/
http://stackoverflow.com/

Books: (for more refer to perlbook_list.txt)
https://www.perl.org/books/beginning-perl/
http://docstore.mik.ua/orelly/perl/cookbook/

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Perl debugger

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018

perl -d

- Use ‘perl –d scriptname’ to start debugger

- Perl debugger is a fully integrated part to Perl interpreter, that means code must first pass the compiling process to be able to use debugger

- Frequently used debugger commands:

  h: type the help information
  n: execute next statement
  s: single step execution
  r: start/restart/continue run the code
  b: set breakpoints
  v: view source code in the context

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Data::Dumper

- Perl module commonly used to print out the variable structure and value; but more convenient

- Usage:

```
use Data::Dumper qw(Dumper);

print Dumper \@an_array;
print Dumper \%a_hash;
print Dumper $a_reference;
```


Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Data::Dumper Code Example

```
[yshen16@scc4 session1]$ more codeEx_useDumper.pl
#!/usr/local/bin/perl
use 5.010;
use strict;
use warnings;
use Data::Dumper qw(Dumper); # this is the custom module added for this particular purpose

# in Perl
#
my $a = 1;
my %b_hash = (
    1 => 'apple',
    2 => 'pearl',
    3 => 'orange',
);
my @c = sort keys %b_hash;

print "Here is the data in these variables:\n";
print Dumper $a;
print Dumper \%b_hash;
print Dumper \@c;
[yshen16@scc4 session1]$ perl codeEx_useDumper.pl
Here is the data in these variables:
$VAR1 = 1;
$VAR1 = {
          '1' => 'apple',
          '3' => 'orange',
          '2' => 'pearl'
        };
$VAR1 = [
          '1',
          '2',
          '3'
        ];
[yshen16@scc4 session1]$
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Q & A

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Evaluation Please @

http://scv.bu.edu/survey/tutorial_evaluation.html

Thank You !!

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Spring 2018