Web Server

Capacity Test
Design of Experiment

Prof. Domenico Cotroneo

Ing. Pietro Liguori

Objectives

1. Experimental Setup
   - Setup Server
   - Jmeter and httperf
   - Parameters collection
2. Capacity Test and Performance Analysis
3. Workload Characterization
4. Hypothesis Tests in MATLAB
5. Experimental Design and Analysis

3. Workload Characterization

Workload Characterization

- The goal of the ***workload characterization*** is to have a set of parameters whose ability to describe the load is as much application-independent as possible
- Key Requirements:
  - ***Comparability***: the parameters describing workload of different applications should be comparable
  - ***Basic Description***: application's behavior varies according to the applied workload, depending on the *amount* and *type* of work
  - ***Specialization Ability***: it should be possible to specialize the high-level application-independent parameters into application-dependent workload parameters
  - ***Realistic***: workload parameter values should fall inside realistic ranges, i.e., actually observed during operation
  - ***Practicability***: the number of workload parameters should be kept reasonably low

High Level Workload Characterization

- At high level, the load imposed to an application can be represented as a generic request of service, considered from the user point of view
- A request of service is characterized by the following parameters:
  - ***Intensity****:* Number of requests per second;
  - ***Size****:* Overall size of exchanged data in input/output;
  - ***Type of Requests****:* How many requests can be served by the application;
  - ***Variation of Requests****:* Given a request of type T, this parameter represents the probability that the next request will be of a type different from T

Application-dependent Workload Characterization

- Since the goal is to conduct real experiments, the high-level workload parameters need to be refined into application-dependent ones
- In this stage, the experimenter has to characterize the parameters with respect to the application(s) involved in the experimental campaign.

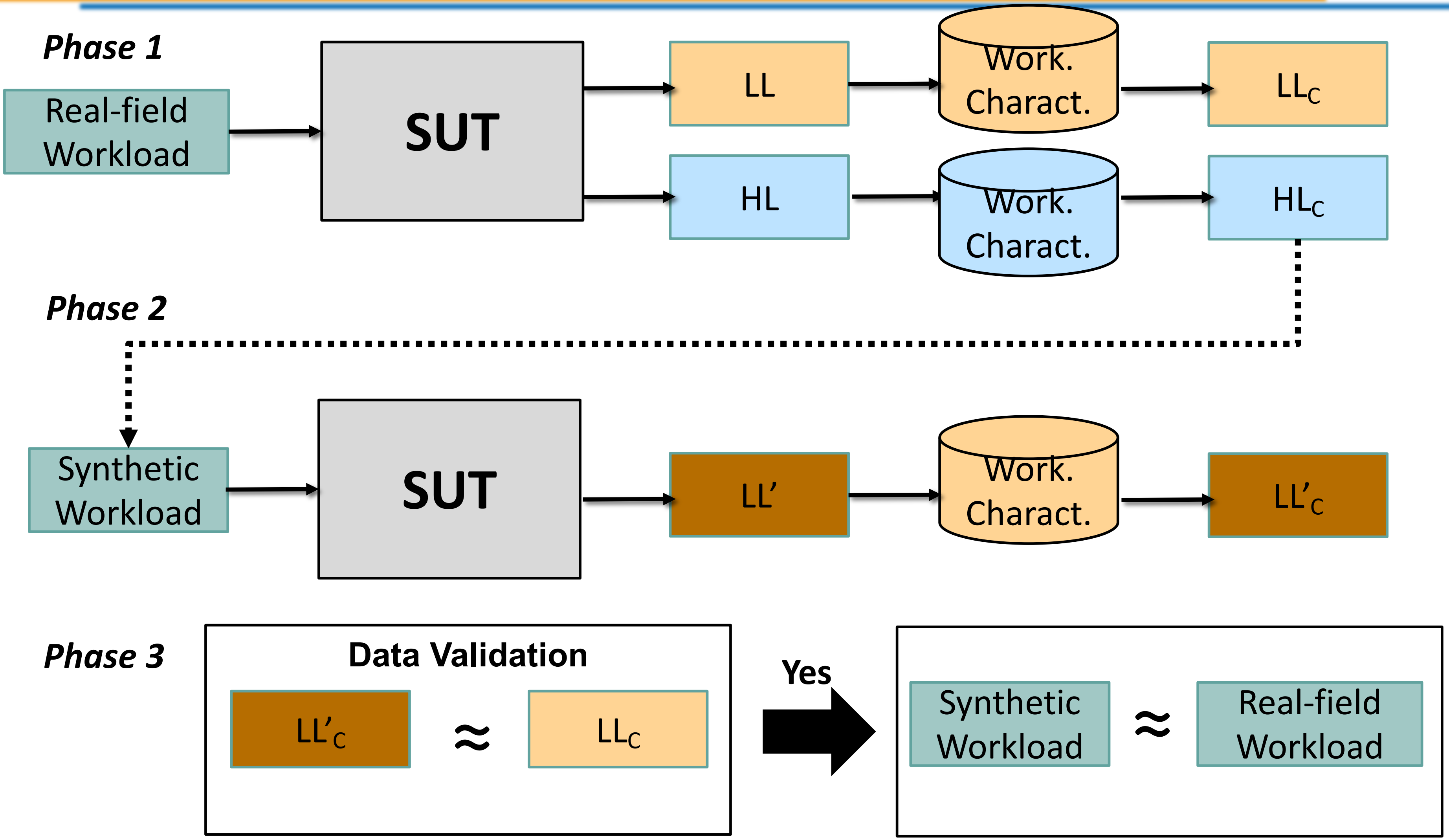
Bovenzi, A., Cotroneo, D., Pietrantuono, R., & Russo, S. (2011, November). ***Workload characterization for software aging analysis****. In 2011 IEEE 22nd International Symposium on Software Reliability Engineering (pp. 240-249). IEEE.*

Exercise

- Objective: Characterization of the observed workload

1. Generate a random workload (assuming it is a **real-field** workload)
   - Collect values of *application-level* (*high-level*) workload parameters
   - Collect values of *system-level* (*low-level*) workload parameters
   - Use techniques explained in the course to characterize the workload
2. Create **synthetic** workload
3. Validate the synthetic workload

Homework Overview



Generate the workload

- **Challenge**: Generate <u>your</u> real-field workload!
- For example, you can:
    - Create one or more thread-groups
    - Define the number of users (threads) per thread-group
    - Set the duration of the test (e.g., 5 min)
    - Set the ramp-up period
    - Use a constant throughput timer to set the request rate
    - Use a random controller to generate random requests

Real-filed workload configuration

- The configuration of the workload parameter depends on many factors:
  - the performance of the server
  - the number of concurrent threads
  - the number of requests
  - the request rate
  - the size of the resources
  - …
- Avoid error samples

Label	# Sam...	Average	Min	Max	Std. D...	Error %
HTTP ...	2130	38	4	304	39,53	0,00%
HTTP ...	2128	32	1	247	32,36	0,00%
HTTP ...	2123	22	0	148	25,56	0,00%
TOTAL	6381	31	0	304	33,72	0,00%

Requests

- Use *httpSampler*s to request different resources (web page, images, etc.) on the server
  - Different format
  - Static and or dynamic
  - Different sizes (e.g., small, medium, large)
  - Different resources for each category
  - …
- Use the different configuration of the the thread groups and the different resources to generate **several requests** (HTTP Request 1, HTTP Request 2, etc.)

Requests (cont.)

- **Example**: Suppose to have 3 thread groups and 3 different resource types
  - HTTP Request 1: Thread Group 1, Page Small
  - HTTP Request 2: Thread Group 1, Page Medium
  - HTTP Request 3: Thread Group 1, Page Large
  - HTTP Request 4: Thread Group 2, Page Small
  - HTTP Request 5: Thread Group 2, Page Medium
  - HTTP Request 6: Thread Group 2, Page Large
  - HTTP Request 7: Thread Group 3, Page Small
  - HTTP Request 8: Thread Group 3, Page Medium
  - HTTP Request 9: Thread Group 3, Page Large

Thread Groups concurrency

- If you use multiple Thread Groups, they will be executed in parallel
- In some scenarios it may cause dependency between thread groups
- To run your thread groups consecutively, enable the following option in the test plan panel

Collect Parameters

- Collect load-generator (**high-level**) parameters
  - Simply run the configured test
  - Results will be saved in the specified file (in the *SampleDataWriter* field)

- Collect sys-level (**low-level**) parameters
  - Using (one or more) utilities (e.g., *top, free,* ***vmstat****, iostat, ps*) collect the low-level parameters related to: memory, cpu, disk utilization, etc.
  - Run the utility on the server side while the test has been running and redirect output on a textual file

Collect Parameters (high-level)

- Collect load-generator (**high-level**) parameters
  - Simply run the configured test
  - Results will be saved in the specified file (in the *SampleDataWriter* field)

timeStamp	elapsed	label	responseCod	responseMe	threadName	dataType	success	failureMessa	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
1,64E+12	15	HTTP Reques	200	OK	Thread Group 1 1-4	text	true		512314	128	5	5	http://192.1	7	0	2
1,64E+12	29	HTTP Reques	200	OK	Thread Group 1 1-1	text	true		973114	128	7	7	http://192.1	6	0	2
1,64E+12	105	HTTP Reques	200	OK	Thread Group 1 1-3	text	true		358714	128	30	30	http://192.1	8	0	5
1,64E+12	10	HTTP Reques	200	OK	Thread Group 1 1-6	text	true		102714	128	8	8	http://192.1	6	0	3
1,64E+12	13	HTTP Reques	200	OK	Thread Group 1 1-7	text	true		409914	128	10	10	http://192.1	6	0	3
1,64E+12	22	HTTP Reques	200	OK	Thread Group 1 1-26	text	true		153914	128	30	30	http://192.1	12	0	3
1,64E+12	16	HTTP Reques	200	OK	Thread Group 1 1-8	text	true		358714	128	11	11	http://192.1	11	0	3
1,64E+12	44	HTTP Reques	200	OK	Thread Group 1 1-2	text	true		307514	128	11	11	http://192.1	10	0	3
1,64E+12	26	HTTP Reques	200	OK	Thread Group 1 1-10	text	true		921914	128	14	14	http://192.1	8	0	2
1,64E+12	30	HTTP Reques	200	OK	Thread Group 1 1-9	text	true		819514	128	15	15	http://192.1	9	0	4
1,64E+12	24	HTTP Reques	200	OK	Thread Group 1 1-13	text	true		205114	128	18	18	http://192.1	9	0	4
1,64E+12	19	HTTP Reques	200	OK	Thread Group 1 1-29	text	true		768314	128	30	30	http://192.1	11	0	2
1,64E+12	17	HTTP Reques	200	OK	Thread Group 1 1-30	text	true		665914	128	30	30	http://192.1	8	0	2
1,64E+12	19	HTTP Reques	200	OK	Thread Group 1 1-15	text	true		614714	128	19	19	http://192.1	10	0	4
1,64E+12	41	HTTP Reques	200	OK	Thread Group 1 1-11	text	true		870714	128	19	19	http://192.1	7	0	4
1,64E+12	21	HTTP Reques	200	OK	Thread Group 1 1-17	text	true		51512	127	22	22	http://192.1	20	0	2

Collect Parameters (low-level)

- Collect sys-level (**low-level**) parameters
  - Using (one or more) utilities (e.g., *top, free,* ***vmstat****, iostat, ps*) collect the low-level parameters related to: memory, cpu, disk utilization, etc.
  - Run the utility on the server side while the test has been running and redirect output on a textual file

vmstat –n 1 300 > output_file

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
2	0	124808	159084	42760	740548	3	19	180	132	135	367	3	2	95	1	0
0	0	124808	159076	42760	740552	0	0	0	0	70	214	1	0	99	0	0
0	0	124808	159076	42760	740552	0	0	0	0	68	160	1	0	99	0	0
0	0	124808	159076	42760	740552	0	0	0	0	45	112	0	0	100	0	0
0	0	124808	159076	42760	740552	0	0	0	0	142	208	1	0	99	0	0
0	0	124808	159076	42768	740552	0	0	0	32	757	423	1	11	88	0	0
0	0	124808	159076	42768	740556	0	0	0	20	119	183	0	1	99	0	0
1	0	124808	159076	42768	740556	0	0	0	0	65	151	0	0	100	0	0
0	0	124808	159076	42768	740556	0	0	0	0	72	166	0	1	99	0	0
0	0	124808	159076	42768	740556	0	0	0	0	51	125	1	0	99	0	0
0	0	124808	159076	42768	740556	0	0	0	0	152	234	2	1	97	0	0
0	0	124808	159076	42776	740556	0	0	0	32	633	403	1	8	90	1	0
0	0	124808	159076	42776	740560	0	0	0	0	118	114	0	1	99	0	0
0	0	124808	159076	42776	740560	0	0	0	0	41	77	0	0	100	0	0
0	0	124808	159076	42776	740560	0	0	0	0	48	101	0	0	100	0	0
0	0	124808	159076	42776	740560	0	0	0	0	34	80	0	0	100	0	0
0	0	124808	159076	42776	740560	0	0	0	0	104	119	0	1	99	0	0
0	0	124808	159076	42784	740560	0	0	0	16	652	371	0	13	87	0	0

Workload Characterization

- Characterize the **high-level** WL by using the parameters collected with the JMeter listener
  - Import the csv file in JMP
  - Use **PCA** to reduce dimensionality
  - Apply the **clustering** to reduce samples to consider

- Perform the workload characterization also on the **low-level** WL by using the parameters collected with vmstat (or any other utility)
  - You can create a csv file with the low-level parameters

PCA & Clustering

R	S	T	U	V	W	X	Y	Z	AA
Principale1	**Principale2**	**Principale3**	**Principale4**	**Principale5**	**Principale6**	**Principale7**	**Principale8**	**Principale9**	**Cluster**
1,694216982	-1,167911674	2,322677457	12,38035214	5,492631608	4,631301151	-6,27747507	7,056223279	3,911911301	1
0,474055045	-1,328356994	-1,311107268	0,903696863	-0,664277677	-0,548793963	0,201366748	0,200629082	-0,446076035	2
0,420353307	-1,28185054	-1,367460057	0,93075978	-0,702784995	-0,53211517	0,18394217	0,150299838	-0,374465898	2
0,169575889	-1,259354908	-1,72172783	0,836781899	-0,654129478	-0,370701669	0,111290737	-0,101931684	-0,081758054	2
0,578363851	-1,423271191	-1,236547854	0,822101723	-0,586451229	-0,528677091	0,204232925	0,180963501	-0,418970781	2
2,825087789	-2,871650723	0,536634645	-0,706101984	0,840849461	0,335021034	0,233032675	-0,751017048	1,075639841	4
0,429965704	-1,434646699	-1,426305311	0,723239293	-0,443734651	-0,112976565	0,314547814	-0,18248004	-0,028177946	2
0,135177993	-1,075244022	-1,326070987	0,741338719	-0,657660795	-0,287905319	0,038339606	-0,113700391	-0,067717248	2
0,345431495	-1,361234719	-1,554436911	0,688168822	-0,520293112	-0,343645494	0,095515719	-0,135488459	0,004629699	2
0,343094057	-1,209888747	-1,408014474	0,951978401	-0,728855314	-0,525822594	0,174937178	0,126510548	-0,331222192	2
0,872638821	-1,430663922	-0,835912864	0,849114328	-0,597739435	-0,648964011	0,234113322	0,336050516	-0,534365295	2
2,325386962	-2,50727947	0,173244449	-0,336844532	0,554970659	0,198045001	0,336539075	-0,485420464	0,619450207	4
0,348421832	-1,362806444	-1,541526793	0,643370604	-0,486149461	-0,316011422	0,085256109	-0,187521204	0,08728772	2
0,093691579	-1,185350407	-1,734190607	0,828054438	-0,648680179	-0,361668206	0,106483043	-0,122482749	-0,034074288	2
0,126890036	-1,214488384	-1,702306236	0,808867833	-0,624618466	-0,367535616	0,114633644	-0,101305026	-0,064280752	2
0,085817098	-1,178079913	-1,73901764	0,834880998	-0,654625072	-0,364393075	0,106978239	-0,118300987	-0,039938027	2
0,331734943	-1,347455548	-1,552182983	0,657481504	-0,498710241	-0,321142928	0,085926286	-0,18009692	0,076896176	2
2,615856043	-2,778133	0,205719191	-0,96504975	0,918040239	0,37244227	-0,084446465	-1,106015565	1,672735963	4

Synthetic Workload

- Use the characterization of the high-level workload to generate a **synthetic** workload
  - Use this workload to perform a new set of requests to the server (set the same duration of the real workload)
  - Collect the sys-level parameters (e.g., vmstat)

label	req/min	bytes	grpThreac	Cluster
1	30	139410	15	1
1	60	139409	15	2
3	30	1661191	15	3
3	60	1661192	15	4
1	30	139410	30	5
1	60	139409	25	6
1	60	139410	30	7
3	30	1661192	30	8
3	60	1661192	30	9

Example of synthetic workload

Synthetic Workload

	label					
Conteggio % del totale % di colonne % di righe	HTTP Request 1	HTTP Request 2	HTTP Request 3	HTTP Request 4	HTTP Request 5	Totale
Cluster 1	120 24,59 100,00 75,47	29 5,94 24,37 18,24	0 0,00 0,00 0,00	10 2,05 12,50 6,29	0 0,00 0,00 0,00	159 32,58
Cluster 2	0 0,00 0,00 0,00	90 18,44 75,63 56,25	0 0,00 0,00 0,00	70 14,34 87,50 43,75	0 0,00 0,00 0,00	160 32,79
Cluster 3	0 0,00 0,00 0,00	0 0,00 0,00 0,00	90 18,44 100,00 53,25	0 0,00 0,00 0,00	79 16,19 100,00 46,75	169 34,63
Totale	120 24,59	119 24,39	90 18,44	80 16,39	79 16,19	488

- **HTTP 1: TG1, Small**
- **HTTP 2: TG1, Medium**
- **HTTP 3: TG1, Large**
- **HTTP 4: TG2, Medium**
- **HTTP 5: TG2, Large**

JMP
Fit Y by X
Y: Label
X: Cluster

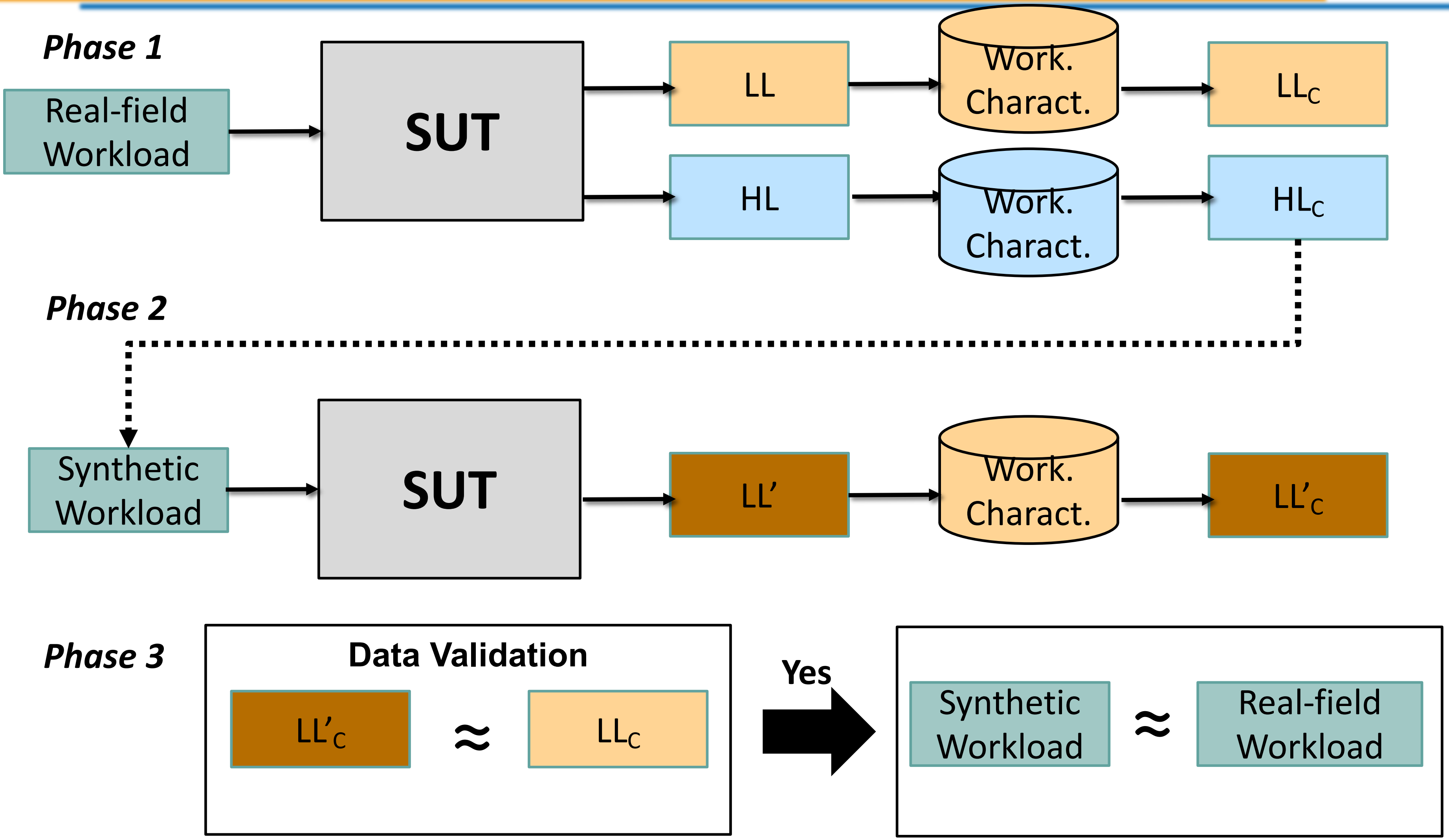
- To create the synthetic workload, choose a request representative for each cluster, and discard the other ones
- E.g., we keep the HTTP Request 1, 2 and 3 and discard the request HTTP Request 4 and 5

Data Validation

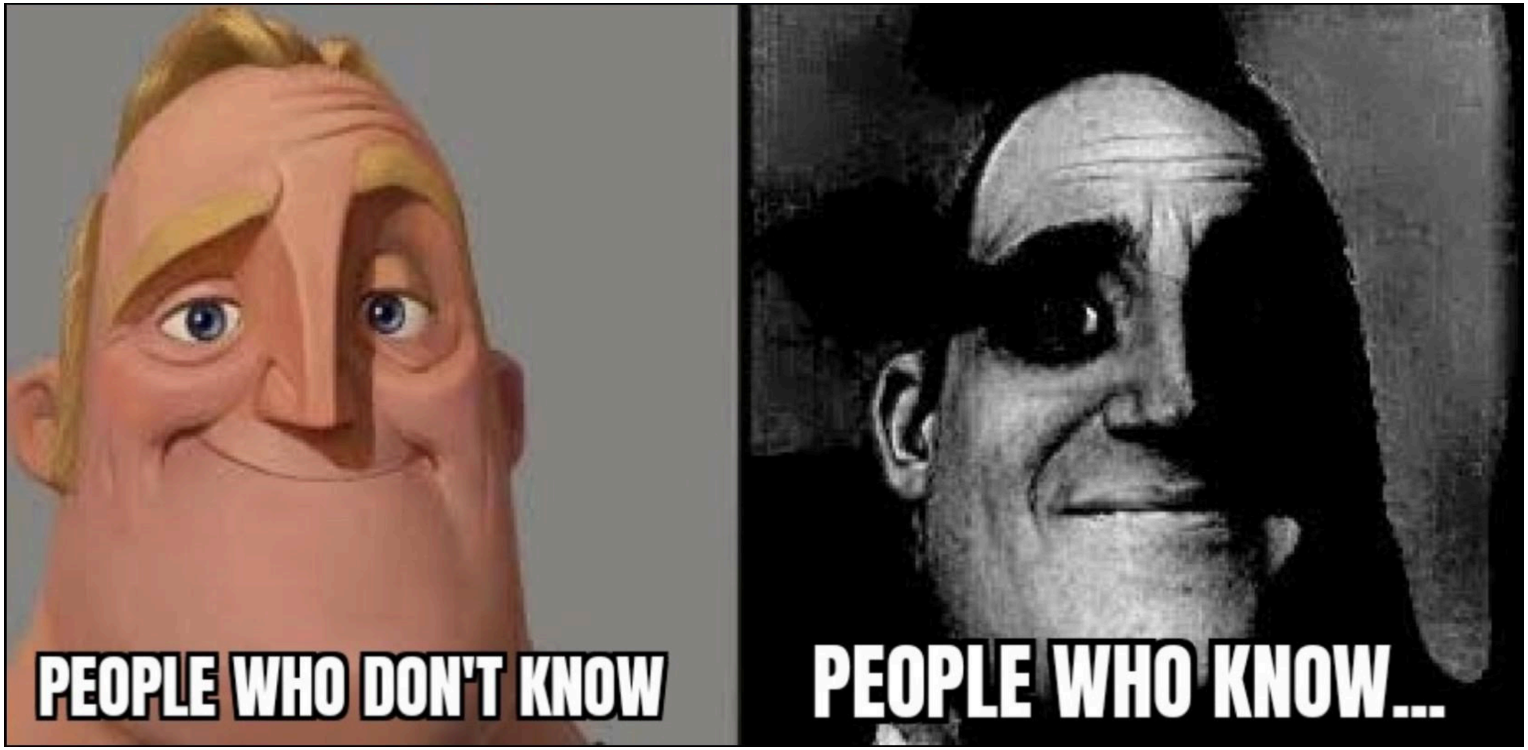
- Characterize the low-level parameters of the synthetic workload
  - PCA & Clustering: use the same number of components and clustering set in the real (low-level) workload characterization (in order to perform a comparison)

- Verify that the synthetic (low-level) workload LL'_C is representative of the real (low-level) workload LL_C with a **hypothesis test**
  - H_0: no significant difference between specified populations
  - Is the null hypothesis H_0 rejected?

Homework Overview (again)



What hypothesis test?



3. Hypothesis Tests (MATLAB)

Testing the means

- **One sample hypothesis test**: $H_0\colon \mu = \mu_0$
  - e.g., zero-mean test: testing whether the mean is significantly different form zero ($H_0\colon \mu_0 = 0$)
- **Paired Observations**: $H_0\colon \mu_1 = \mu_2$
  - *n* units in each sample such that there is a one-to-one correspondence between the i^{th} unit of sample A and the i^{th} unit of sample B
  - e.g., the same subject measured twice (the samples are not independent)
- **Two-sample (unpaired) Observations:** $H_0\colon \mu_1 = \mu_2$
  - No correspondence. Two sample of size n_A and n_B are available

One sample t-test

h = ttest(x,m)

- **Null Hypothesis H_0**: It returns a test decision for the null hypothesis that the data in x comes from a normal distribution with mean m and unknown variance.
- **Alternative Hypothesis H_1**: the alternative hypothesis is that the mean is not m
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

Assumptions

- The data values are continuous
- The data values are independent
- The data samples have been randomly sampled from a population
- The population from which we are collecting our data samples is normally distributed

z-test

h = ztest(x,m,sigma)

- **Null Hypothesis H_0**: It returns a test decision for the null hypothesis that the data in the vector x comes from a normal distribution with mean m and a standard deviation sigma
- **Alternative Hypothesis H_1**: The alternative hypothesis is that the mean is not m.
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

Example: Energy Bar

- protein = [20.70, 27.46, 22.15, 19.85, 21.29, 24.75, 20.75, 22.91, 25.34, 20.33, 21.54, 21.08, 22.14, 19.56, 21.10, 18.04, 24.12, 19.95, 19.72, 18.28, 16.26, 17.46, 20.53, 22.12, 25.06, 22.44, 19.08, 19.88, 21.39, 22.33, 25.79]

[h, p] = ttest(protein, 20)

h = 1

p = 0.0046

Example: Energy Bar

Is the *t*-test an appropriate method to test that the energy bars have 20 grams of protein ?

- The data values are **independent**. The grams of protein in one energy bar do not depend on the grams in any other energy bar. An example of dependent values would be if you collected energy bars from a single production lot.
- The data values are grams of protein. The measurements are **continuous**.
- We assume the energy bars are a simple **random sample** from the population of energy bars available to the general consumer (i.e., a mix of lots of bars).

We decide that the *t*-test is an appropriate method.

Paired or two sample t-test?

- **The paired t-test** is used when data is in the form of matched pairs
  - We only require that the difference of each pair is <u>normally distributed</u>

h = ttest(x,y)

- **Two-sample t-test** is used when the data of two samples are statistically independent
  - We need to assume that the data from both samples are <u>normally distributed</u> and they have the <u>same variances</u>

h = ttest2(x,y)

Paired *t*-test

h = ttest(x,y)

- **Null Hypothesis H_0**: returns a test decision for the null hypothesis that the data in x – y comes from a normal distribution with mean equal to zero and unknown variance
- **Alternative Hypothesis H_1**: The alternative hypothesis is that the data in x - y does not have a mean equal to zero
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

Two-sample *t*-test

h = ttest2(x,y)

- **Null Hypothesis H_0**: It returns a test decision for the null hypothesis that the data in vectors x and y comes from independent random samples from normal distributions with equal means and equal but unknown variances.
- **Alternative Hypothesis H_1**: The alternative hypothesis is that the data in x and y comes from populations with unequal means
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

Example: Exam Scores

- exam1 = [63, 65, 56, 100, 88, 83, 77, 92, 90, 84, 68, 74, 87, 64, 71, 88];
- exam2 = [69, 65, 62, 91, 78, 87, 79, 88, 85, 92, 69, 81, 84, 75, 84, 82];

[h, p] = ttest(exam1, exam2)

h = 0

p = 0.4650

Example: Exam Scores

Is the paired t-test an appropriate method to evaluate the difference in difficulty between the two exams?

- Subjects are **independent**. Each student does their own work on the two exams
- Each of the **paired measurements** are obtained from the same subject. Each student takes both tests.
- We assume that the distribution of differences is **normally distributed**.

Example: Body fat percentage

- men = [13.3, 6.0, 20.0, 8.0, 14.0 19.0, 18.0, 25.0, 16.0, 24.0 15.0, 1.0, 15.0];
- women = [22.0, 16.0, 21.7, 21.0, 30.0, 26.0, 12.0, 23.2, 28.0, 23.0];

[h, p] = ttest2(men, women)

h = 1

p = 0.0107

Example: Body fat percentage

To conduct a valid test:

- Data values must be **independent** → Measurements for one observation do not affect measurements for any other observation
- Data in each group must be obtained via a **random sample** from the population
- Data in each group are **normally distributed**
- Data values are **continuous**
- The **variances** for the two independent groups are **equal**

Two-sample *t*-test without Assuming Equal Variances

```
h = ttest2(x,y,'Vartype','unequal')
```

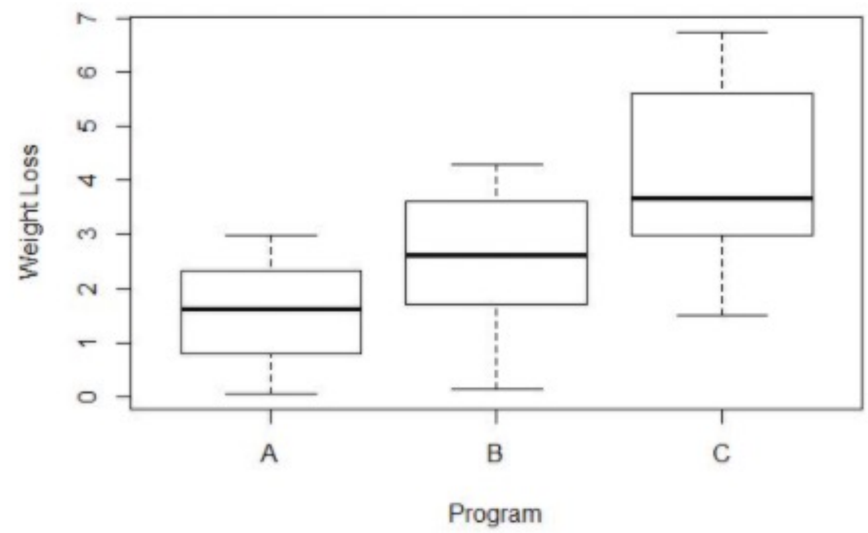
- Test the null hypothesis that the two data vectors are from populations with equal means, without assuming that the populations also have equal variances
- The returned value of h = 0 indicates that ttest2 does not reject the null hypothesis at the default 5% significance level even if equal variances are not assumed

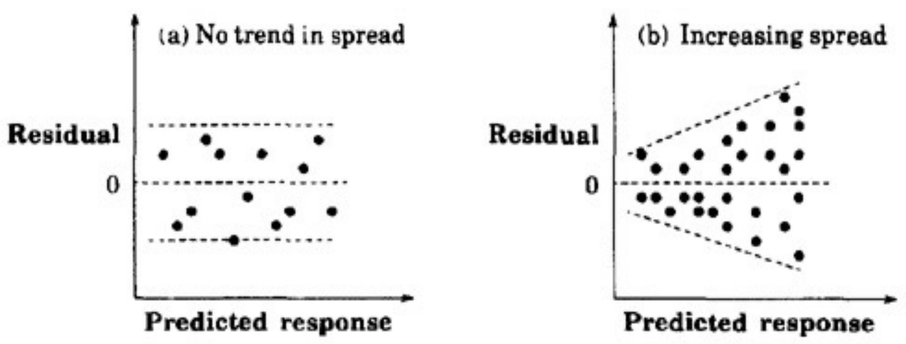
Assumption of Equal Variances: Homoscedasticity

- It assumes that different samples have the same variance, even if they came from different populations.
  - The assumption is found in many statistical tests, including Analysis of Variance (ANOVA) and Student's T-Test.
  - Other tests, like Welch's T-Test, don't require equal variances at all.
- Running a test without checking for equal variances can have a significant impact on your results and may even invalidate them completely

Equal Variances

- **Box Plot**: The longer the box, the higher the variance. For example, we can see that the variance is a bit higher for participants in program C compared to both program A and program B.
- **Scatter Plot of Errors versus Predicted Response**: If the spread in one part of the graph seems significantly different than that in other parts, then the assumption of constant variance is not valid.





Two-sample *F*-test for equal variances

h = vartest2(x,y)

- **Null Hypothesis H_0**: It returns a test decision for the null hypothesis that the data in vectors x and y comes from normal distributions with the same variance
- **Alternative Hypothesis H_1**: The alternative hypothesis is that they come from normal distributions with different variances
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

Parametric vs Non-parametric Tests

- All previous tests are based on assumptions
- **Parametric tests** are those that make assumptions about the parameters of the population distribution from which the sample is drawn.
  - This is often the assumption that the population data are ***normally distributed***
- **Non-parametric tests** are “distribution-free” and, as such, can be used for non-normal variables

Central Limit Theorem (CLT)

CLT or not CLT,
that is the question

Central Limit Theorem (CLT)

- *If you have a population with mean μ and standard deviation σ and take sufficiently large random samples from the population with replacement, then the distribution of the sample means will be approximately normally distributed*

1. The data must follow the randomization condition (it must be **sampled** randomly)
2. Samples should be **independent** of each other (one sample should not influence the other samples)
3. The sample size should be **sufficiently large** (a sample size of 30 is considered sufficient)

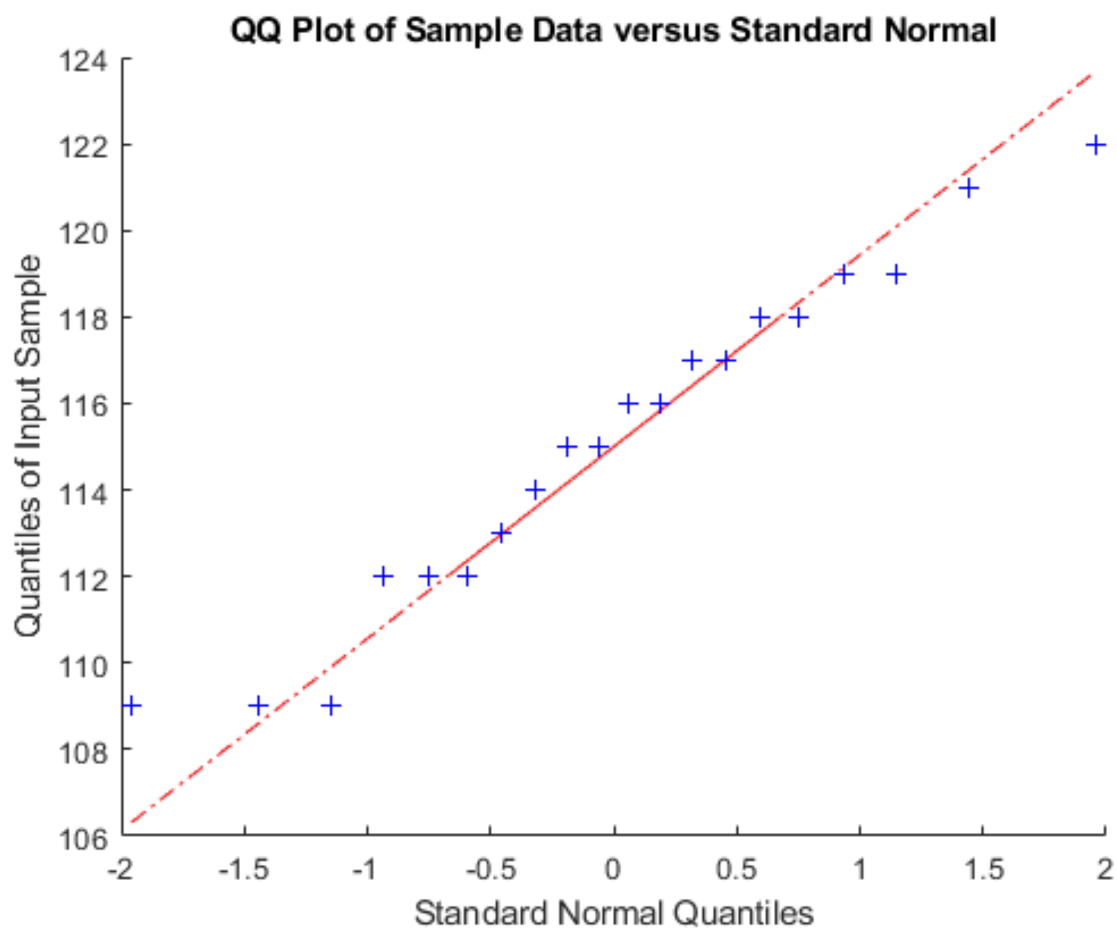
Parametric or Non-Parametric Test?

- Does our data come from a normal distribution?
- **Solution**(s):
  - **Visual Tests** (e.g., quantile-quantile plot)
  - Kolmogorov-Smirnov test (MATLAB)
  - Shapiro-Wilk (JMP)
  - KSL Test (JMP)
  - …
- **Answer**:
  - H_0 NOT rejected → Parametric Test
  - H_0 rejected → Non-parametric Test

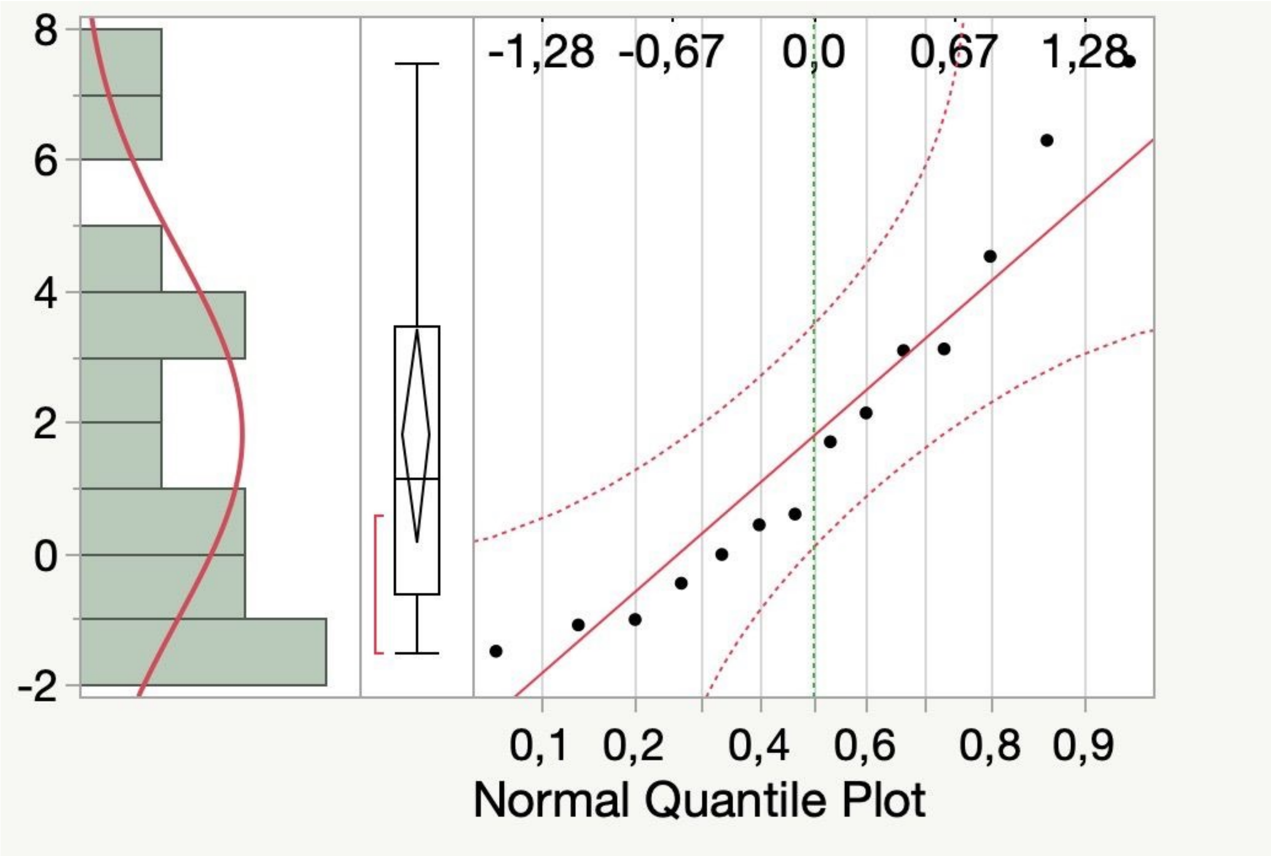
Visual Test

- Use a quantile-quantile plot to determine whether data follows a normal distribution

qqplot(x)



Visual Test (JMP)



Analyze → Distribution → Normal Quantile Plot

One-sample Kolmogorov-Smirnov test

h = kstest(x)

- **Null Hypothesis H_0**: data in vector x comes from a standard normal distribution
- **Alternative Hypothesis H_1**: data in vector x does not come from a standard normal distribution
- The result h is 1 if the test rejects the null hypothesis at the 5% significance level, or 0 otherwise.

Non-parametric test: Wilcoxon rank sum test

[p, h] = ranksum(x,y)

- It tests the null hypothesis that data in x and y are samples from continuous distributions with equal medians, against the alternative that they are not.
- The test assumes that the two samples are independent.
- x and y can have different lengths.
- p is the *p-value*
- h = 1 indicates a rejection of the null hypothesis, and h = 0 indicates a failure to reject the null hypothesis at the 5% significance level.

What hypothesis test?

Homework Data Validation: Step by Step

For every principal component:

- Check for normality of the data (LL'_C and LL_C) → the visual test is <u>preferred</u>
- ***If*** data is not normally distributed, then apply a non-parametric test (rank sum test)
- ***Else*** check for equal variances (visual test or vartest2)
    - ***if*** equal variances → two sample t-test
    - ***else*** → two sample t-test with unequal variances

What hypothesis test?