Stat 302
Statistical Software and Its Applications
SAS: Data I/O

Yen-Chi Chen

Department of Statistics, University of Washington

Autumn 2016

Getting Data Files

- Get the following data sets from the course web site
  - `patient.txt`: space separated data items, no header.
  - `patient.csv`: single sheet csv spread sheet, no header.
  - `patient_names.csv`: single sheet csv spread sheet, with header line giving variable names.
  - `ice.txt`: space separated data items, with header.
  - `student.txt`: space separated data items, no header.
- Save them to a `data` folder that you create on the UDrive `U:\data` on the terminal server.
- Other data file formats, e.g., fixed column format, are possible, but we won't deal with them here.
  Consult *Learning SAS by Example* by Ron Cody
  on other data formats.

Formatting Raw Data in Text Format

- We need to take a few steps to format our data before reading them via the data step.
  - Use a text editor (Notepad) to make any (global) changes on delimiters and missing values.
  - The SAS default delimter is a blank " ", or several blanks between individual data items.
  - Header Rows: If you have a header row, you can skip it by using `infile "U:\data\patient.txt" firstobs=2;` and specify the header names as well as the data type explicitly in the data step.
  - There are ways to read the header rows to get the name of variables.
  - Missing Values: We must find any missing values or NA's and convert them to a period "." for SAS to recognize as such.
  - The period must be be separated from other values by one or more spaces.
  - Separate adjacent missing values by spaces as well.

Reading Data Values Separated by Spaces

- Read and print the data in `patient.txt` to the screen.

```
data patient1; * data set name;
    infile "U:\data\patient.txt";
    input ID Age Sex $;
    run;
title "Patient DATA 1";
proc print data= patient1;
run;
```

- When you have I/O questions, experiment with the feature in question on some small data set.

Reading Data Values Separated by Comma (.csv files)

- Use the `dsd` (Delimiter-Sensitive Data) option in `infile`.
- Read and print the data in `patient.csv` to the screen.

```
data patient2; * data set name;
    infile "U:\data\patient.csv" dsd;
    input ID Age Sex $; run;
title "Patient DATA 2";
proc print data= patient2; run;
```

- Changes default delimiter to a comma.
- Assumes missing values for empty slots.
  No need for periods to indicate missing values.
- Character values in quotes have the quotes stripped off.
- For a file `fname.txt` with other delimiters like ":" use
  `infile "U:\data\fname.txt" dsd dlm= ':';`
  instead.

Reading Data: Using the Import Wizard

- First create a folder with name `U:\My SAS Files` on the UDrive, if it does not yet exist there.
- On the SAS Tool Bar ⇒ File ⇒ Import Data ...
- `Select a data source from the list below`, choose `Comma Separated Values (*.csv)` ⇒ Next
- Navigate to the file from which you want to import data. Via `Browse...` open `U:\data\patient_names.csv` ⇒ Open
- Under options check `Get variable names from first row` and at First row of data, enter `2` ⇒ OK ⇒ Next
- At Library take WORK, at Member enter `PATIENT3` ⇒ Next
- Browse to the directory where you want the generated SAS import statement saved and specify its file name, `U:\My SAS Files\patient3.sas` ⇒ Finish.

What Has Happened?

- It imported the data set to the `WORK` folder. You can view it by ⇒ View ⇒ Explorer ⇒ Work and double clicking Patient3.
- It also saved the following commands in `U:\My SAS Files\patient3.sas` They can be used in future SAS programs for importing this data set for use with other procs.

```
PROC IMPORT OUT= WORK.PATIENT3
            DATAFILE= "U:\data\patient_names.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;
```

- I won't elaborate on `PROC IMPORT` used in place of `data`.
- To this we can add the following `proc print` commands to print out the data as in our two previous examples.

```
title "Patient DATA 3";
proc print data= patient3; run;
```

The Need for Permanent SAS Data Sets

- SAS procs only work on SAS data sets,
  which are created with the data input step.
- They are temporarily stored in the WORK library folder.
- After a SAS session closes these data sets are gone.
  They need to be recreated for each new SAS session.
- This would require another data input step.
- No big deal for small data sets, but for large ones it would be preferable to have a SAS data set from the start.

How to Create Permanent SAS Data Sets

```
libname mydata "U:\data"; *an existing location;
data mydata.patient4;
   infile "U:\data\patient.csv" dsd;
   input ID Age Sex $ ;
run;
title "Patient Data 4";
proc print data=mydata.patient4;
run;
```

- These lines create the permanent SAS data set `patient4` `U:\data\patient4.sas7bdat`.
- That data set also appears in the temporary Library folder `Mydata`. `Mydata` disappears after the end of a SAS session.
- Instead of the libref `mydata` you can use any other proper SAS name with ≤ 8 characters.

Comments

- When you delete `U:\data\patient4.sas7bdat` it also disappears from the temporary Library folder `Mydata`.
- When you delete `patient4` from the temporary Library folder `Mydata` it also disappears from `U:\data`
- If you rename it to `U:\data\patient5.sas7bdat`, it also renames to `patient5` in `Mydata`, after stepping out and back into the `Mydata` library.
- In a later SAS session or in the same session you can access `patient4` by giving another libref statement, e.g., `libname mydata2 "U:\data";` and use `mydata2.patient4` wherever you used `mydata.patient4` before.
- View `mydata` or `mydata2` as conduits to `U:\data`, and whatever you do (delete or rename) w.r.t. any SAS data set in one it is also done in the other. Play around with this.

How to Use Permanent SAS Data Sets

- Prior to using a permanent data set, such as `patient4`, in a new SAS session,you need an appropriate libname statement, i.e., you need a conduit, e.g., in a new SAS session try

```
libname mydata "U:\data";
title "Patient Data 4";
proc print data=mydata.patient4;
run;
```

- SAS needs to know where to find a permanent SAS data set.
- Running simply the first line above, you can look at the data via `SAS Explorer` ⇒ `Libraries` ⇒ the newly created folder `Mydata` ⇒ double click `patient4`, which opens up VIEWTABLE on that file.

How Export Permanent SAS Data Sets to CSV File – 1

- The following code saves the permanent SAS data set `patient4.sas7bdat` in folder `U:\data` to a file `U:\data\odsexample.csv`

```
libname mydata 'U:\data';
ods csv file='U:\data\odsexample.csv';
proc print data=mydata.patient4 noobs; run;
ods csv close;
```

- ODS stands for Output Delivery System
- The ODS CSV opens the CSV file as an output destination.
- Close file with ODS CLOSE following PROC PRINT.

How Export Permanent SAS Data Sets to CSV File – 2

- Here is a method that outputs data without the name of variables.

```
libname mydata 'U:\data';
ods csv file='U:\data\odsexample.csv';
proc report data=mydata.patient4;
define _all_ / display ' ';
run;
ods csv close;
```

- `proc report` is pretty much the same as `proc print` but some defaults and arguments are different.

Summary Statistics – 1: `freq`

```
libname mydata "U:\data";
title "Gender Frequencies";
proc freq data=mydata.patient4;
   table Sex; run;
```

Gender Frequencies

The FREQ Procedure

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	5	50.00	5	50.00
M	5	50.00	10	100.00

- Without the variable after `table` you will get errors.

Summary Statistics – 2: means

```
libname mydata "U:\data";
title "Age Summary";
proc means data=mydata.patient4
     n mean std median clm alpha=.005;
   var Age; run;
```

Age Summary

The MEANS Procedure

Analysis Variable : Age					
N	Mean	Std Dev	Median	Lower 99.5% CL for Mean	Upper 99.5% CL for Mean
10	17.0000000	5.4160256	17.5000000	10.6807238	23.3192762

- Without `var Age;` get stats on all numeric variables.

Summary Statistics – 3 `sort` and `by`

```
libname mydata "U:\data";
title "Sorting by Sex";
proc sort data=mydata.patient4;
  by Sex; run;
```

- This sorts the SAS data set by Sex (also in its permanent location). Needed if you split analyses using `by` .
- See what happens when using `by Sex Age` and `by Age Sex`.

```
title "Summaries by Sex";
proc means data=mydata.patient4;
  var Age;
  by Sex; run;
  * first sort by Sex alone again,
    if you tried the above: by Age Sex;
```

Summary Statistics – 4

The MEANS Procedure

Sex=F

Analysis Variable : Age				
N	Mean	Std Dev	Minimum	Maximum
5	20.0000000	4.2426407	14.0000000	24.0000000

Sex=M

Analysis Variable : Age				
N	Mean	Std Dev	Minimum	Maximum
5	14.0000000	5.0497525	8.0000000	21.0000000

Data Manipulation – 1

```
data patient5;
infile "U:\data\patient.csv" dsd;
input ID Age Sex $;
if Sex = "M";
run;
title "Patient DATA 5";
proc print data= patient5; run;
```

Obs	ID	Age	Sex
1	31	12	M
2	99	17	M
3	75	8	M
4	54	12	M
5	74	21	M

Data Manipulation – 2

```
data patient5;
infile "U:\data\patient.csv" dsd;
input ID Age Sex $;
if ID <= 20 then IDgroup = "A";
if ID > 20 and ID<= 50 then IDgroup = "B";
if ID > 50 and ID<= 70 then IDgroup = "C";
if ID > 70 then IDgroup = "D";
run;
title "Patient DATA 5 with IDgroup";
proc print data= patient5; run;
```

Data Manipulation – 3

Obs	ID	Age	Sex	IDgroup
1	31	12	M	B
2	62	18	F	C
3	50	20	F	B
4	99	17	M	D
5	53	14	F	C
6	75	8	M	D
7	54	12	M	C
8	58	24	F	C
9	4	24	F	A
10	74	21	M	D

Data Manipulation – 4: `in (...)`

```
data patient5;
infile "U:\data\patient.csv" dsd;
input ID Age Sex $;
if ID <= 20 then IDgroup = "A";
if ID > 20 and ID<= 50 then IDgroup = "B";
if ID > 50 and ID<= 70 then IDgroup = "C";
if ID > 70 then IDgroup = "D";
if IDgroup in ("A", "B","C") then ID_lab = 1;
run;
title "Patient DATA 5 with IDgroup";
proc print data= patient5; run;
```

Data Manipulation – 5

Obs	ID	Age	Sex	IDgroup	ID_lab
1	31	12	M	B	1
2	62	18	F	C	1
3	50	20	F	B	1
4	99	17	M	D	.
5	53	14	F	C	1
6	75	8	M	D	.
7	54	12	M	C	1
8	58	24	F	C	1
9	4	24	F	A	1
10	74	21	M	D	.

Data Manipulation – 6: where

```
title "Patient DATA 5 with IDgroup = A or C";
proc print data= patient5;
where IDgroup in ("A", "C");
run;
```

Obs	ID	Age	Sex	IDgroup	ID_lab
2	62	18	F	C	1
5	53	14	F	C	1
7	54	12	M	C	1
8	58	24	F	C	1
9	4	24	F	A	1

Latent Heat for Fusion of Ice: Analysis using SAS

```
data ice;
   infile "U:\data\ice.txt" firstobs=2;
   input Heat Method $ ; run;
title "Latent Heat of Fusion of Ice";
proc print data=ice; run;
title "Latent Heat of Fusion of Ice,
  Testing H: mean=80 for Method A";
proc ttest data=ice H0=80;
 var Heat;
 where Method = "A"; run;
title "Latent Heat of Fusion of Ice,
  Testing Equality of Methods A & B";
proc ttest data = ice;
   class Method; * sorted by method first!;
   var heat; run;
```

Latent Heat for Fusion of Ice: Data, t-Test $H : \mu_A = 80$

Latent Heat of Fusion of Ice

Obs	Heat	Method
1	79.982	A
2	80.041	A
3	80.018	A
4	80.041	A
5	80.030	A
6	80.029	A
7	80.038	A
8	79.968	A
9	80.049	A
10	80.029	A
11	80.019	A
12	80.002	A
13	80.022	A
14	80.020	B
15	79.939	B
16	79.980	B
17	79.971	B
18	79.970	B
19	80.029	B
20	79.952	B
21	79.968	B

Latent Heat of Fusion of Ice, Testing H: mean=80 for Method A

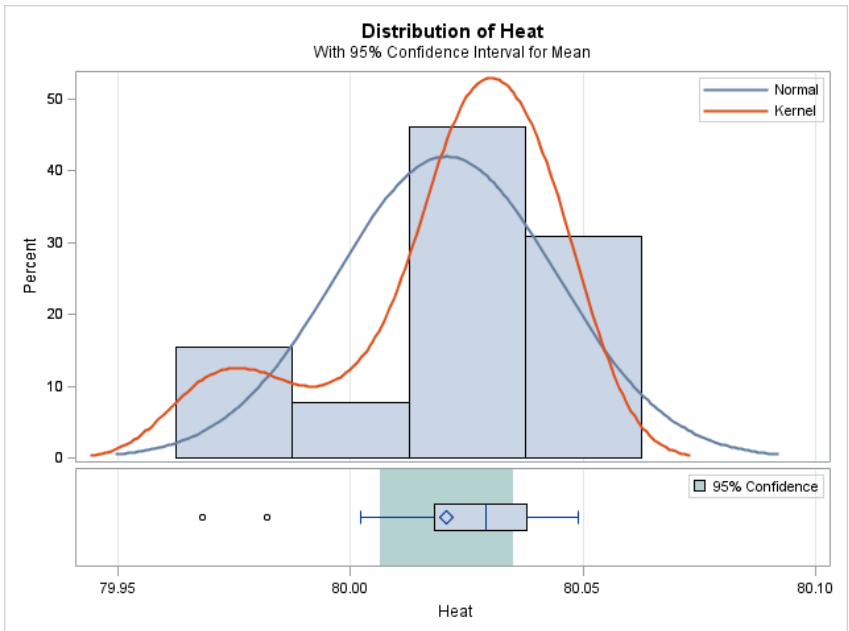
The TTEST Procedure

Variable: Heat

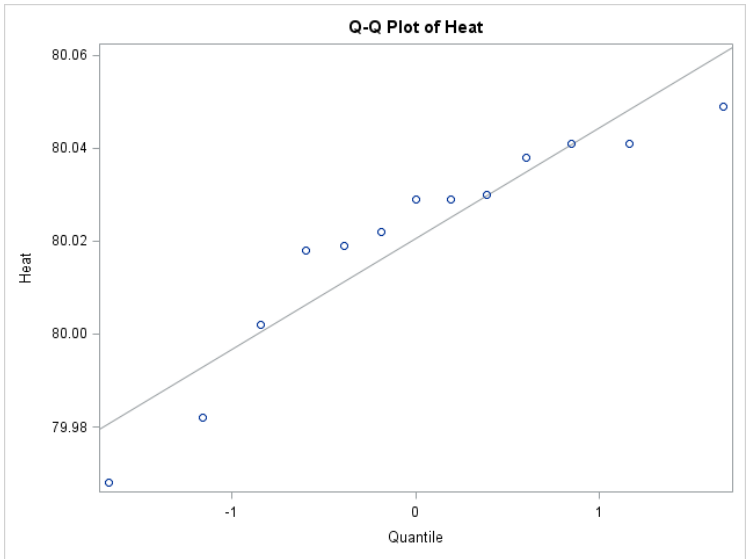
N	Mean	Std Dev	Std Err	Minimum	Maximum
13	80.0206	0.0238	0.00660	79.9680	80.0490

Mean	95% CL Mean		Std Dev	95% CL Std Dev	
80.0206	80.0062	80.0350	0.0238	0.0171	0.0393

DF	t Value	Pr > |t|
12	3.13	0.0088



Method "A" QQ-Plot & t-Test for $H : \mu_A = \mu_B$



Latent Heat of Fusion of Ice, Testing Equality of Methods A & B

The TTEST Procedure

Variable: Heat

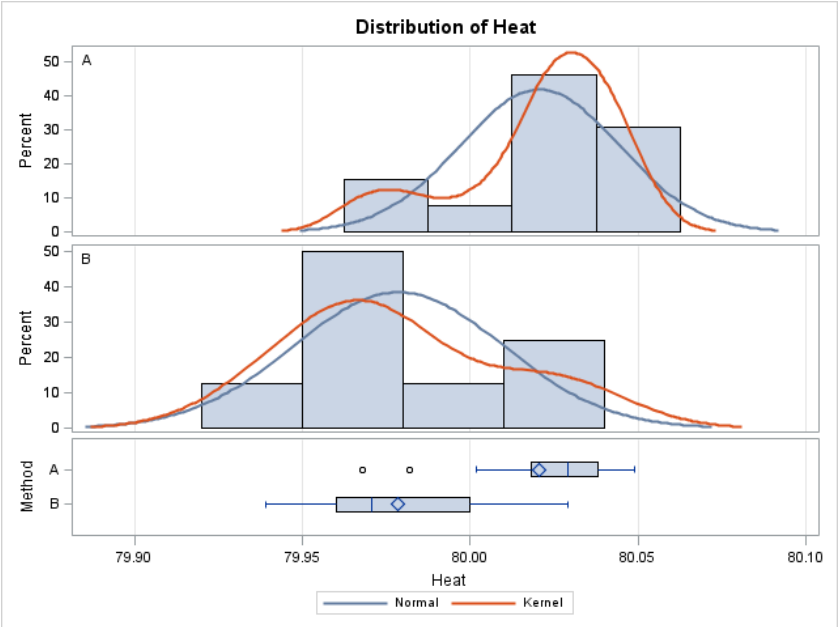
Method	N	Mean	Std Dev	Std Err	Minimum	Maximum
A	13	80.0206	0.0238	0.00660	79.9680	80.0490
B	8	79.9786	0.0311	0.0110	79.9390	80.0290
Diff (1-2)		0.0420	0.0267	0.0120		

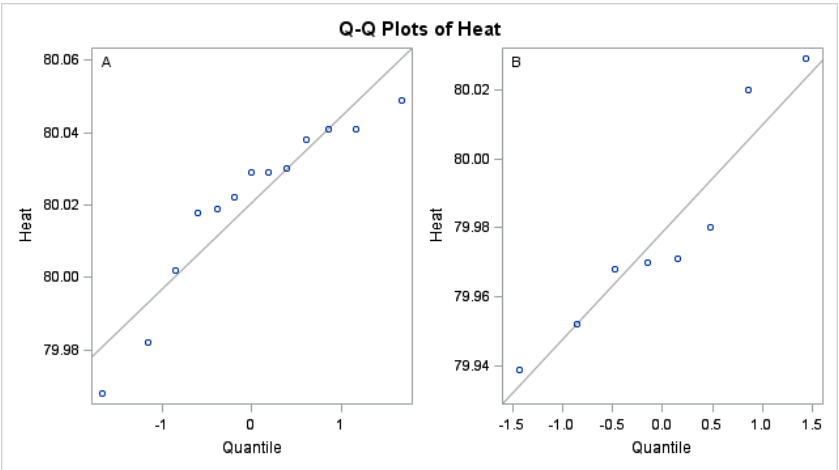
Method	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
A		80.0206	80.0062	80.0350	0.0238	0.0171	0.0393
B		79.9786	79.9526	80.0046	0.0311	0.0206	0.0633
Diff (1-2)	Pooled	0.0420	0.0169	0.0671	0.0267	0.0203	0.0390
Diff (1-2)	Satterthwaite	0.0420	0.0141	0.0699			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	19	3.50	0.0024
Satterthwaite	Unequal	12.03	3.27	0.0066

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	7	12	1.71	0.3943

Latent Heat for Fusion of Ice: 2 Sample t-test

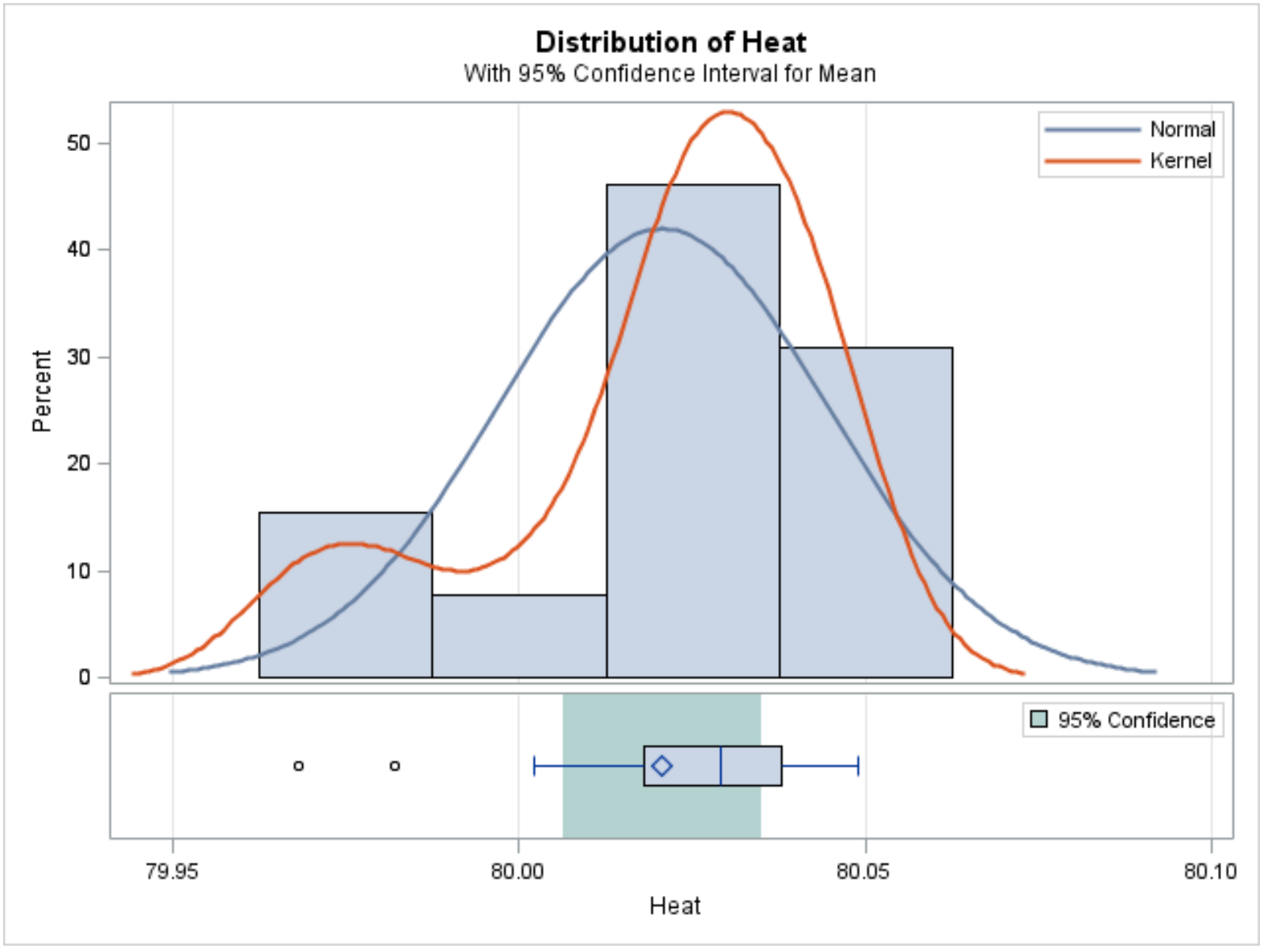




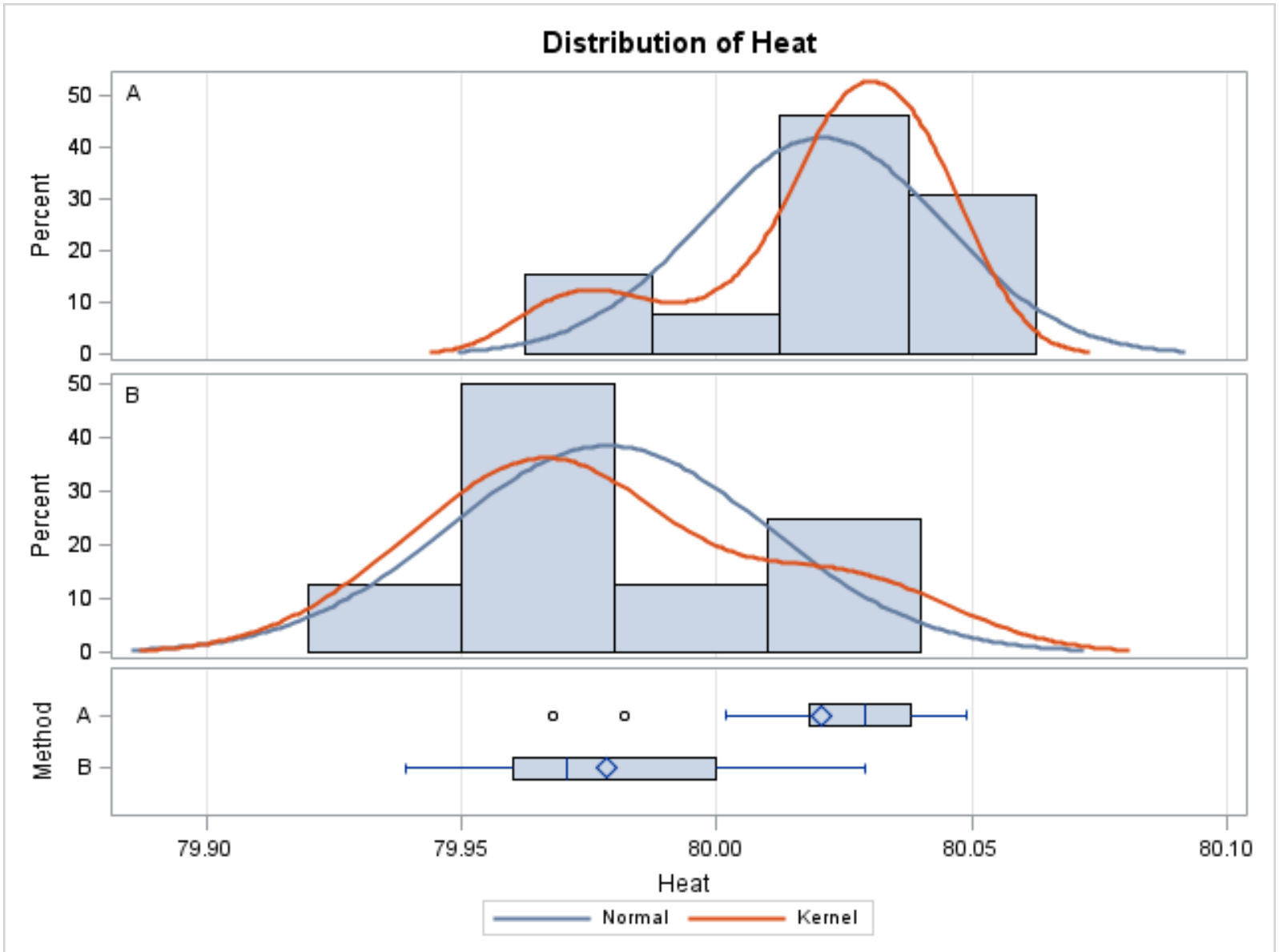
Comments on Output Illustrations

- In R you would use `t.test`.
- In this case SAS presents a whole bunch of pages as results, some in tabular form, some in the form of graphics.
- This is typical for packages like SAS. It is a package deal!
- The previous output illustrations were done by printing specific page pairs to PDF and including them via trim and clip parameters using `includegraphics` in LaTeX.
- For graphics output you can right click on the graphic and save it as a `.png` file, which you then include like any other graphic in your LaTeXfile, using `includegraphics`.
- Right clicking tabular output allows saving as Excel file.
- The next 3 slides show previous graphics via `.png` versions.

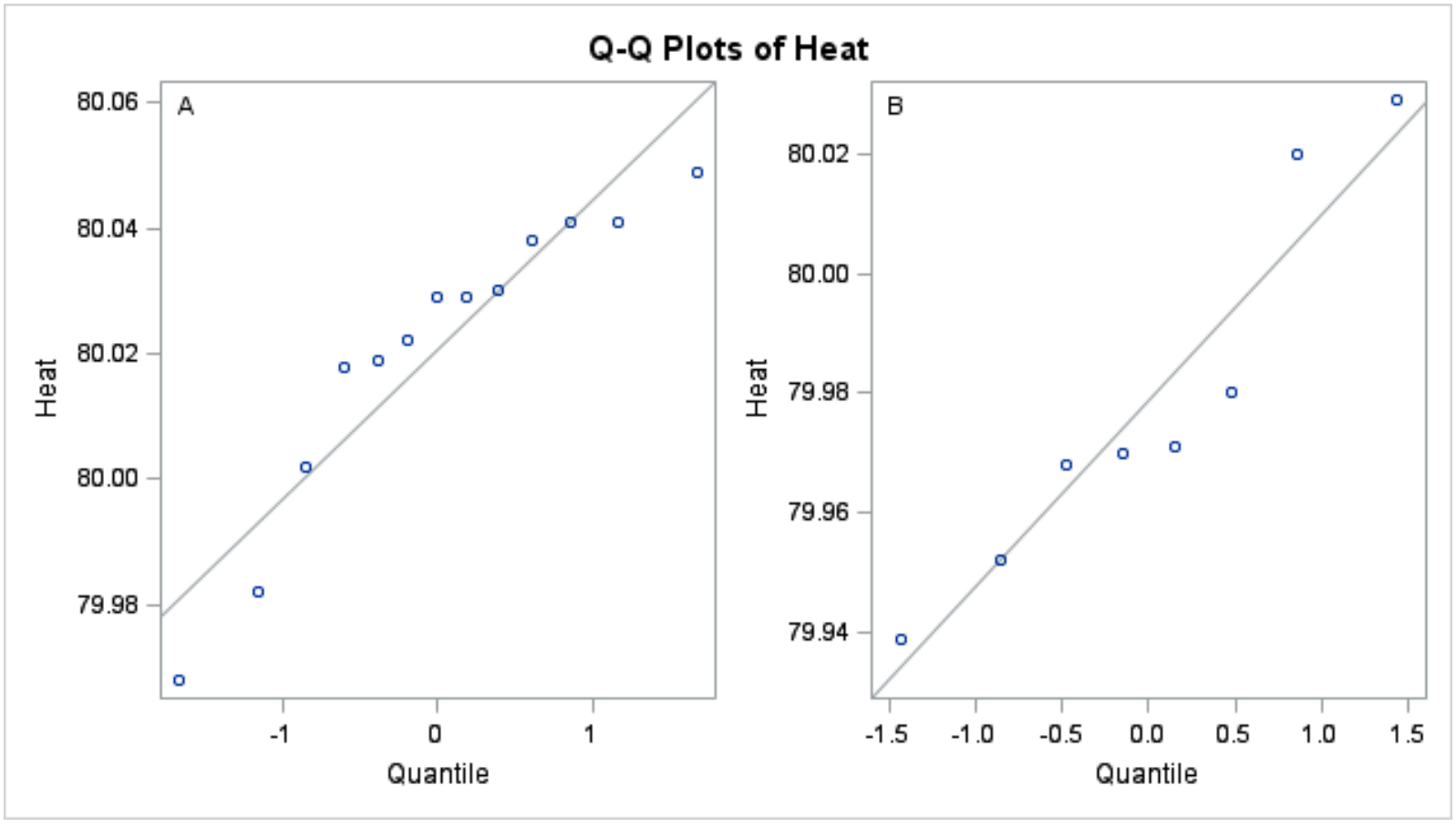
Latent Heat for Fusion of Ice: Data, t-Test $H_0 : \mu_A = 80$



2 Sample t-test for H_0 : "A" = "B"



Methods "A" and "B" QQ-Plots



SAS Procs

- There are a large number of SAS Procs.
- We have seen examples usages of FREQ, MEANS, SORT, and TTEST. Others of interest are: ANOVA, BOXPLOT, CORR, NPAR1WAY, PLOT, REG.
- Each such Proc has quite a few usage options.
- To access documentation with examples on these Procs click on SAS Procs under the next bullet.
- SAS Procs or search for SAS Procs in Google.

In-class Exercise

1. Now import the data `student.txt` into a data object called `student` with the three variables as `Age, Major, GPA`. Note that the variable `Major` is a character variable.
2. Use `proc freq` to obtain a frequency table of `Major`.
3. Use `proc means` to analyze variable `GPA`.
4. Sort the data by variable `Age` and print out the result.
5. Sort the data by variable `Major Age` and print out the result.
6. Use `proc means` to analyze variable `GPA` for each `Major`.
7. Create a new variable `Group` and for those students with `Major` being `Math` or `Stat`, assign their `Group` to be 1 otherwise the `Group` is 2.
8. Sort the data by variable `Group`, print out the data and then use `proc means` to analyze the variable `Age` for the two groups.
9. Output the data into a `.csv` file called `new_student.csv`.