STAT 8678 - SAS Programming & Data Analysis

Chi-Kuang Yeh

2026-02-12

Table of contents

Preface

Description

This course covers programming using the SAS statistical software package, and it provides an introduction to data analysis stressing the implementation using SAS.

Topics include two main parts:

1) **SAS Programming**: data management and manipulation, basic procedures, macro programming;
2) **Data Analysis**: descriptive statistical analysis, one- and two-sample inference, basic categorical data analysis, regression analysis, and other selected topics.

Prerequisites

MATH 4544/6544 – Biostatistics, or equivalent.

Instructor

Chi-Kuang Yeh, Assistant Professor in the Department of Mathematics and Statistics, Georgia State University.

- Office: Suite 1407, 25 Park Place.
- Email: cyeh@gsu.edu.

Office Hour

10:00–13:00 on Monday, or by appointment.

Grade Distribution

- Assignments: 60%
- Exam: 20%
- Project: 20%

Assignment

☒ A1, due on Jan 28, 2026
☒ A2, due on Feb 8, 2026
☐ A3, due on Feb 15, 2026

Midterm

☐ March 4, 2026

Topics and Corresponding Lectures

Those chapters are based on the lecture notes. This part will be updated frequently.

Status	Topic	Lecture
	Welcome and Overview	1
	Part 1: Basics	
	Basic SAS Operation	2
	SAS Syntax	3
	Import and Export Data	4
	Random Variable	5
	Part 2: Statistical Analysis	
	One Sample Problem as Example	
	Introduction to Statistical Inference I	6
	Introduction to Statistical Inference II	7
	One Sample Nonparametric Test	8
	One Sample Proportion Test	9
	Introduction to SAS Marco	10

Recommended Textbooks

- Statistics 480: Introduction to SAS, The Pennsylvania State University.
- SAS Training, SAS Institute.
- SAS Resources, University of California, Los Angeles.

Acknowledgments

Special thanks to Li-Hsiang Lin for providing the base materials given on this website.

Part I

Introduction

1 Introduction to Basic SAS Operation

Learning objective:

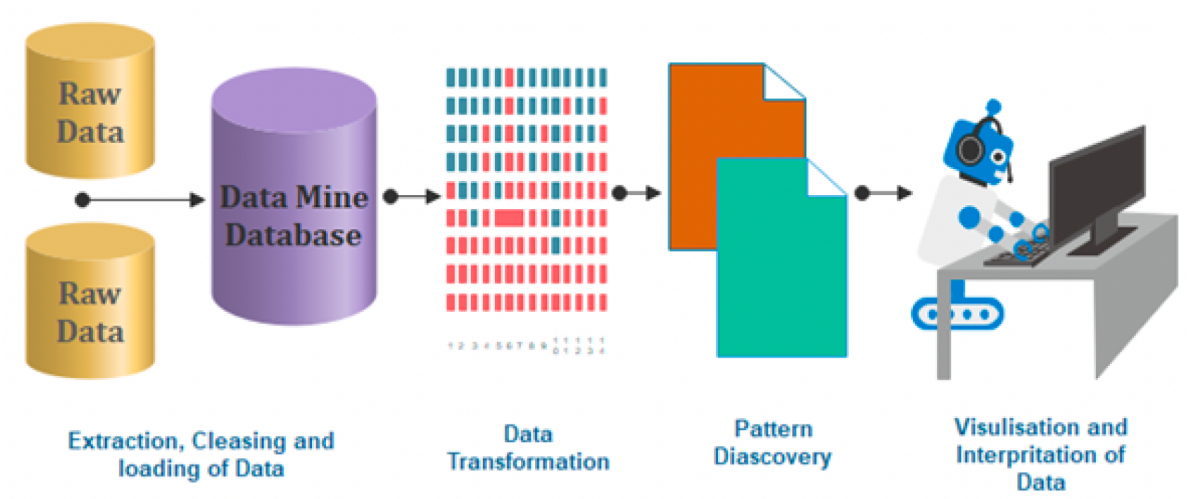
1. Familiarize ourselves with SAS windows (editor, log, output)
2. Create a dataset
3. Sorting Data (by 1 or more variables)
4. Obtain summary statistics of variables

1.1 Introduction to SAS

Q: What is SAS?

SAS (Statistical Analysis Software) is a prominent tool in the field of Data Analytics, offering a comprehensive suite for data manipulation, mining, management, and retrieval across various sources, coupled with robust statistical analysis capabilities. It excels in a range of functions including *data management, statistical analysis, report generation, business modelling, application development, and data warehousing.* SAS is user-friendly, featuring a point-and-click interface for those without technical expertise, while also providing deeper functionality through the SAS programming language. This software is instrumental in employing qualitative methods and processes that enhance employee productivity and business profitability.

Within SAS, data extraction and categorization into tables are pivotal for identifying and understanding data trends. This versatile suite supports advanced analytics, business intelligence, predictive analysis, and data management, facilitating effective operation in dynamic and compet- itive business environments. Additionally, SAS's platform-independent nature allows it to operate seamlessly across various operating systems, including Linux, Windows, Mac, and Ubuntu. SAS provides extensive support to programmatically transform and analyze data in the comparison of drag and drop interface of other Business Intelligence tools. It provides very fine control over data manipulation and analysis.



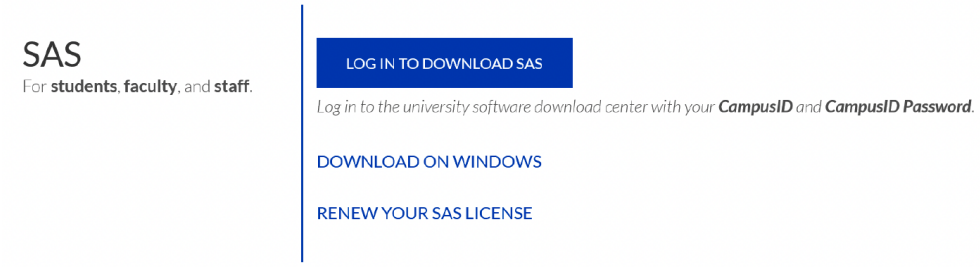
1.1.1 SAS Installzation

Georgia State University (GSU) has purchased license, so we can access SAS University Edition for free!

To install SAS University Edition, choose from the following options:

- **Option 1**:

  Download on your personal PC: Free SAS license available to GSU students, faculty, and staff via Technology Services (download required; check system requirements): Download from https://technology.gsu.edu/technology-services/software-equipment/university-licensed-software/ (Need to log-in from your GSU Account)



`Get Help for the Installation from <https://gsutech.service-now.com/sp>`

- **Option 2**:

- On Campus Access: SAS can be found on all GSU Library PCs: Floors 1-4 (not available on Library Macs, because there is no Mac version of SAS)
- Graduate Biostatistics Computer Lab (SPH): 6th floor of the Urban Life building (swipe card access required)
- Common MILE Lab whose opening time is
  - Monday & Wednesday: 9 – 18
  - Tuesday & Thursday: 9 – 17
  - Friday: 9 – 15

- **Option 3**:

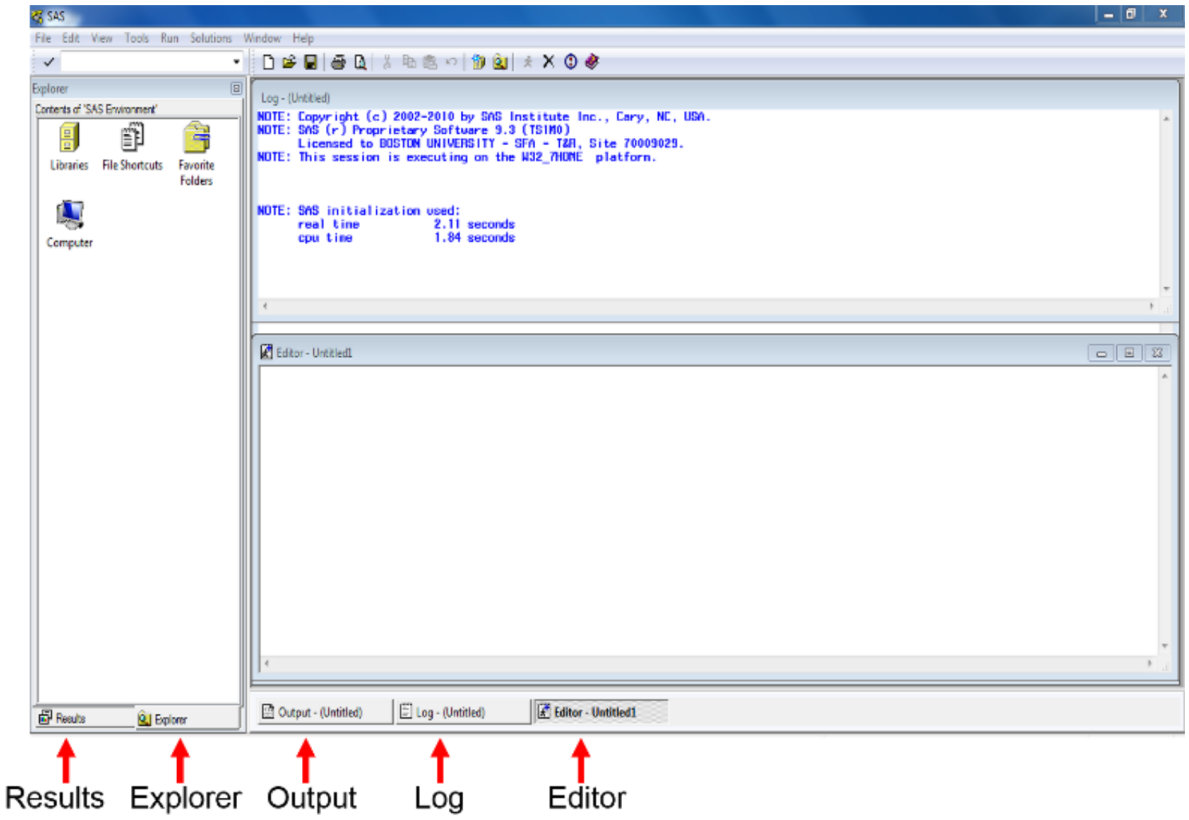
  Access via VLab, GSU's Remote Desktop Environment. Download and Connect to Cisco AnyConnect Client to connect to GSU's VPN (secureaccess.gsu.edu). Once connected to the VPN, login to VLab at: https://vlab.gsu.edu/ to access SAS.

- **Option 4**:

  Access via SAS OnDemand for Academics/SAS Studio. If you do not already have one, create a SAS profile at https://welcome.oda.sas.com/ Then, sign in with credentials and click SAS®Studio to access the web-based SAS environment.

1.1.2 SAS Windows

Once SAS has started, the screen will look similar to the following: The main SAS window is divided into several sub-windows:



- The menu and toolbar along the top of the window
- The **explorer/results browser** along the left hand side, where you can a listing of the results of successful SAS program.
- The **log** to the top right. This gives you information about possible errors after you have run your SAS program.
- The **program editor** below the log on the bottom right, where you create your SAS program.
- The windows bar along the bottom for you to switch all windows.

The **Editor (Program Editor)** window is a text editor that facilitates writing SAS programs (code). The Log window displays system messages, errors, and resource usage and is thus used to review program statements. The Output window displays output from statistical procedures run within the SAS program; however this is no longer the default. In SAS 9.3 output is sent to the Results Viewer which opens automatically when you run a procedure that generates output. The Results window displays a map of the Output window, and is useful for navigating the results of complicated analyses. Finally, the Explorer window contains all of the data sets in the current SAS session.

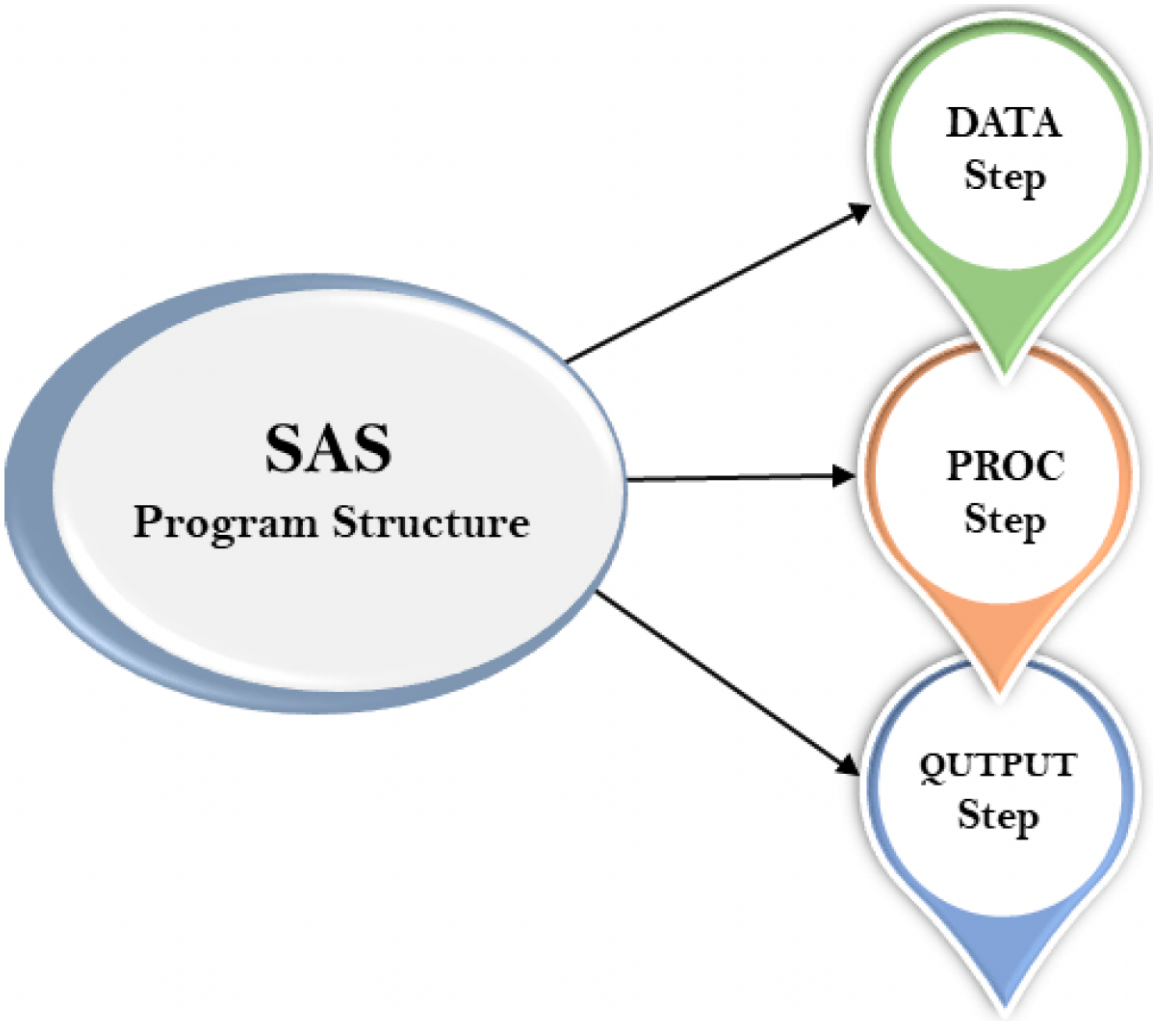
These windows can be moved or resized as desired. Only one SAS window is active at a time. The active window will have a shaded title bar at the top of the window, and a highlighted

windows bar at the bottom of the screen. In the above example, the Program Editor is the active window, with an "*Untitled*" program name. Note that the menu options for the SAS toolbar along the top of the screen depend on which window is currently active. (The active window can be changed by clicking on that window with the mouse, or by selecting the desired window from the Window menu.)

1.2 SAS Program

The programming structure of SAS consists of **three** significant steps:

- **DATA** step: create and modify a SAS data set for follow-up analysis
- **PROC** step: conduct data analysis
- **OUTPUT** step: show the analysis results




```
*Syntax of the SAS program:;
DATA dataset name; /* Name of the data set. */
INPUT var1,var2; /* Defines the variables in this data set. */
NEW_VAR; /* Creates a new variable. */
LABEL; /* Assign labels to variables. */
DATALINES; /* Enters the data. */
RUN;
```

1.3 SAS Dataset

SAS dataset is used to organize data values in a tabular form, i.e., in the form of rows for observations and columns for variables.

A SAS data set is a matrix whose each column is for each **variable** and whose each row for each **observation** (e.g., subject).

Data sets can be entered in the SAS programming code or can be read in from a variety of external sources, such as text files, csv files, and Microsoft Excel. In subsequent classes we will discuss reading in data sets from external files. Once a data set has been created, commands or procedures can operate on these data sets.

Other than these steps programming structure also includes data set, label, variables, values, and run.

1.4 SAS Examples

1.4.1 Creating a Dataset

Our first task in using SAS will be to create a small dataset and "print" that dataset to the output window. As we mentioned in previous paragraph SAS programs usually start with a DATA step where the dataset is created. Once the dataset is available, various procedures can be run on the dataset. The example below is written in the SAS Program window. The program creates a dataset called "People" with 3 variables (columns) which are 'gender', 'height', and 'weight' and 14 observations (rows). Note that the values of the variables on each line are separated by one or more blanks. A few other things that you should note:

- All SAS statements end with a semicolon (**;**)
- More than one SAS statement can be put on a line, or a SAS statement can continue across several lines, if every statement *ends with a semicolon.*
- Data listed as part of the program is also terminated with a semicolon. Data does not have to be entered in the program; it can also be read from files that are external to the SAS program (more on that next week)
- *gender* is a character variables as indicated by the $, and height and weight are numeric variables.
- Once the dataset is created, various SAS procedures (called **PROC**s) can be used to analyze the data and present results. We will start with a listing of the data created with a procedure called **PROC PRINT.**

```
title1 'STAT 8678 Example 1';
title2 'Your name';
```

```
DATA people;
INPUT gender $ height weight;

DATALINES;
m 63 125
m 76 195
f 62 109
m 75 186
f 67 115
f 60 120
m 75 205
m 71 185
m 63 140
f 59 135
f 65 125
m 68 167
m 72 220
f 66 155
;

PROC PRINT DATA=people;
RUN;
```

The LOG window gives information on the execution of the program. If your program did not execute properly you should examine the log for error messages that may explain the failure. The program would then be modified if necessary and rerun.

SAS creates a new window called the **Results Viewer** when the program is executed and produces output. This window is in HTML format and a new tab for the window is created below the left-hand windows.

1.4.2 Sorting Data

The data can be sorted (in our case by *gender*) using **PROC SORT** by adding the following lines to the program. We can then go ahead and print our new dataset sorted by gender with the proc print step.

> **Reminder**
>
> Any procedural step we do must begin with PROC and every line must end with a semicolon and the command run;

```
PROC SORT data = people;
BY gender;
RUN;

PROC PRINT data=people;
TITLE3 "Raw data sorted only by gender";
RUN;
```

> **Note**
>
> Any time we use PROC SORT our original dataset is sorted. SAS does not create a copy then sort!

1.5 Generating Summary Statistics

The last procedure to be executed in this exercise is **PROC UNIVARIATE**. This procedure will allow us to see summary statistics for any *quantitative variable*. The output from PROC UNIVARIATE will be important for the early part of this statistical methods class. It will provide measures of central tendency (mean, median, mode) and measures of dispersion (variance, standard deviation, range) as well as other basic statistics.

```
PROC UNIVARIATE DATA=people PLOT;
  BY gender;
  TITLE3 "Univariate procedure output done separately by gender";
  TITLE4 "The analysis was done for two quantitative variables";
  VAR HEIGHT WEIGHT;
RUN;
```

The code below applies the procedure only to the variable gender. It can be run on any quantitative variable and it could be run on several variables at the same time by listing several variables in the **VAR** statement, which would provide a separate analysis for each variable.

The results for PROC UNIVARIATE will be listed in the results viewer.

1.6 Other notes

Other Note

- SAS does not distinguish between upper-case letters or lower-case letters in the program, either can be used. However, it does distinguish between upper and lower case in datasets, so the character strings "Carol", "carol" and "CAROL" would be considered different values of the variable "name" in the program above.
- Comments: Additionally, you may add comments anywhere in your program either by beginning the statement with an asterisk (*) and ending it with a semicolon (;) or by beginning with /* and ending with */. These comments may be thought of as marginal notes, and will show in the program editor and log, but not in the output window.

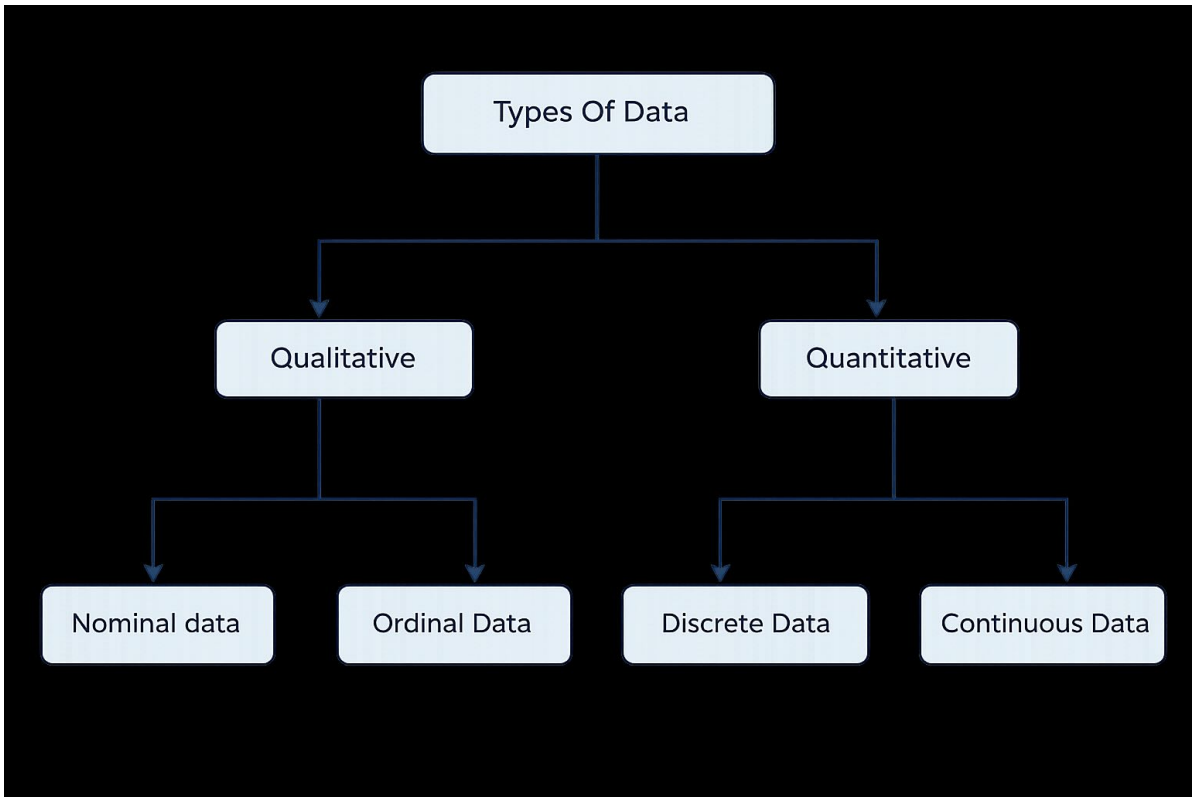
Shortcuts:

- F3 or the "running man": submits/run your program;
- F4: recalls text once the program editor;
- F5: directs user to the program editor;
- F6: directs user to the log;
- F7: directs user to the output;
- Ctrl+E clears content in the current window.
- Also, text can be copied with Ctrl+C, cut with Ctrl+X, or pasted with Ctrl+V.

1.7 Data Type

We can classify variables into *quantitative* variables and *qualitative* variables:

- Qualitative variables yield non-numerical information. Qualitative variables are often referred to as categorical variables, such as blood type. Qualitative variables can be further classified as
  - A nominal variable is a qualitative variable where no ordering is possible or implied in the levels, such as gender.
  - A ordinal variable is a qualitative variable with an order implied in the levels, such as health (poor, reasonable, good, or excellent)
- Quantitative variables yield numerical measurements. Quantitative variables can be further classified as discrete or continuous.
  - A discrete variable can assume only a countable number of values, such as headache severity scores.
  - A continuous variable is one that can take any one of an uncountable number of values in an interval, such as weight.



Question:

What is the type of each variable in the following dataset?

- AGE: The respondent's age in years
- GENDER: The respondent's sex coded 1 for male and 2 for female
- HAPPY: The respondent's general happiness, coded:1 for "Not too happy"2 for "Pretty happy"3 for "Very happy"
- TVHOURS: The average number of hours the respondent watched TV during a day

Table 1.1: Survey Respondent Summary

Respondent	AGE	Gen	HAPPY	TVHOURS
1	41	1	2	0
2	25	2	1	0
3	43	1	2	4
4	38	1	2	2
5	53	2	3	2
6	43	2	2	6

7	56	2	2	2

Answer:

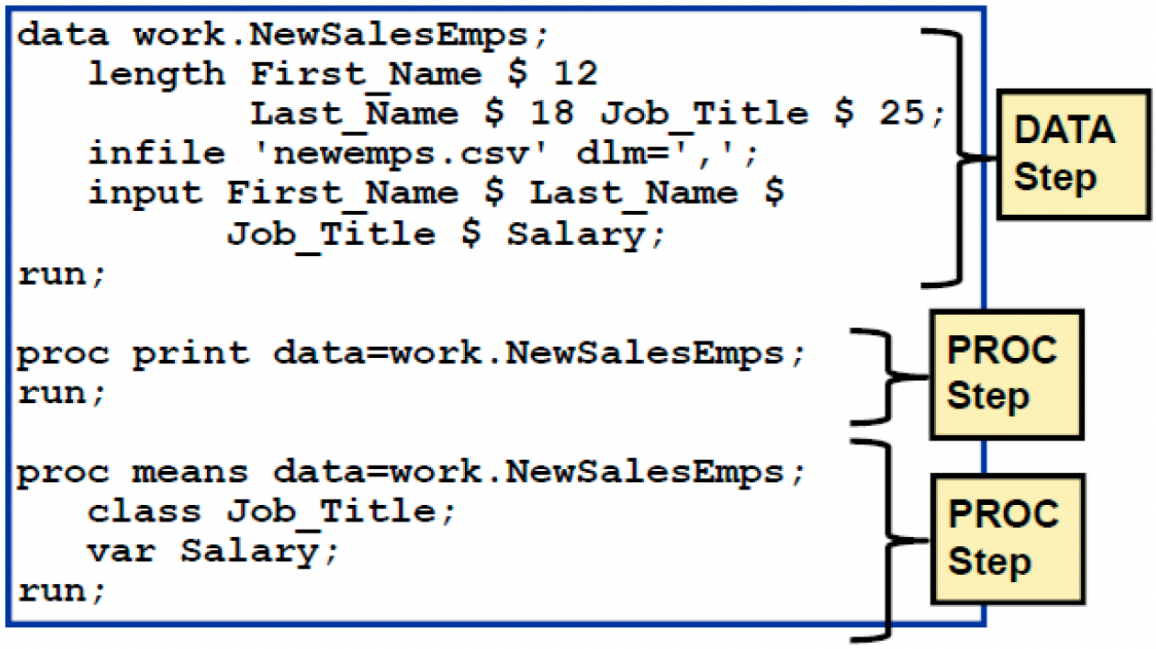
- Age: Continuous variable;
- SEX: qualitative;
- HAPPY: Discrete variable;
- TVHOURS: Continuous variable

2 Working with SAS Syntax

Learning objective:

1. Identify the characteristics of SAS statements.
2. Explain SAS syntax rules.
3. Insert SAS comments using two methods.
4. Identify SAS syntax errors.
5. Diagnose and correct a program with errors.
6. Save the corrected program

In general, writing a SAS program is to write a sequence of steps, and a step is a sequence of SAS statements:



How many statements are in the following step?

```
data work.NewSalesEmps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;
```

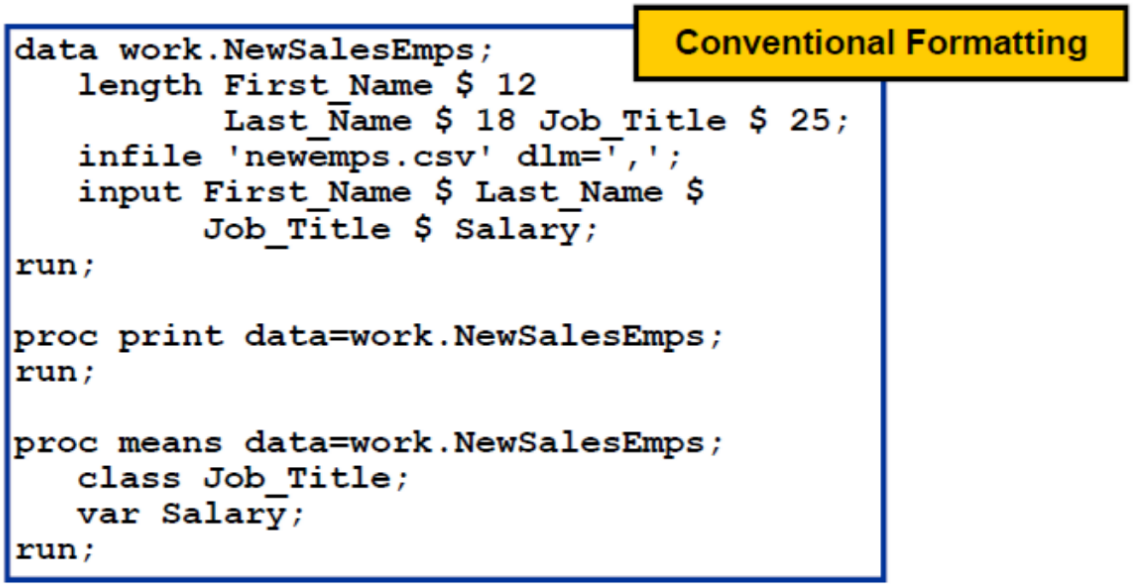
2.1 SAS Statments

SAS statements have these characteristics (Check the highlight context in the following examples):

- Usually begin with an identifying keyword.
- Always end with a semicolon.

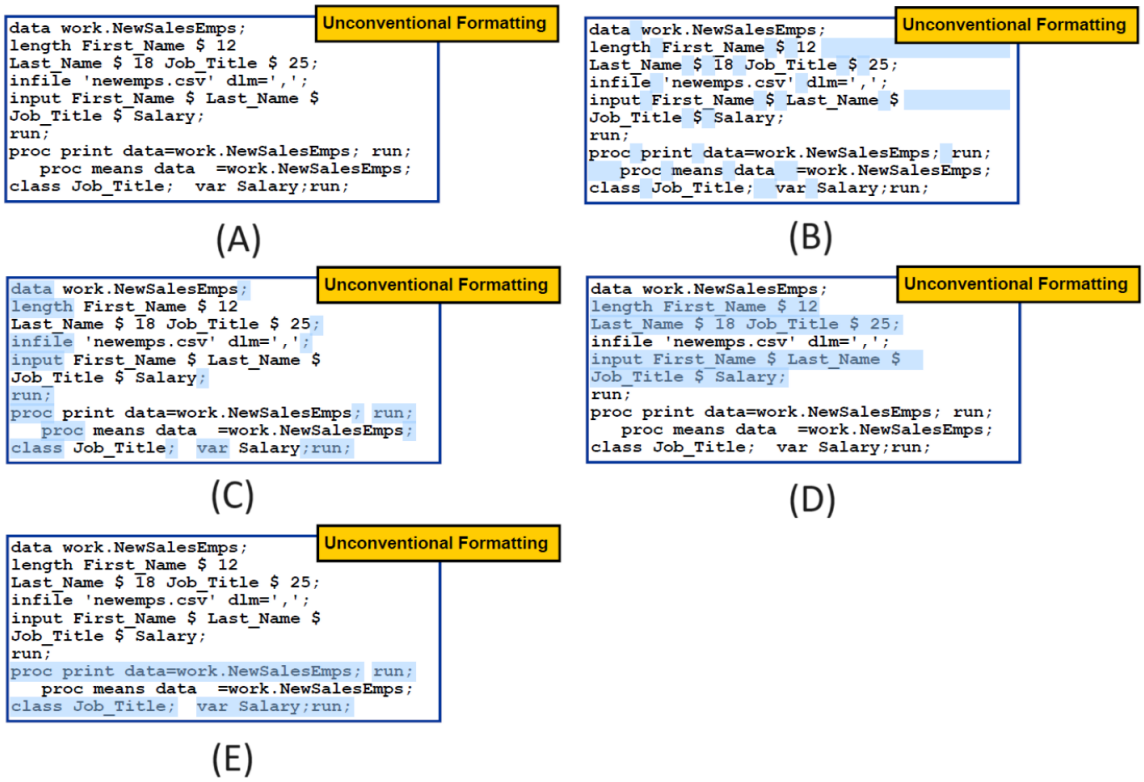
```
data work.NewSalesEmps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.NewSalesEmps;
run;

proc means data=work.NewSalesEmps;
   class Job_Title;
   var Salary;
run;
```

SAS programming statements are easier to read if you begin DATA, PROC, and RUN statements in column one and indent the other statements. This makes it more structured. Also, consistent spacing also makes a SAS program easier to read.



Overall, we can type SAS statements in the following ways:

- One or more blanks can be used to separate words.
- They can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.



Sometimes we may want to make comments/notes to easy remember our SAS statements or to let other people easily understand what our code want to say. These comments are text that SAS ignores during processing. You can use comments anywhere in a SAS program to document the purpose of the program, explain segments of the program, or mark SAS code as non-executing text.

There are two ways to insert comments in a SAS program:

1. Using an asterisk (*) to begin the comment and a semicolon (;) to end the comment.

```
* This is a comment in SAS;
```

2. Using slash-asterisk (/*) to begin the comment and asterisk-slash (*/) to end the comment.

```
/* This is a comment in SAS */
```

Avoid placing the /* comment symbols in columns 1 and 2. On some operating environments, SAS might interpret these symbols as a request to end the SAS job or session. An example is given below:

```
*----------------------------------------*
|   This program creates and uses the    |
|   data set called work.NewSalesEmps.   |
*----------------------------------------*;
data work.NewSalesEmps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary /*numeric*/;
run;
/*
proc print data=work.NewSalesEmps;
run;
*/
proc means data=work.NewSalesEmps;
   *class Job_Title;
   var Salary;
run;
```

2.2 Diagnosing and Correcting Syntax Errors

Syntax errors occur when program statements do not conform to the rules of the SAS language.

Examples of syntax errors:

- misspelled keywords
- unmatched quotation marks
- missing semicolons
- invalid options

When SAS encounters a syntax error, SAS prints a warning or an error message to the log; for example,

```
ERROR 22-322: Syntax error, expecting one of the following:
              a name, a quoted string, (, /, ;, _DATA_, _LAST_,
              _NULL_.
```

When SAS encounters a syntax error, SAS underlines the error and the following information is written to the SAS log:

- the word **ERROR** or **WARNING**
- the location of the error + an explanation of the error

This program has three syntax errors. What are the errors?

```
daat work.NewSalesEmps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.NewSalesEmps
run;

proc means data=work.NewSalesEmps average max;
   class Job_Title;
   var Salary;
run;
```

2.3 Solution to the example

1. Exercise 1: 5
2. Exercise 2:

```
daat work.NewSalesEmps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.NewSalesEmps
run;

proc means data=work.NewSalesEmps average max;
   class Job_Title;
   var Salary;
run;
```

3 Import and Export Dataset

Learning objective:

1. Import a csv file into SAS
2. Export a csv file from SAS after data process

One of the strengths of SAS as a data analysis tool is its ability to read data from many sources, subset or combine data sets, and modify the datasets to accomplish various tasks. The most common types of external data sets used in SAS are EXCEL files (XLS extent), comma separated value files (CSV extent) and various space separate text files (PRN or TXT extent). A CSV file is actually a text file and can be read in any text reader (NOTEPAD or WORDPAD in Windows). In fact, the SAS files themselves, as well as the LOG and the LST files produced by a SAS by a batch submit, are also simple text files.

Format	Description	File Extension
WK1	Lotus 1 spreadsheet	.WK1
WK3	Lotus 3 spreadsheet	.WK3
WK4	Lotus 4 spreadsheet	.WK4
EXCEL	Excel Version 4 or 5 spreadsheet	.XLS
EXCEL4	Excel Version 4 spreadsheet	.XLS
EXCEL5	Excel Version 5 spreadsheet	.XLS
EXCEL97	Excel 97 spreadsheet	.XLS
DLM	delimited file (default delimiter is a blank)	.*
TAB	delimited file (tab-delimited values)	.TXT
CSV	delimited file (comma-separated values)	.CSV

3.1 Reading from External Files

The **PROC IMPORT** statement is the best way to enter external data sets. The CSV file we will be using is called "grades.csv". Download and save it in your favourite folder and mark the complete path to it. Then use the following code to import it, making sure you put the correct path on the **DATAFILE** argument.

```
PROC IMPORT OUT= GRADES_temp
  DATAFILE= "Put Your Path Here/grades_temp.csv" DBMS=CSV REPLACE;
  GETNAMES=YES;
  DATAROW=2;
RUN;
```

The **IMPORT** statement reads the dataset and stores it as the value designated by "OUT" in this case it will be saved as "Grades" in the library "Work".

The **DBMS** statement defines the type of input SAS should be reading. The following table gives you all the possible choices. The REPLACE argument forces SAS to overwrite any older datasets with the same name.

The **GETNAMES** = YES or NO statement for spreadsheets and delimited external files, determines whether to generate SAS variable names from the column names in the input file's first row of data. If you specify **GETNAMES** = NO or if the column names are not valid SAS names, PROC IMPORT uses the variable names VAR0, VAR1, VAR2, and so on. You may replace the equals sign with a blank.

The **DATAROW** argument tells SAS where to start reading for input data. In our case it is row 2 since row 1 is used for variable names.

We use the print procedure to see the dataset,

```
PROC PRINT DATA=GRADES_temp;
RUN
```

The first few rows are shown as follows

The SAS System

Obs	Student	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5	Quiz6	Midterm	EC	Gender
1	.	7	7	7	7	7	7	25	2	M
2	1	4.55	6.8	6	5.1	5.75	6.7	13	1	M
3	2	5	6.1	6.7	5.4	7	3.8	21.5	1	M
4	3	3.75	6.5	6.9	2.1	6.6	5.5	21	2	M
5	4	5.05	6.2	7	5.9	6.75	6	24.5	1	M
6	5	5.35	6.8	6.6	5.1	6.6	6.5	24	1	M
7	6	7	7	6.7	6.1	7	7	25	1	F

3.2 Export a File from SAS

After reading a dataset into SAS, we may need to conduct some initial step/analysis to re-organize dataset for doing further data analysis. In the *grade* example, the first row shows the maximum points available for each quiz. We need to remove this row so that our analysis is correct. This can be done by the following SAS code

```
DATA GRADES;
  SET GRADES_temp NOBS=COUNT
    IF _n_         <= 1 THEN DELETE;
RUN;
```

Note: GRADES_temp is the name of the old dataset and *GRADES* is the name of the new dataset after the first row is deleted.

After re-organize the dataset, we may want to export the updated dataset for use in the future:

This can be done by

```
PROC EXPORT DATA= GRADES
  OUTFILE= "Put Your Path Here/grade_v2.csv" DBMS=CSV REPLACE;
  REPLACE;
RUN;
```

We will talk about more reorganizing skills in SAS in the next topic.

Which statement is true concerning the DATALINES statement based on reading the comment?

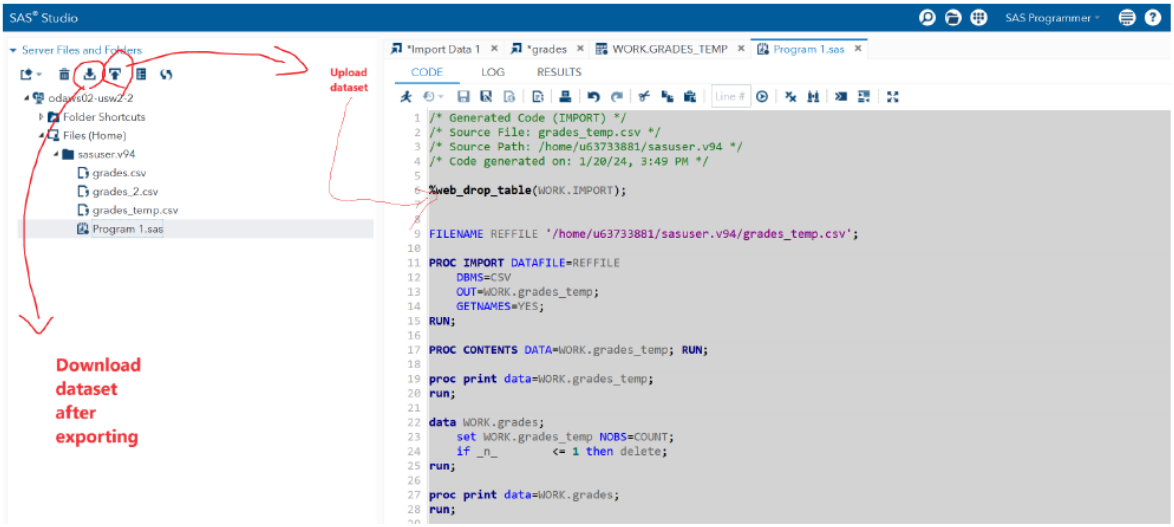
a. The DATALINES statement is used when reading data located in a raw data file.

b. The DATALINES statement is used when reading data located directly in the program.

3.3 Import & Export in SAS Virtual Studio

The key is

- Upload data before conducting "Import"
- Download data after conducting "Export"



A example given below will be demonstrated during the class.

```
/* Generated Code (IMPORT) */
/* Source File: grades_temp.csv */
/* Source Path: /home/u63733881/sasuser.v94 */
/* Code generated on: 1/20/24, 3:49 PM */

%web_drop_table(WORK.IMPORT);

FILENAME REFFILE '/home/u63733881/sasuser.v94/grades_temp.csv';

PROC IMPORT DATAFILE=REFFILE
    DBMS=CSV
    OUT=WORK.grades_temp;
    GETNAMES=YES;
RUN;

PROC CONTENTS DATA=WORK.grades_temp;
RUN;

PROC print data=WORK.grades_temp;
RUN;

DATA WORK.grades;
SET WORK.grades_temp NOBS=COUNT;
IF _n_ <= 1 THEN DELTE;
RUN;
```

```
PROC PRINT DATA=WORK.grades;
RUN;

PROC EXPORT data=WORK.grades
    OUTFILE = "/home/u63733881/sasuser.v94/grades_2.csv"
    DBMS = csv
    REPLACE;
RUN;
```

3.4 Solution to the example

Answer of the example: b

4 Random Variables

Learning Objectives

1. Distinguish between **discrete** and **continuous** random variables
2. Define random variables for real-world data problems
3. Identify appropriate probability distributions for common data types
4. Connect random variables to **data columns** used in SAS programs to represent a real world question

4.1 What Is a Random Variable?

A **random variable (RV)** is a numerical quantity whose value depends on the outcome of a random experiment.

We typically denote a random variable by an uppercase letter, such as X and its realized value by a lowercase letter, such as $X = x$. The random variable can be *continuous* or *discrete*.

In practice, we often observe multiple realizations, or running the random experiment multiple times, say n times or n realizations. We denote these realizations as:

$$X_1 = x_1,\ X_2 = x_2,\ \ldots,\ X_n = x_n.$$

The number n is called the **sample size**.

Define whether the following random variables are discrete or continuous, and the possible values that X takes.

- Number of emails received by a server in one hour ($X = 0, 1, 2, \ldots$: discrete)
- Time (in minutes) until a machine fails ($X = x \in [0, \infty)$: continuous)
- Total number of defects on a manufactured item ($X = 0, 1, 2, \ldots$: discrete)
- Daily maximum temperature in Atlanta (in degrees Fahrenheit) ($X = x \in (-\infty, \infty)$: continuous)
- Whether a randomly selected loan defaults within one year ($X = 1$ if default, 0 otherwise: discrete)

Define whether the following random variables are discrete or continuous, and the possible values that X takes.

1. Number of transactions made by a customer in a day
2. Response time (in seconds) of a web service request
3. Count of hospital admissions in a city per week discrete)
4. Proportion of time a system is idle during a day

4.2 Discrete vs Continuous Random Variables

4.2.1 Discrete Random Variables

A **discrete random variable** takes values in a **finite or countable set**.

Examples:

- Number of heads in three coin flips
- Number of students passing an exam
- Number of events occurring in a fixed time period

A discrete random variable is described by a **probability mass function (PMF)**:

Value of X	x_1	x_2	x_3	...	x_m
Probability	p_1	p_2	p_3	...	p_m

These probabilities satisfy:

- $0 \leq p_i \leq 1$,
- $\sum_{i=1}^{m} p_i = 1$.

We calculate the probability of events modelled by discrete random variables by summing up the probability p_i for the values x_i that make up the event.

Suppose the length X (in minutes) of an international phone call has distribution:

X	1	2	3	4
P(X)	0.2	0.5	0.2	0.1

Then, calculate the following probabilities

a. $P(X \leq 2)$
b. $P(X < 2)$
c. $P(X > 1)$

4.3 Common Discrete Distributions

4.3.1 Bernoulli Distribution

Used for **binary outcomes**:

- success / failure
- yes / no
- 1 / 0

Notation: $X \sim \text{Ber}(p)$, where p is the probability of success.

Example:

Whether a patient has diabetes (1 = yes, 0 = no).

How to specify the probability p will be discussed in the following lecture. This is related to the procedure of statistical inference.

4.3.2 Poisson Distribution

Used to model **counts of events over time or space**.

Notation: $X \sim \text{Poi}(\lambda)$

where λ is the **mean rate**.

Examples:

- Number of trades per day
- Number of system failures per week
- Number of arrivals to a service queue

Similar to the probability p from the Bernoulli distribution, how to specify the rate λ will be discussed in the following lecture.

4.4 Continuous Random Variables

A **continuous random variable** can take **any value** x in an interval.

Examples:

- Height of individuals
- Time until failure of a component
- Test scores treated as continuous

Probabilities are defined using **density functions**, not point probabilities. Some common continuous distributions are as follows.

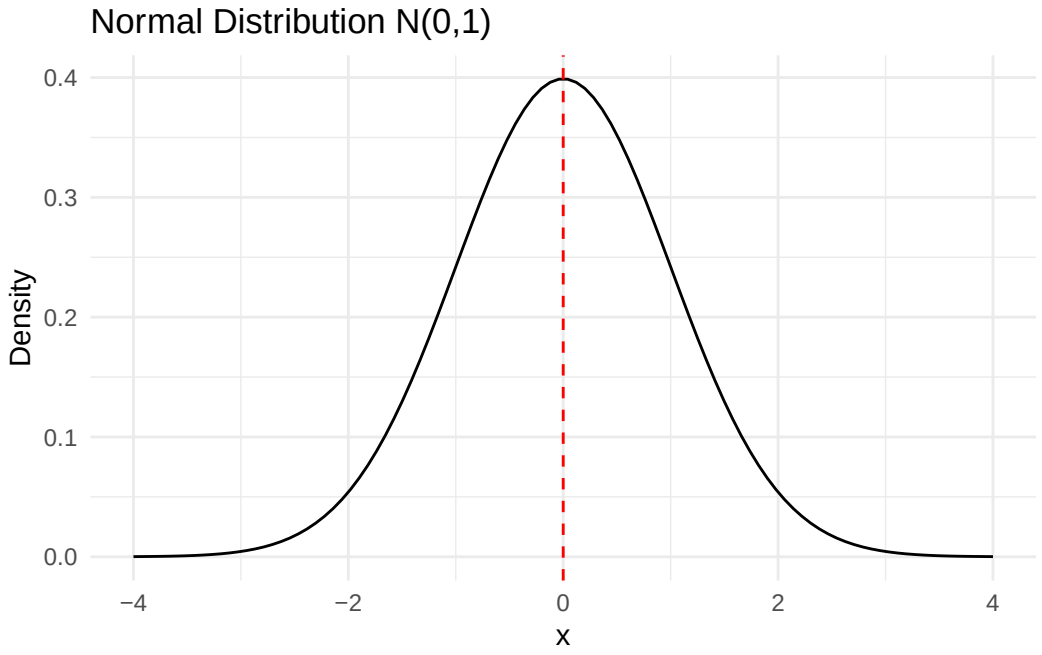
4.4.1 Normal Distribution

The normal distribution is also called the **Gaussian distribution**. It is characterized by two parameters: the mean μ and the standard deviation σ. It is unimodal and symmetric around the mean which is the centre of the mass. A continuous random variable X that has a normal distribution is said to be *normal* or *normally distributed*.

Notation: $X \sim N(\mu, \sigma^2)$. Sometimes the standard deviation may be used instead of the variance, which is the square of the variance.

Institution of the normal distribution:

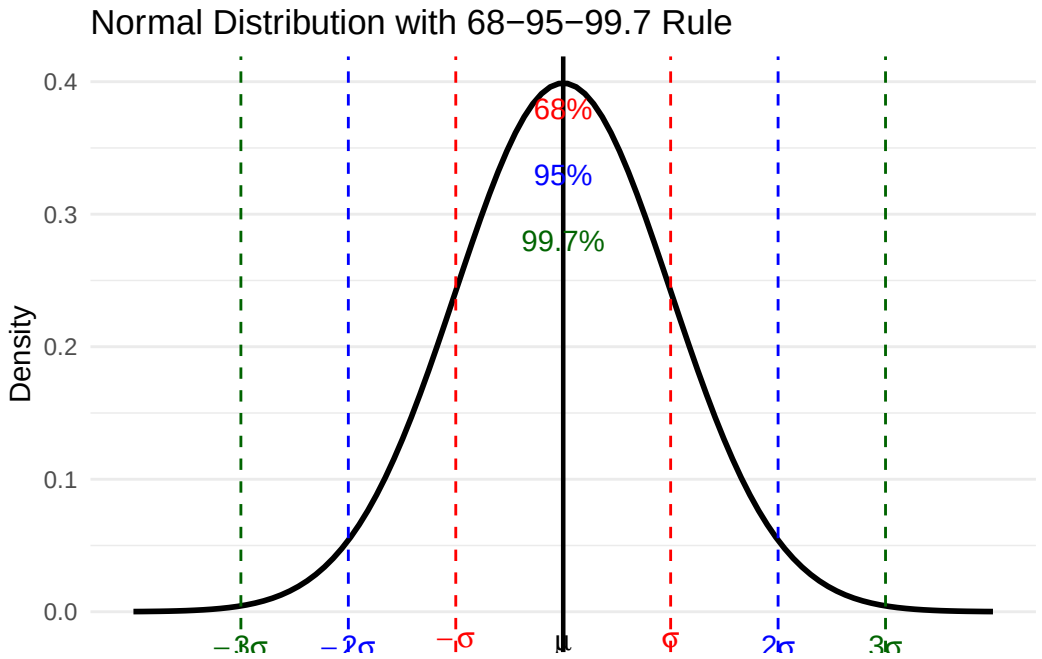
- mean: μ
- variance: σ^2



A distribution plot of the normal distribution with mean 0 and standard deviation 1 is shown above. It is often referred as the *standard normal distribution*. In practice, we would like to "standardized" the data to have mean 0 and standard deviation 1 for the subsequent analysis.

Normal distribution play an important role in statistical inference. One of the important property is the 68-95-99.7 rule: - About 68% of the data falls within one standard deviation of

the mean - About 95% of the data falls within two standard deviations of the mean - About 99.7% of the data falls within three standard deviations of the mean



4.4.2 Exponential Distribution

The exponential distribution is another common distribution where a few outcomes are most likely with a rapidly decreasing probability for larger values. It is often used to model waiting times or lifetimes of objects. It is similar to the *geometric distribution* in the discrete case.

Two ways to specify the parameter.

1. $X \sim \text{Exp}(\lambda)$, where λ is the *rate* parameter.
2. $X \sim \text{Exp}(\beta)$, where β is the *scale* parameter, and $\beta = 1/\lambda$.

The notation is either $X \sim \text{Exp}(\lambda)$ or $X \sim \text{Exp}(\beta)$. But to be sure which parameterization is used, we need to check the definition of the distribution.

Please define adequate random variables for the following data example, and thaink about what distribution can be used to model the data.

- Suppose we want to know whether the rate of diabetes of a certain area is too high. The researchers randomly discuss with 10 individuals in a certain area to know whether they have diabetes. The data are collected as "yes" or "no" answers as follows

Individual	1	2	3	4	5	6	7	8	9	10
Outcome	1	0	0	0	0	1	0	0	0	0

- A comapny that manufactures light blulb claims that a particular type of light bulb will last 850 hours on average. To justify the claim more scientifically, a researcher randomly selects 10 light bulbs of that type and measures the lifetimes (in hours) of the light bulbs as follows:

Light Bulb	1	2	3	4	5	6	7	8
	831	832	840	819	822	836	829	817

In the following classes we will assume the dataset we have can be well-modelled represented for the underline studies. The data collection, however, is beyond the scope of the class. For those interested in data collection, please refer to the *survey sampling* or *experimental design* area.

4.5 Solution to the exercise

Answer of the exercise

Exercise 1

1. Number of transactions made by a customer in a day ($X = 0, 1, 2, ...$: discrete)
2. Response time (in seconds) of a web service request ($X = x > 0$: continuous)
3. Count of hospital admissions in a city per week ($X = 0, 1, 2, ...$: discrete)
4. Proportion of time a system is idle during a day ($X = x \in [0, 1]$: continuous)

Exercise 2:

- $P(X \leq 2) = 0.7$
- $P(X < 2) = 0.2$
- $P(X > 1) = 0.8$

Part II

Statistical Analysis

5 Introduction to Statistical Inference I

Learning Objectives

1. Be familar with the difference about a probability problem and a statistical problem
2. Applied the one sample t-test to do inference for one sample mean problem

5.1 Probability versus Statistics

- In *probability*, we assume that random variables $X_1, \ldots, X_n$ follow a distribution with **known parameters**.
  Under this model, we can calculate probabilities of events of interest.
- In *statistics*, although we still use a distribution to model $X_1, \ldots, X_n$, the **parameters of the distribution are assumed to be unknown**.
  Our primary goal is to use observed data, the realization $X_1 = x_1, \ldots, X_n = x_n$—to make inference about these unknown parameters.

There are three common goals of statistical inference:

- **Point estimation**
- **Interval estimation**
- **Hypothesis testing**

> **Note**
>
> Note that there may be multiple valid methods for achieving each goal, even when working with the same dataset.

5.2 Example: One-Sample Mean Problem

The term *one sample* does **not** mean that there is only one observation. Instead, it means that there is **one population** under study.

In this problem, we are interested in making inference about the population mean, denoted by μ.

5.2.1 Problem Formulation

Suppose we want to conduct statistical inference on the mean length of a certain type of court case. Let

$$\mu = \text{the mean length of a certain type of court case.}$$

However, the true value of μ is unknown because we cannot observe all realizations of this process. As a result, statistical inference is required.

If μ were known, no inference would be necessary. What we can do in practice is to **collect data**. Suppose we randomly select 20 court cases of the same type from historical records and observe their case lengths (in days):

$$43,\ 90,\ 84,\ 87,\ 116,\ 95,\ 86,\ 99,\ 93,\ 92,$$
$$121,\ 71,\ 66,\ 98,\ 79,\ 102,\ 60,\ 112,\ 105,\ 98.$$

From a statistical perspective, these observed values are treated as **realizations of random variables**. Specifically, we denote

$$X_1 = 43,\ X_2 = 90,\ \dots,\ X_{20} = 98.$$

To model the data, we assume that the random variables

$$X_1, X_2, \dots, X_{20}$$

are **independent and identically distributed** according to a normal distribution with mean μ and variance σ^2, that is,

$$X_i \overset{\text{i.i.d.}}{\sim} N(\mu, \sigma^2), \qquad i = 1, \dots, 20.$$

Here, both μ and σ^2 are unknown parameters. In this example, our primary interest lies in estimating the population mean μ, while the variance σ^2 is treated as a nuisance parameter.

We can use the **maximum likelihood estimator** (MLE) or the **method of moments estimator** (MME) to construct a point estimator of μ. Under this model, however, they are coincide. The resulting estimator is the **sample mean**, denoted by

$$\hat{\mu} = \bar{X}_{20},$$

where

$$\bar{X}_{20} = \frac{1}{20}\sum_{i=1}^{20} X_i.$$

Under the assumed model, we may use either the **maximum likelihood estimator** or the **method of moments** to construct a point estimator for the population mean μ. In this model, both methods lead to the same estimator: the **sample mean**. We denote this estimator by

$$\hat{\mu} = \bar{X}_n.$$

Pay careful attention to the notation. We use a **capital letter** $\bar{X}_n$ to emphasize that the estimator itself is a **random variable**. Since the estimator is a function of the random variables $X_1, \ldots, X_n$, it is also random and therefore follows a probability distribution. The probability distribution of an estimator is called its **sampling distribution**. We will return to the concept of sampling distributions later in the course.

5.2.2 Formal Definitions

To be more precise, we introduce the following definitions.

- A **statistic** is any function of the random variables $X_1, \ldots, X_n$. Because it is a function of random variables, a statistic is itself a random variable and therefore has a probability distribution. If we use the parametric model to describe it, then there is a distribution that depends on the unknown parameters.
- A **(point) estimator** of a parameter θ, denoted by $\hat{\theta}$, is a statistic used to estimate θ. (Note, an estimator is a random variable because it is a statistics).
- A **(point) estimate** is the numerical value obtained by evaluating the estimator using the observed data $x_1, \ldots, x_n$. In particular, we use **lowercase notation** to indicate an estimate. For example, $\bar{x}_n$ denotes the observed value of the estimator $\bar{X}_n$.

So far, we have constructed a point estimator (and hence a point estimate). To construct a confidence interval or perform hypothesis testing, we need to know the **sampling distribution** of the estimator $\bar{X}_n$.

Under the normal model with unknown variance, the sampling distribution can be expressed as

$$\frac{\bar{X} - \mu}{S/\sqrt{n}} \sim t_{n-1},$$

where

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}},$$

and t_{n-1} denotes the **Student's** t distribution with $n-1$ degrees of freedom.

5.2.3 Confidence Interval for μ

Based on this sampling distribution, a $(1-\alpha) \times 100\%$ confidence interval for μ is given by

$$\left[\bar{X} - t_{n-1,\alpha/2}\frac{S}{\sqrt{n}},\ \bar{X} + t_{n-1,\alpha/2}\frac{S}{\sqrt{n}}\right].$$

5.2.4 Hypothesis Testing for μ

Using the same sampling distribution, we can conduct one of the following hypothesis tests:

- **Two-sided test**
$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu \neq \mu_0$$
- **Left-tailed test**
$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu < \mu_0$$
- **Right-tailed test**
$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu > \mu_0$$

In some textbooks, equivalent hypotheses are written as

- $$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_1 : \mu \neq \mu_0$$
- $$H_0 : \mu \geq \mu_0 \quad \text{vs.} \quad H_1 : \mu < \mu_0$$
- $$H_0 : \mu \leq \mu_0 \quad \text{vs.} \quad H_1 : \mu > \mu_0$$

Fortunately, **SAS** can perform point estimation, confidence interval construction, and hypothesis testing simultaneously, provided we clearly specify:

- **The statistical question**
  (one-sample mean, one-sample variance, two-sample problem, etc.)
- **The inferential goal**
  (point estimation, confidence interval, hypothesis testing)
- **The statistical model and method**
  (for example, why we use the t distribution for the one-sample mean problem instead of the normal distribution or another alternative)

One of the main goals of this course is to help you build a mental "library" of statistical tools. Later, when you encounter a statistical question, you will be able to identify an appropriate method and then use SAS to obtain numerical results for interpretation.

5.3 Revisit the data example by using SAS

The court length data can be read by the following DATA step:

```
DATA TIME;
  INPUT TIME @@;
DATALINES;
43 90 84 87 116 95 86 99 93 92
121 71 66 98 79 102 60 112 105 98
;
RUN;
```

The only variable in the DATA set, *time*, is assumed to be normally distributed. The trailing at signs (@@) indicate that there is more than one observation on a line. The following statements invoke PROC TTEST for a one-sample t test:

```
PROC TTEST H0=80 PLOTS(SHOWH0) SIDES=2 ALPHA=0.1;
  VAR TIME;
RUN;
```

- THE VAR statement indicates that the *time* variable is being studies
- the H0= option specifies that the mean of the time variable should be compared to the null value rather than the default value of 0
- the PLOTS(SHOWH0) option requests that this null value be displayed on all relevant graphs
- the SIDE=2 option specifies that a the focus of the research question, namely whether the mean count case length is not equal to 80days, rather than less or greater than 80 days (in which case you would use the SIDE=L or SIDE=U options, respectively)

- the ALPHA=0.1 option requests 90% confidence interval rather than the default 95% confidence interval

The output is presented in the table below.

The TTEST Procedure

Variable: time

N	Mean	Std Dev	Std Err	Minimum	Maximum
20	89.8500	19.1456	4.2811	43.0000	121.0

Mean	90% CL Mean		Std Dev	90% CL Std Dev	
89.8500	82.4474	97.2526	19.1456	15.2002	26.2374

DF	t Value	Pr > |t|
19	2.30	0.0329

> **Note**
>
> Some SAS procedures produce graphs as automatically as they produce tables if the ODS GRAPHICS option is used. The graphs are integrated with tables in the ODS output.

```
ODS GRAPHICS ON;

PROC TTEST H0=80 PLOTS(SHOWH0) SIDES=U ALPHA=0.1;
  VAR TIME;
RUN;

ODS GRAPHICS OFF;
```

you will see the following results which include the same table you see in the last figure but with two more figures. We will talk those two Figures in the following classes, especially the QQ plot.

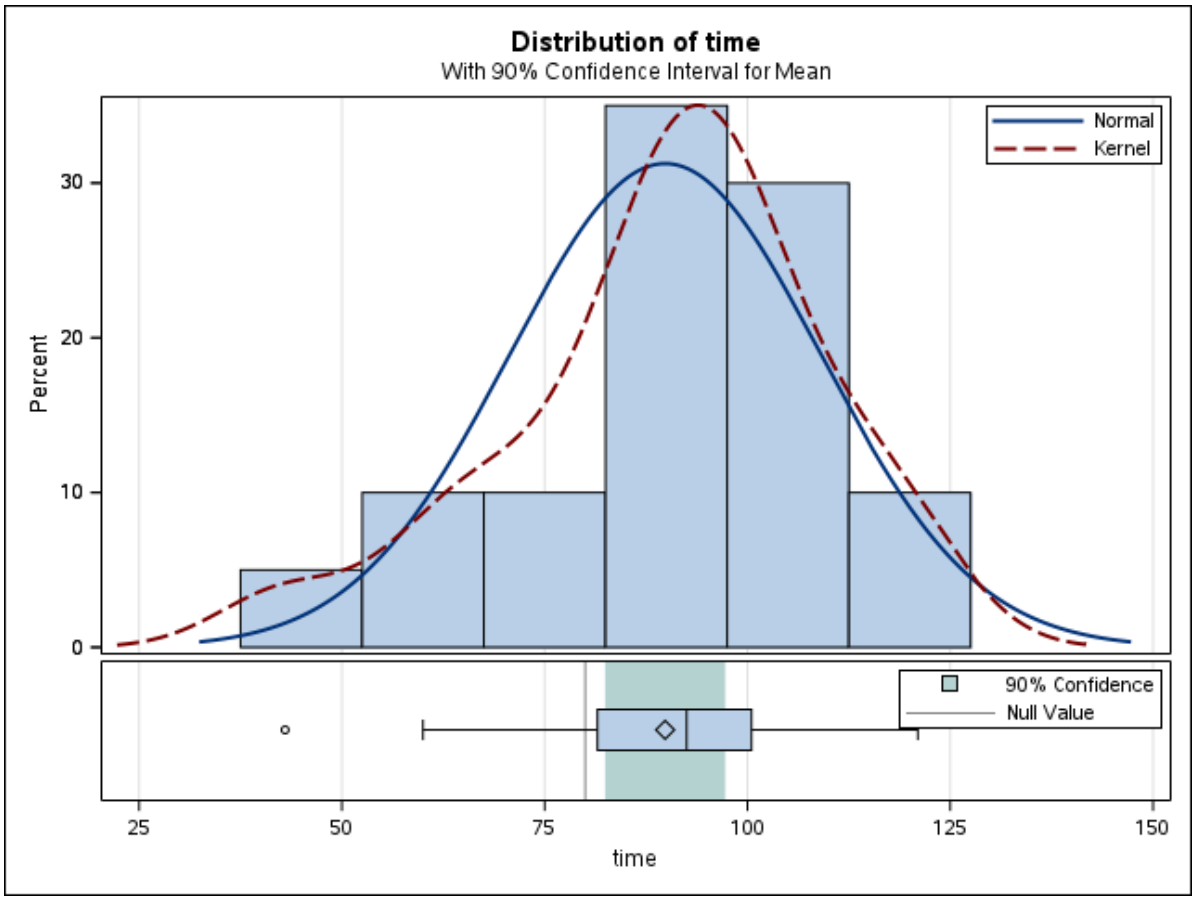
The TTEST Procedure

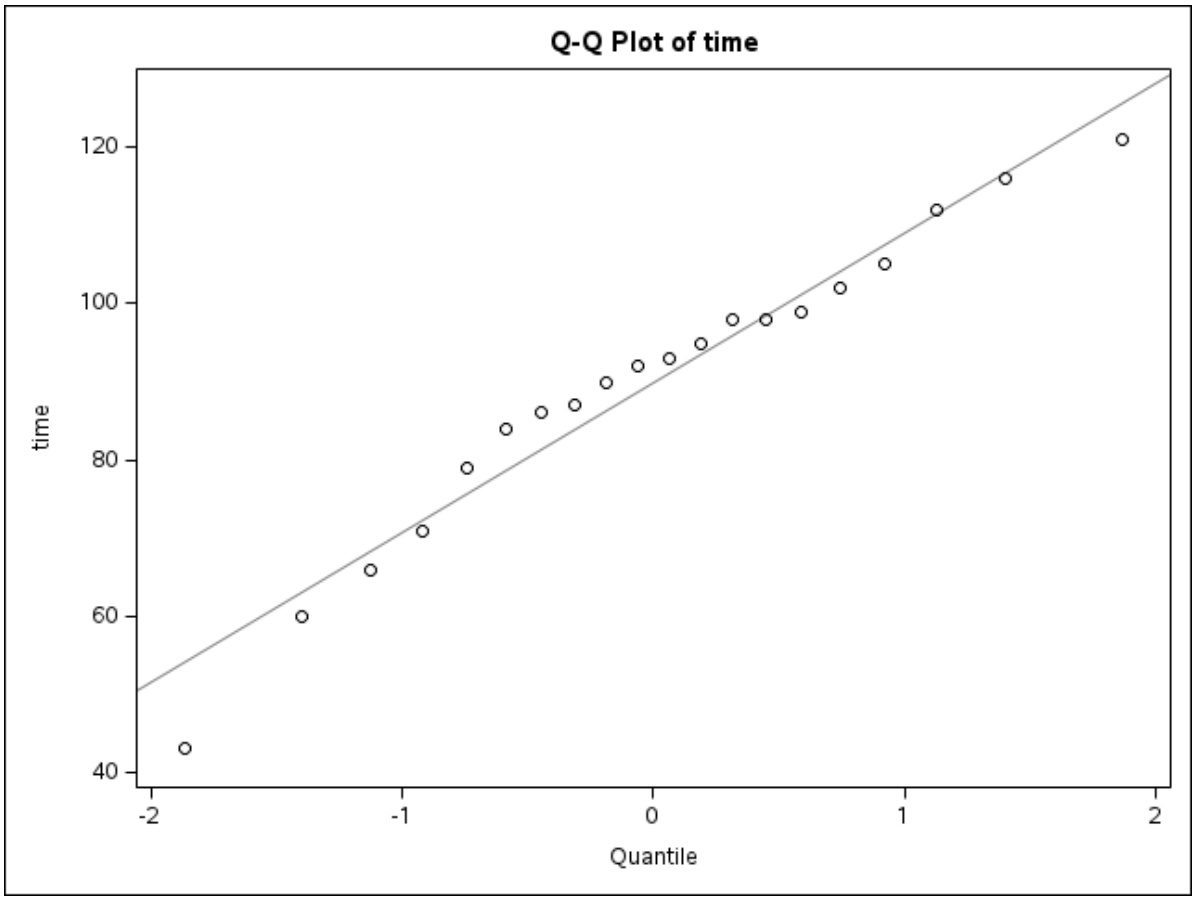
Variable: time

N	Mean	Std Dev	Std Err	Minimum	Maximum
20	89.8500	19.1456	4.2811	43.0000	121.0

Mean	90% CL Mean		Std Dev	90% CL Std Dev	
89.8500	82.4474	97.2526	19.1456	15.2002	26.2374

DF	t Value	Pr > |t|
19	2.30	0.0329


Distribution of time
With 90% Confidence Interval for Mean
Normal
Kernel
Percent
30
20
10
0
90% Confidence
Null Value
25
50
75
100
125
150
time



5.4 More about hypothesis and confidence intervals

5.4.1 How to construct a confidence interval?

Consider a **95% confidence interval** for the population mean μ. Under the normal approximation (or by the Central Limit Theorem), we have

$$P(-1.96 < Z < 1.96) \approx 0.95,$$

where Z is a standard normal random variable.

Equivalently, this can be written as

$$P\left(-1.96 < \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} < 1.96\right) \approx 0.95.$$

5.4.2 Rearranging the Inequality

Rearranging the terms inside the probability statement yields

$$P\left(\bar{X} - 1.96\frac{\sigma}{\sqrt{n}} < \mu < \bar{X} + 1.96\frac{\sigma}{\sqrt{n}}\right) \approx 0.95.$$

5.4.3 Large-Sample Confidence Interval

Thus, a **large-sample 95% confidence interval** for μ is given by

$$\left[\bar{X} - 1.96\frac{\sigma}{\sqrt{n}},\ \bar{X} + 1.96\frac{\sigma}{\sqrt{n}}\right].$$

More compactly, we often write this interval as

$$\bar{X} \pm 1.96\frac{\sigma}{\sqrt{n}}.$$

5.5 3.2 Hypothesis Testing and Confidence Intervals Always Agree

5.5.1 Interpretation of the p-value

The p-value is the probability of observing data **at least as favorable to the alternative hypothesis** H_A as the data actually observed, **assuming the null hypothesis** H_0 is true.

In this example, the observed sample mean is either greater than 3.56 or less than 3.18, and the null hypothesis assumes the true population mean is $\mu = 3.37$.

5.5.2 Computing the p-value

We compute the p-value as

$$P(\bar{X} > 3.56 \text{ or } \bar{X} < 3.18 \mid \mu = 3.37).$$

This can be written as the sum of two tail probabilities:

$$P(\bar{X} > 3.56 \mid \mu = 3.37) + P(\bar{X} < 3.18 \mid \mu = 3.37).$$

Standardizing using the normal distribution yields

$$P\left(Z > \frac{3.56 - 3.37}{0.31/\sqrt{147}}\right) + P\left(Z < \frac{3.18 - 3.37}{0.31/\sqrt{147}}\right).$$

Evaluating the standardized values gives

$$P(Z > 7.43) + P(Z < -7.43).$$

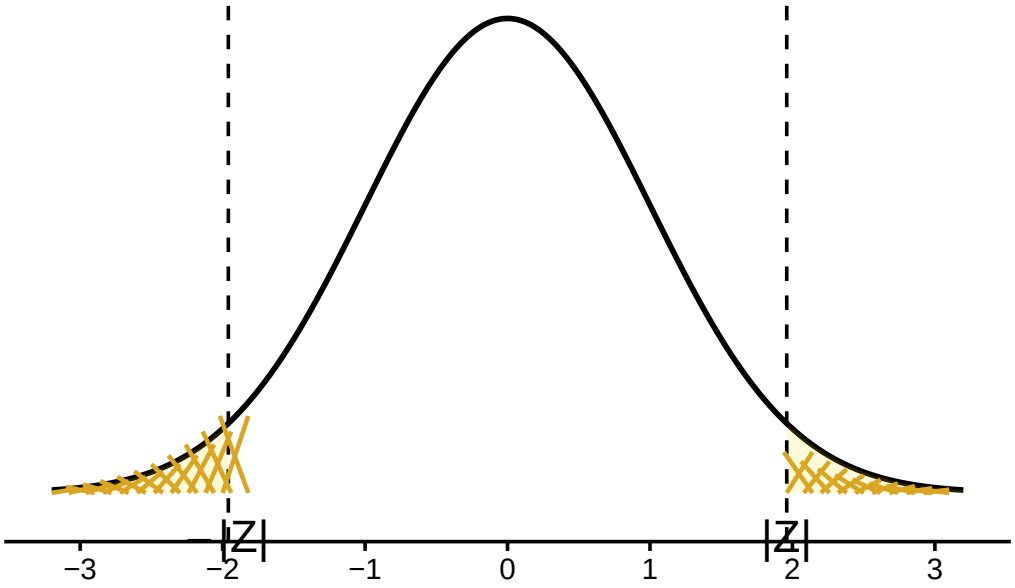
This probability is extremely small:

$$10^{-13} \approx 0.$$

5.5.3 Connection to Confidence Intervals

Because the p-value is essentially zero, we strongly reject H_0 at any reasonable significance level.

Equivalently, the hypothesized value $\mu = 3.37$ does **not** lie inside the corresponding confidence interval for μ.



This illustrates a key principle:

> **Hypothesis testing and confidence intervals always lead to the same conclusion when they are constructed at compatible significance levels.**

6 Introduction to Statistical Inference II

Learning Objectives

1. Distinguish between a *probability problem* and a *statistical (inference) problem*
2. Apply the *one-sample t* test to inference for a population mean
3. Conduct *model diagnostics* for the normality assumption
4. Correctly *interpret point estimates, confidence intervals, and hypothesis tests*

6.1 1. Recall: One-Sample Mean Problem

In the previous class, we studied inference for a *population mean* and show how to run SAS to obtain the statistical inference result from one sample t test. The example for illusrtratioe is to study the mean length of a certain type of the court case, i.e.,

$$\mu = \text{the mean length of a certain type of court case.}$$

The data for the study is

$$43,\ 90,\ 84,\ 87,\ 116,\ 95,\ 86,\ 99,\ 93,\ 92,$$
$$121,\ 71,\ 66,\ 98,\ 79,\ 102,\ 60,\ 112,\ 105,\ 98.$$

and we implement the statistical model:

$$X_i \overset{\text{i.i.d.}}{\sim} N(\mu, \sigma^2), \qquad i = 1, \dots, 20.$$

With this set up, we can use SAS, such as through the following code

```
ODS GRAPHICS ON;

PROC TTEST H0=80 PLOTS(SHOWH0) SIDES=U ALPHA=0.1;
  VAR TIME;
RUN;

ODS GRAPHICS OFF;
```

and obtain the statistical results. The results let us know

- A point estimate of μ
- An interval estimate of μ
- A hypothesis testing under $H_0 : \mu = 80$ versus H_1,

from t-distribution for the one sample mean problem.

From the example, we may think the beginning of conducting statistical analysis procedure as

i. Formulate the statistical problem

ii. Collection the adequate dataset and think the dataset as realization of some random variables

iii. Think some statistical models (i.e., assumptions) to be considered for the random variables. This step may include some [iv]

iv. preliminary data analysis

In this lecture, we will continue the remaining two steps:

v. Model diagnostics

vi. Interpret the results

6.2 Model Diagnostics

In this step, we pause and think carefully about the assumptions underlying our analysis.

Question:

What statistical models are we implementing?

In this course, the key assumptions for the one-sample mean problem are:

- **A normality assumption**
- **An independence assumption**

The independence assumption is primarily determined by *how the data are collected*. Once the data have been sampled, this assumption is generally *not testable* from the data alone.
We therefore assume that the sampling process was conducted correctly and focus our attention on checking the *normality assumption*.

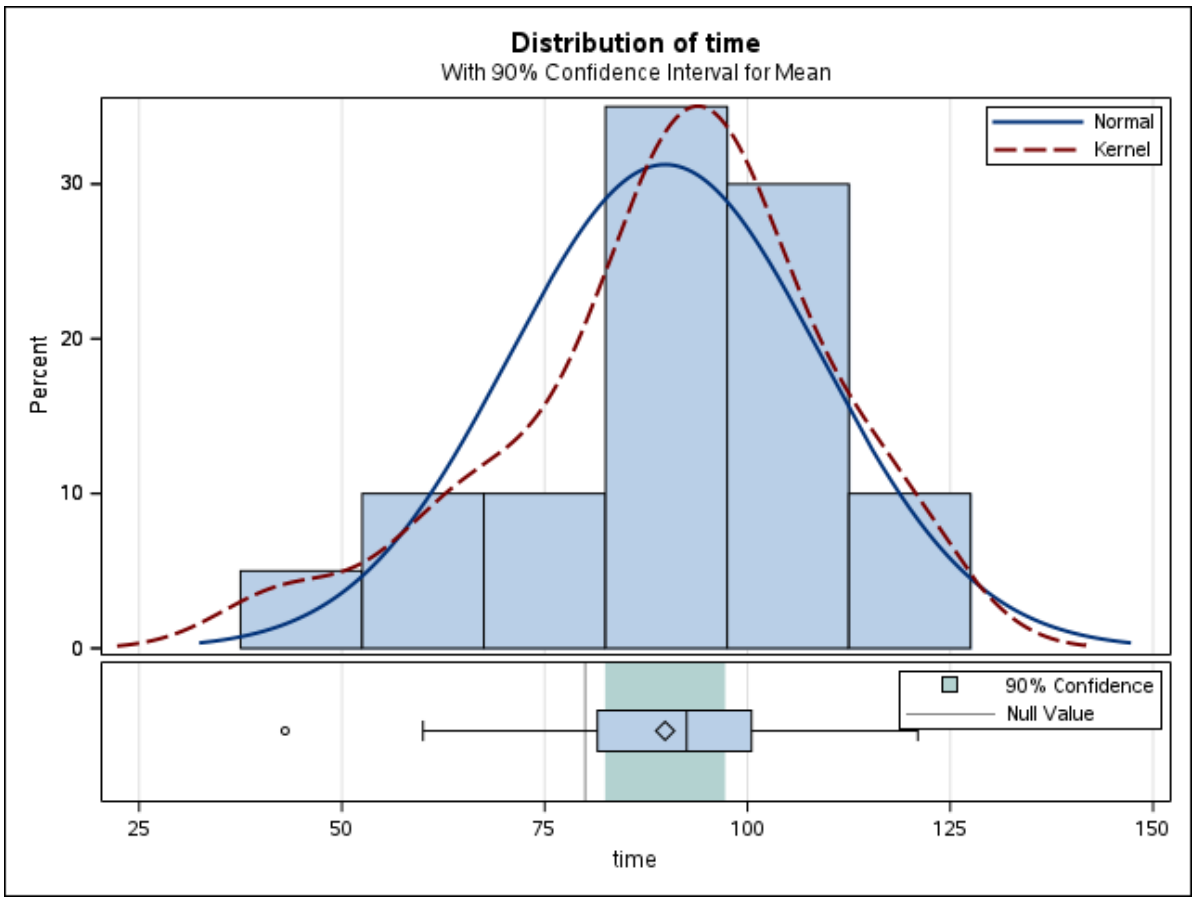
There are two broad classes of methods:

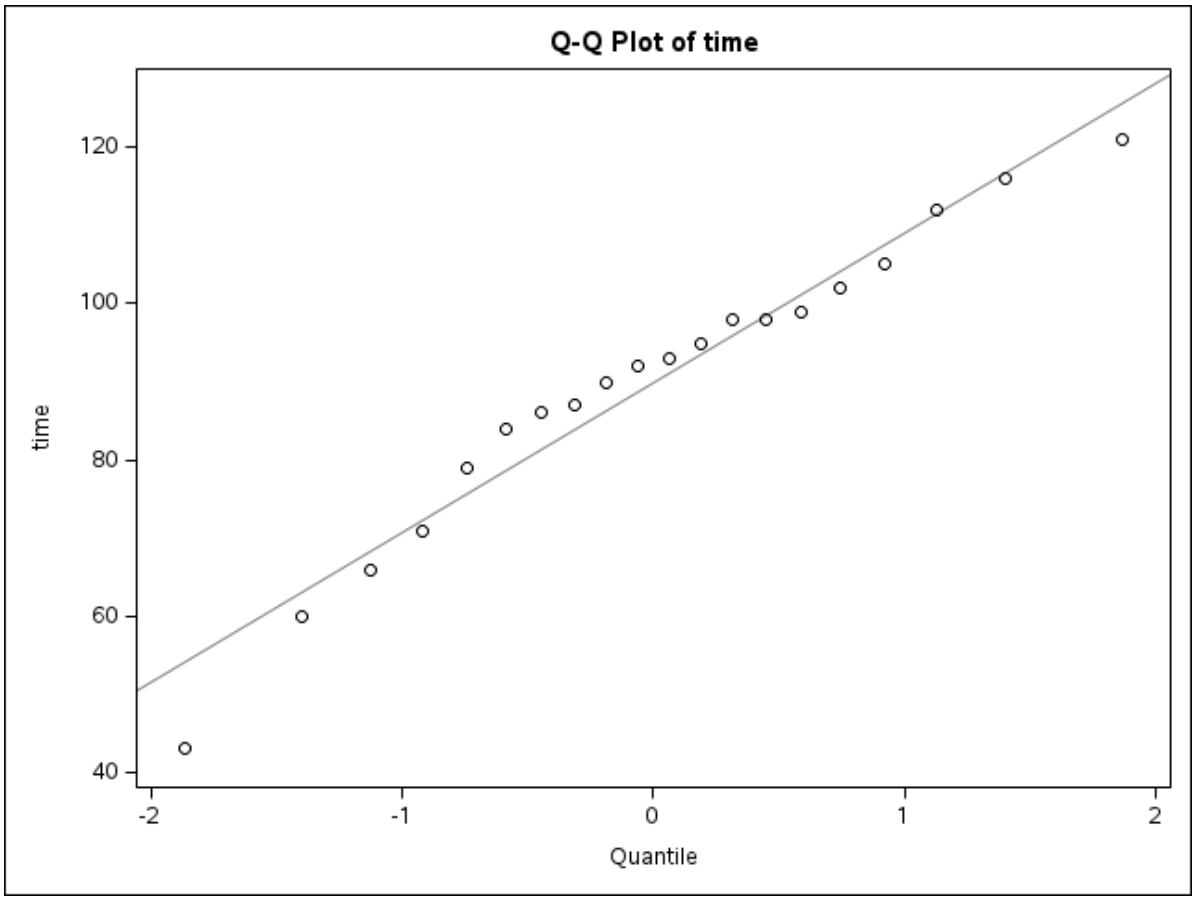
- *Graphical (visual) diagnostics*
- *Formal hypothesis testing methods*

6.2.1 Graphical Diagnostics for Normality

Common graphical tools include:

- **Histogram and Density Plots**:
  A kernel density estimate (KDE) provides a smooth estimate of the underlying probability density function, analogous to a histogram but without binning. KDEs represent the data using a continuous curve and are particularly useful for visualizing distributional shape.
- **Q–Q Plot (Quantile–Quantile Plot)**
  A Q–Q plot compares *empirical quantiles* of the observed data with *theoretical quantiles* from a normal distribution.
  If the normality assumption is reasonable, the points should fall approximately along a straight line.





6.2.2 Formal Hypothesis Testing for Normality

In addition to graphical methods, we can also assess the normality assumption using *formal numerical tests*.
One commonly used method is the **Kolmogorov–Smirnov (K–S) test**, which evaluates whether a sample plausibly comes from a specified distribution, such as the normal distribution.

The K–S test is widely used because many statistical procedures rely on the assumption that the data are normally distributed. When this assumption is violated, standard inference procedures may no longer be valid. The following step-by-step example demonstrates how to perform a Kolmogorov–Smirnov test for normality using **SAS**.

Step 1: Create the dataset

```
DATA time;
    INPUT time @@;
```

```
    DATALINES;
43 90 84 87 116 95 86 99 93 92
121 71 66 98 79 102 60 112 105 98
;
RUN;
```

Step 2: Perform the normality test

```
/* Perform Kolmogorov-Smirnov test */
PROC UNIVARIATE DATA=time;
    HISTOGRAM time / NORMAL(mu=est sigma=est);
RUN;
```

The UNIVARIATE Procedure
Fitted Normal Distribution for time

Parameters for Normal Distribution		
Parameter	**Symbol**	**Estimate**
Mean	Mu	89.85
Std Dev	Sigma	19.14563

Goodness-of-Fit Tests for Normal Distribution				
Test	**Statistic**		**p Value**	
Kolmogorov-Smirnov	**D**	0.12997262	**Pr > D**	>0.150
Cramer-von Mises	**W-Sq**	0.05410070	**Pr > W-Sq**	>0.250
Anderson-Darling	**A-Sq**	0.31058808	**Pr > A-Sq**	>0.250

Quantiles for Normal Distribution		
	Quantile	
Percent	**Observed**	**Estimated**
1.0	43.0000	45.3106
5.0	51.5000	58.3582
10.0	63.0000	65.3139
25.0	81.5000	76.9365
50.0	92.5000	89.8500
75.0	100.5000	102.7635
90.0	114.0000	114.3861
95.0	118.5000	121.3418
99.0	121.0000	134.3894

Remarks

Graphical diagnostics are informal but highly informative.
They allow us to detect skewness, heavy tails, and outliers that may invalidate normal-based inference.
Formal hypothesis tests for normality will be introduced next, but should always be interpreted in conjunction with these visual tools.

Important Note

For the best practice:
Never rely on a single normality test. Always combine numerical tests with visual inspection.

6.3 Interpretation of Results

6.3.1 Point Estimation

Point estimation is the process of using sample data to compute a single numerical value that estimates an unknown population parameter, such as the population mean.

When we report a point estimate, it is desirable to understand whether the estimator has good statistical properties. In particular, we often examine whether a point estimator is:

- **Consistent**: As the sample size increases, the estimator becomes closer to the true parameter value.
- **Unbiased**: The expected value of the estimator equals the true population parameter. For example, the sample mean is an unbiased estimator of the population mean.
- **Efficient (or best unbiased)**: Among all unbiased and consistent estimators, it has the smallest variance, meaning the estimator varies less from sample to sample.

How to rigorously verify these properties is a major topic in theoretical statistics courses. In practice, when these properties are unknown or difficult to assess, the safest interpretation of a point estimate is simply to report it with its associated unit. For example, using the court length data, we may state:

> *A reasonable estimate for the average court length is approximately 89.85 days.*

6.3.2 Confidence Intervals

A confidence interval (CI) provides a range of plausible values for an unknown population parameter. A common—but informal—interpretation of a 95% confidence interval is that we are "95% confident" the true parameter lies within the interval. While this interpretation is widely used, the **strictly correct interpretation** is based on repeated sampling:

> If the same study were repeated infinitely many times, and a 95% confidence interval were constructed each time, then approximately 95% of those intervals would contain the true parameter value.

As an example, suppose the 90% confidence interval for the population mean court length is [15.2, 26.2]. This means that, under the confidence interval procedure used, intervals constructed in this way would contain the true mean in 90% of repeated samples.

Factors Affecting Confidence Interval Width

The width of a confidence interval depends on several factors:

- **Sample size**
  Larger samples typically produce narrower (more precise) confidence intervals.
- **Variability of the outcome**
  For continuous outcomes, higher variability (larger standard deviation) leads to wider intervals.
- **Outcome type**
  - For binary outcomes, precision depends on the event probability.
  - For time-to-event outcomes, precision depends on the number of observed events.

All of these factors influence the standard error of the estimator, which directly determines the width of the confidence interval.

6.3.3 Hypothesis Testing

Hypothesis testing evaluates whether observed data are consistent with a specified statistical assumption, typically called the **null hypothesis**.

The result of a hypothesis test allows us to decide whether the assumption is supported by the data or whether there is sufficient evidence to reject it. The strength of this evidence is quantified by the *p-value*.

A **p-value** is defined as:

> The probability of observing data at least as extreme as the data actually observed, assuming the null hypothesis is true.

For example, if a hypothesis test yields a p-value of 0.01, this means that—if the null hypothesis were true—there would be only a 1% chance of observing data this extreme.

A small p-value provides evidence against the null hypothesis, while a large p-value indicates that the data are consistent with it.

If the observed data are very unlikely to occur under the conditions described by the null hypothesis, then the null hypothesis is unlikely to be true. In such cases, we reject the null hypothesis in favor of the alternative hypothesis, and the result is said to be **statistically significant**.

A commonly used decision rule is based on a significance level of 0.05. If the p-value is less than or equal to 0.05, this is typically taken as evidence against the null hypothesis, and we reject it in favor of the alternative hypothesis. However, a p-value cannot be used to prove that a hypothesis is true. Instead, it quantifies the strength of evidence against the null hypothesis.

Consider the court length data. Suppose we conduct a hypothesis test with

$$H_0 : \mu = 80 \quad \text{versus} \quad H_1 : \mu > 80,$$

and obtain a p-value of 0.0164.

If we choose a significance level of 0.05, then since $0.0164 < 0.05$,

we reject the null hypothesis. At the 5% significance level, there is strong statistical evidence that the population mean court length is **not equal to 80**.

When the p-value Is Large

If the p-value is greater than 0.05, we **do not reject** the null hypothesis. This does **not** mean that the null hypothesis is true. Rather, it means that the data do not provide strong enough evidence to conclude that the population mean court length differs from 80.

In hypothesis testing, failing to reject the null hypothesis should be interpreted as *insufficient evidence*, not as confirmation of the null hypothesis.

7 One Sample Nonparametric Test

Learning Objectives

1. Be familiar with the difference about parameter tests and nonparameteric tests
2. Applied the one sample non-parametricc test to do inference for one sample mean problem

7.1 Motivation

A **parametric test** specifies certain assumptions about the distribution of responses in the population from which the sample is drawn. The validity and interpretability of parametric test results depend critically on whether these assumptions are satisfied.

In contrast, a **nonparametric test** is based on a model that imposes only very general conditions and does **not** assume a specific parametric form for the population distribution. For this reason, nonparametric tests are often referred to as **distribution-free tests**.

Although nonparametric methods are not completely assumption-free, the assumptions they require are generally **fewer and weaker** than those of parametric tests.

7.1.1 Key Characteristics of Nonparametric Tests

Nonparametric test statistics typically rely on simple features of the data, such as:

- Signs of measurements
- Ranks or orderings
- Category or frequency counts

As a result:

- Linear transformations (stretching or compressing the scale) do **not** affect the test statistic.
- The null distribution of the test statistic can often be derived **without specifying the population distribution**.

Nonparametric tests therefore avoid assumptions such as:

- Normality
- Homogeneity of variance

Moreover, nonparametric methods usually compare **medians rather than means**, which makes them less sensitive to outliers.

Advantages of Nonparametric Tests

- Applicable to **all measurement scales**
- Particularly useful when the **sample size is very small**, unless the distribution is known
- Easier to learn and compute
- Require **fewer assumptions**
- More **robust** due to weaker modeling assumptions
- Do not require explicit population parameters
- In some cases, results can be **as exact** as parametric procedures

Disadvantages of Nonparametric Tests

- Often **less powerful** than parametric tests when parametric assumptions hold
- Provide less information about population parameters
- Interpretation is usually framed in terms of **location or rank**, not means
- Some procedures do not extend easily to complex models

Summary

- Parametric tests rely on strong distributional assumptions but can be powerful and informative.
- Nonparametric tests trade efficiency for **robustness and flexibility**.
- In practice, nonparametric methods serve as valuable alternatives when assumptions are questionable or sample sizes are limited.

In the next section, we will study **specific one-sample nonparametric procedures** and implement them in **SAS**.

7.2 One Sample Nonparametric Test in SAS

Base SAS provides two commonly used **one-sample nonparametric tests** through the `PROC UNIVARIATE` procedure:

- **Sign test**
- **Wilcoxon signed-rank test**

Both tests are designed for situations in which we want to make inference about the **location** of a population, typically interpreted as the **median** rather than the mean.

Suppose we are interested in testing whether the **median resting pulse rate of marathon runners** differs from a specified value. If the normality assumption required for a one-sample t-test is questionable, nonparametric alternatives provide a robust solution.

By default, both tests examine the hypothesis that the median of the population from which the sample is drawn is equal to a specific value,s which is zero by dafault. However, we note that

- **Wilcoxon signed-rank test**
  - Assumes the population distribution is **symmetric**
  - Generally more powerful when the symmetry assumption holds
- **Sign test**
  - Does **not** require symmetry
  - Uses only the **sign** of deviations from the hypothesized median
  - More robust but typically less powerful

Both tests can also be extended to **paired (related) samples**, which will be discussed later when we cover comparisons of two related samples.

Both the sign test and the Wilcoxon signed-rank test are **automatically available** in `PROC UNIVARIATE`.

```
/* Syntax: PROC UNIVARIATE */
PROC UNIVARIATE <options>;
    BY <variables>;
    CDFPLOT <variables> </ options>;
RUN;

CLASS variable-1 <(v-options)> <variable-2 <(v-options)>></ KEYLEVEL=value1 (value1 value2
)>;FREQ variable;HISTOGRAM <variables> </ options>;ID variables;INSET keyword-list </ options
<OUT=SAS-data-set> <keyword1=names ...keywordk=names> <percentile-options>;PPPLOT <variables>
</ options>;PROBPLOT <variables> </ options>;QQPLOT <variables> </ options>;VAR variables;WE
variable;
```

- The `PROC UNIVARIATE` statement invokes the **UNIVARIATE procedure**, which provides detailed descriptive statistics, distributional summaries, and diagnostic plots for numerical variables.
- The `VAR` Statement
  - Specifies the **numeric variables** to be analyzed.
  - **Required** if the `OUTPUT` statement is used.

  - If omitted, **all numeric variables** in the data set are analyzed.

- The PLOT statement (CDFPLOT, HISTOGRAM, PPPLOT, PROBPLOT, and QQPLOT) create graphical displays
- the INSET statement enhances these displays by adding a table of summary statistics directly on the graph.

You can specify one or more of each of the plot statements, the INSET statement, and the OUTPUT statement. If you use a VAR statement, the variables listed in a plot statement must be a subset of the variables listed in the VAR statement.

You can specify a `BY` statement to obtain separate analyses for each `BY` group. The `FREQ` statement specifies a variable whose values provide the frequency for each observation. The `ID` statement specifies one or more variables to identify the extreme observations. The `WEIGHT` statement specifies a variable whose values are used to weight certain statistics.

You can use a `CLASS` statement to specify one or two variables that group the data into classification levels. The analysis is carried out for each combination of levels in the input data set, or within each BY group if you also specify a BY statement. You can use the `CLASS` statement with plot statements to create comparative displays, in which each cell contains a plot for one combination of classification levels.

We revisit the **court length example** to demonstrate how one-sample **nonparametric tests** can be used as an alternative or complement to the one-sample t-test.

Suppose that, for some reason, we believe the t-test may not be an appropriate choice—perhaps due to concerns about normality—or we simply wish to **double-check our conclusions** using nonparametric methods. In this case, we can apply the nonparametric procedures available in `PROC UNIVARIATE`.

Step 1: Input the Data

We begin by entering the court length data into SAS.

```
DATA time;
    INPUT time @@;
    DATALINES;
43  90  84  87  116  95  86  99  93  92
121 71  66  98  79  102 60  112 105 98
;
RUN;
```

Step 2: Perform a One-Sample Nonparametric Test

To test whether the population median court length differs from 80 days, we use PROC UNIVARIATE with the MU0= option.

```
/* Perform one-sample nonparametric test */
PROC UNIVARIATE DATA=time MU0=80;
    VAR time;
RUN;
```

Step 3: Interpret the Output

From the output, SAS provides: + Test statistics + p-values for both nonparametric tests

These results allow us to assess whether there is statistical evidence that the population median court length differs from 80 days, without relying on the normality assumption required by the t-test.

7.3 Discussion: One-Sided vs Two-Sided Tests

Question:
Is this a *one-sided* or a *two-sided* test?

In this example, **SAS reports only two-sided p-values** by default for the nonparametric tests in `PROC UNIVARIATE`. If a **one-sided test** is desired, SAS does not directly provide the result.

However, a **simple (though not exact) workaround** can be used to approximate the one-sided p-value from the two-sided p-value.

7.3.1 Approximate One-Sided p-Value Calculation

1. Let

$$p^* = \frac{\text{two-sided p-value}}{2}.$$

2. Then proceed according to the alternative hypothesis:

7.3.1.1 Case 1: Right-sided test

Testing

$$H_1 : \mu > 80 \quad (\text{or } \mu > \mu_0)$$

- If the sample mean $\bar{x} > \mu_0$, then

$$\text{one-sided p-value} = p^*.$$

- If the sample mean $\bar{x} < \mu_0$, then

$$\text{one-sided p-value} = 1 - p^*.$$

7.3.1.2 Case 2: Left-sided test

Testing

$$H_1 : \mu < 80 \quad (\text{or } \mu < \mu_0)$$

- If the sample mean $\bar{x} < \mu_0$, then

$$\text{one-sided p-value} = p^*.$$

- If the sample mean $\bar{x} > \mu_0$, then

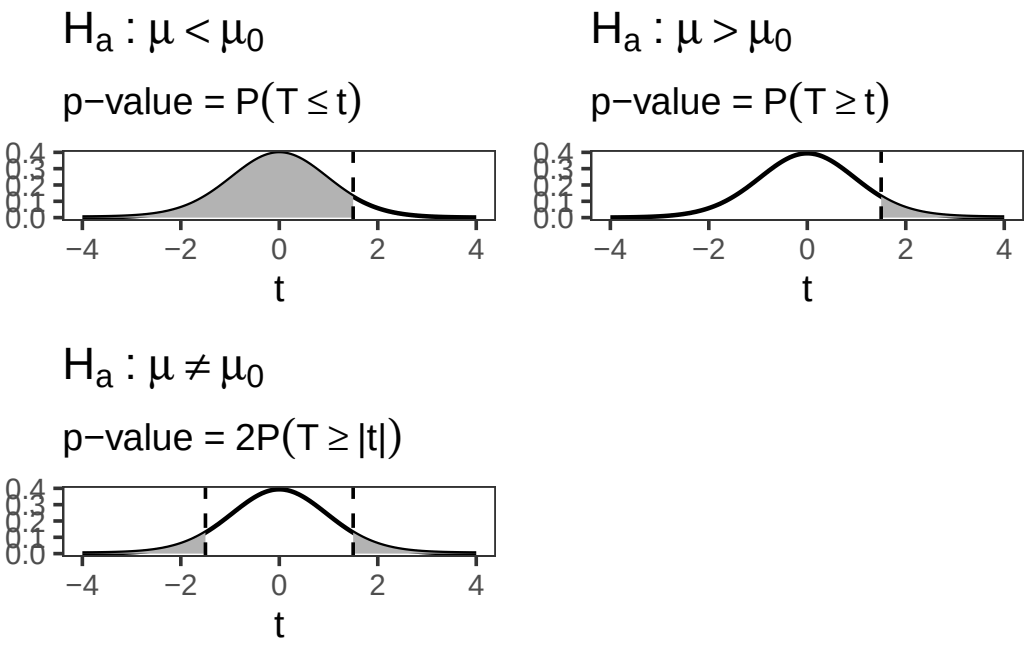
$$\text{one-sided p-value} = 1 - p^*.$$

7.3.2 Why Does This Work?

A two-sided p-value measures evidence in **both directions** away from the null hypothesis. Dividing it by two isolates the probability mass in **one tail** of the sampling distribution.

However, this adjustment is only valid when: - The test statistic is symmetric under the null hypothesis - The observed statistic is in the direction specified by the alternative

Because these conditions are not always guaranteed for nonparametric tests, this method should be viewed as an **approximation**, not an exact one-sided test.



Key takeaway:
SAS nonparametric procedures default to two-sided inference. While approximate one-sided p-values can be obtained manually, interpretation should be done with care.

8 One Sample Proportion Test

Learning Objectives

1. Apply the binomial test to conduct statistical inference for a population proportion.

8.1 Motivation

Suppose we observe a **categorical variable with two levels**, such as

- `SMOKES = 1`: smoker
- `SMOKES = 2`: non-smoker

In this setting, we are often interested in making inference about the **population proportion** of interest, such as the proportion of people who smoke.

We denote this population proportion by p.

Common inferential goals include:

- Estimating the population proportion p
- Constructing a confidence interval for p
- Performing a hypothesis test for p

Specifically, we may test

$$H_0 : p = p_0,$$

$$H_1 : \begin{cases} p \neq p_0, & \text{(two-sided)}, \\ p < p_0, & \text{(left-sided)}, \\ p > p_0, & \text{(right-sided)}. \end{cases}$$

8.1.1 Sampling Distribution of the Sample Proportion

Let $\hat{p}$ denote the **sample proportion**, computed as

$$\hat{p} = \frac{X}{n},$$

where

- X is the number of successes,
- n is the sample size.

When the sample size is sufficiently large (typically $n \geq 30$), the sampling distribution of $\hat{p}$ is approximately normal:

$$\hat{p} \sim N\left(p, \frac{p(1-p)}{n}\right).$$

This normal approximation forms the basis for both **confidence intervals** and **hypothesis testing** for a population proportion.

8.1.2 Confidence Interval for a Proportion

A $(1-\alpha) \times 100\%$ confidence interval for p is given by

$$\left[\hat{p} - Z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \quad \hat{p} + Z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right].$$

Here, $Z_{\alpha/2}$ is the standard normal quantile satisfying

$$P(Z \geq Z_{\alpha/2}) = \alpha/2, \quad Z \sim N(0,1).$$

Moreover, this normal approximation of $\hat{p}$ is also used to perform hypothesis testing for p.

8.1.3 Standard Error of the Sample Proportion

The **standard error** of $\hat{p}$ is

$$\text{SD}(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}.$$

This expression highlights an important principle:

> The uncertainty in an estimate decreases as the sample size increases.

8.1.4 Interpretation: Effect of Sample Size

Consider two examples where the estimated proportion is the same.

A small college has 8 physics majors, and 5 of them graduate within four years.

$$\hat{p} = \frac{5}{8} = 0.625$$

The standard error is

$$\text{SE}(\hat{p}) = \sqrt{\frac{0.6 \times 0.4}{8}} \approx 0.17.$$

An English department graduates 50 out of 80 students.

$$\hat{p} = \frac{50}{80} = 0.625$$

The standard error is

$$\text{SE}(\hat{p}) = \sqrt{\frac{0.6 \times 0.4}{80}} \approx 0.05.$$

Does this make sense?

> **Note**
>
> In practice, when the normal approximation may not be reliable (e.g., small n), we will instead rely on **exact binomial methods**, which do not require large-sample assumptions.

8.2 Computation

Suppose a college has six majors, labelled A, B, C, D, E, and F. For each major, we observe: + the number of students who graduated within four years (Grads), and + the total number of students in that major (Total).

```
DATA Grads;
  INPUT Major $ Grads Total @@;
  DATALINES;
A 10 22  B 10 32  C 17 25
D  4  7  E  8 14  F 16 28
;
RUN;
```

Obs	Major	Grads	Total
1	A	10	22
2	B	10	32
3	C	17	25
4	D	4	7
5	E	8	14
6	F	16	28

It is easy to write a short DATA step to compute the empirical proportions and a 95% confidence interval for each major.

```
DATA GradRate;
  SET Grads;

  /* Empirical proportion */
  p = Grads / Total;

  /* Standard error under normal approximation */
```

```
  StdErr = SQRT(p * (1 - p) / Total);

  /* 95% Wald confidence interval */
  z = QUANTILE("normal", 1 - 0.05/2);

  LCL = MAX(0, p - z * StdErr);   /* Lower bound */
  UCL = MIN(1, p + z * StdErr);   /* Upper bound */

  LABEL p   = "Proportion"
        LCL = "Lower 95% CI"
        UCL = "Upper 95% CI";
RUN;
```

Major	Grads	Total	Proportion	Lower 95% CL	Upper 95% CL
A	10	22	0.45455	0.24648	0.66261
B	10	32	0.31250	0.15190	0.47310
C	17	25	0.68000	0.49714	0.86286
D	4	7	0.57143	0.20483	0.93803
E	8	14	0.57143	0.31220	0.83065
F	16	28	0.57143	0.38813	0.75473

The output shows that although majors D, E, and F have the same four-year graduation rate (57%), the estimate for the D group, which has only seven students, has twice as much variability as the estimate for the F group, which has four times as many students.

8.3 Automating the Computations with PROC FREQ

In the previous section, we computed the Wald confidence interval for each major using a DATA step. That approach is straightforward, but it becomes inconvenient if we want other binomial confidence intervals (e.g., exact/Clopper–Pearson, Wilson, etc.). A convenient alternative is to use `PROC FREQ` with the `BINOMIAL` option. However, PROC FREQ expects the data in an event / nonevent format (a binary outcome with a frequency count), rather than the events / trials format (Grads, Total) we currently have.

So we first convert each major into two rows: + Graduated="Yes" with Count = Grads + Graduated="No" with Count = Total - Grads

Step 1: Convert events/trials into event/nonevent format

```
/*-----------------------------------------------------------------
  Convert Event/Trials format (Grads, Total)
  into Event/Nonevent format (Graduated, Count)
-----------------------------------------------------------------*/
data GradFreq;
  set Grads;

  Graduated = "Yes";
  Count = Grads;
  output;

  Graduated = "No";
  Count = Total - Grads;
  output;
run;

proc print data=GradFreq;
run;
```

Obs	Major	Grads	Total	Graduated	Count
1	A	10	22	Yes	10
2	A	10	22	No	12
3	B	10	32	Yes	10
4	B	10	32	No	22
5	C	17	25	Yes	17
6	C	17	25	No	8
7	D	4	7	Yes	4
8	D	4	7	No	3
9	E	8	14	Yes	8
10	E	8	14	No	6
11	F	16	28	Yes	16
12	F	16	28	No	12

Step 2: Run PROC FREQ to compute binomial CIs by major

```
/*-------------------------------------------------------------
  Use PROC FREQ to analyze each major separately and compute
  binomial confidence intervals for Pr(Graduated="Yes")
-------------------------------------------------------------*/
proc freq data=GradFreq noprint;
  by notsorted Major;

  tables Graduated / binomial(level="Yes" CL=wald);
  weight Count;

  output out=FreqOut binomial;
run;
```

Step 3: Print a clean summary table

```
proc print data=FreqOut noobs label;
  var Major N _BIN_ L_BIN U_BIN;

  label _BIN_ = "Proportion"
        L_BIN = "Lower 95% CI"
        U_BIN = "Upper 95% CI";
run;
```

Major	Number of Subjects	Proportion	Lower 95% CI	Upper 95% CI
A	22	0.45455	0.24648	0.66261
B	32	0.31250	0.15190	0.47310
C	25	0.68000	0.49714	0.86286
D	7	0.57143	0.20483	0.93803
E	14	0.57143	0.31220	0.83065
F	28	0.57143	0.38813	0.75473

8.4 Visualization of Binomial Proportion

It is often helpful to visualize estimated proportions together with their confidence intervals. When plotting several proportions on the same graph, it is a good idea to **sort the groups** in

a meaningful way—most commonly by the estimated proportion itself. If two groups have the same estimated proportion, the sample size can be used as a tie-breaker.

8.4.1 Why visualization matters

A single number rarely tells the whole story. Plotting proportions with confidence intervals allows us to:

- Compare graduation rates across majors at a glance
- Assess uncertainty in each estimate
- See how sample size affects precision

Groups with **smaller sample sizes** tend to have **wider confidence intervals**, reflecting greater uncertainty.

8.4.2 Adding a reference line

It is often informative to add a **reference line** representing the **overall proportion**, regardless of group membership.

For the graduation data in this example, the overall proportion of students who graduate in four years is

$$\hat{p}_{\text{overall}} = \frac{65}{128} \approx 0.5078.$$

This reference line helps contextualize each major's graduation rate relative to the overall average.

> **Tip:**
> You can obtain the overall proportion in SAS by repeating the `PROC FREQ` analysis **without** a `BY` statement.

8.4.3 Including sample size on the plot

Because uncertainty depends strongly on sample size, it is good practice to display the **number of observations per group** directly on the graph.

In SAS, this can be done using the `YAXISTABLE` statement, which adds a table aligned with the y-axis showing the sample size for each group.

8.4.4 Interpreting the plot

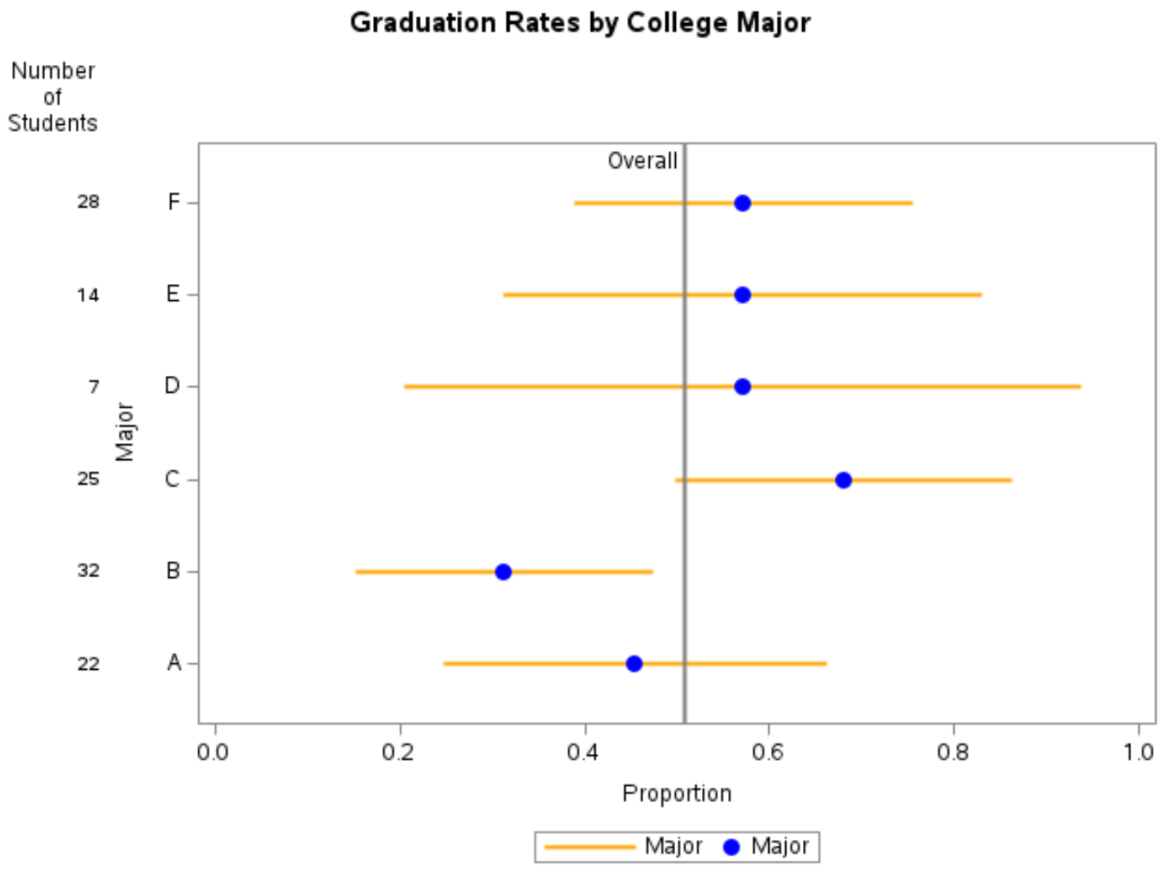
A typical plot of binomial proportions with confidence intervals shows:

- **Points**: estimated proportions
- **Error bars**: 95% confidence intervals
- **Vertical reference line**: overall proportion
- **Table entries**: number of students in each major

Such a graph clearly illustrates that majors with fewer students (for example, very small departments) have much wider confidence intervals than majors with larger enrollments.

8.4.5 Practical visualization advice

- If there are **10 or more categories**, consider using **alternating background color bands** to make it easier to associate confidence intervals with groups.
- Always label axes clearly (e.g., *Proportion* on the x-axis).
- Include the confidence level (e.g., "95% CI") in the caption or title.



8.4.6 Summary

This section demonstrates how visualization complements numerical output:

- `PROC FREQ` provides estimated proportions and confidence intervals.
- Graphing proportions with confidence intervals makes comparisons intuitive.
- Including sample sizes helps readers understand why some intervals are wider than others.

Well-designed graphics are one of the most effective ways to communicate results from binomial proportion analyses.

9 Writing SAS Macro Program: Using One Sample Variance Test/CI as Example

Learning Objectives

1. Apply hypothesis testing and confidence interval construction for **one population variance**.
2. Use this example to motivate **writing reusable SAS macro programs**.

9.1 Motivation

In many applications, we may care more about the **population variance** σ^2 than the population mean μ. For instance, in a manufacturing process, a **large variance** in product measurements indicates unstable quality, even if the mean is acceptable. This is not the only instance, as there are many interesting scientific questions that involve the population variance both in academia and in industry.

9.1.1 Examples

A SCUBA instructor records the dive depths of students during checkout. Although everyone should be at the same depth, the instructor is interested in how much the depths vary.

The instructor believes the standard deviation is **3 feet**, while the assistant thinks it is **less than 3 feet**.

The hypotheses are:

$$H_0 : \sigma^2 = 3^2 \qquad \text{vs} \qquad H_1 : \sigma^2 < 3^2$$

A company produces metal pipes of a standard length. Twenty years ago, the pipe lengths were normally distributed with standard deviation **1.1 cm**.

The company now tests a random sample of **30 pipes** and wants to construct a **95% confidence interval** for σ^2 to determine whether the production quality has changed.

For the examples above, the quantity of the interest is the population variance. But we need something to estiamte it. Let s^2 denote the sample variance. Then the pivotal quantity

$$\frac{(n-1)s^2}{\sigma^2},$$

which follows a chi-square distribution:

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1},$$

where

- n : sample size
- s^2 : sample variance
- σ^2 : population variance

You may think that s as the randomvariable in this test. The distribution is called chi-square distribution which depends on a parameter named the *degrees of freedom* denoted by $\text{df} = n-1$. A test of a single variance may be right-tailed, left-tailed or two-sided. This chi-square result is the foundation for both **confidence intervals** and **hypothesis tests** for σ^2.

> **Note**
>
> A $(1-\alpha) \times 100\%$ confidence interval for the population variance is
>
> $$\left(\frac{(n-1)s^2}{\chi^2_{1-\alpha/2,\, n-1}}, \frac{(n-1)s^2}{\chi^2_{\alpha/2,\, n-1}} \right),$$
>
> where $\chi^2_{p,\, n-1}$ denotes the p-th quantile of the χ^2_{n-1} distribution.
> I won't expect you to know the expression here for now.

Problem

Unfortunately, if you check the built-in SAS procedures, you will find that **there is no basic procedure** that directly handles the **one-sample variance inference problem**.

The good news is that SAS provides a powerful **MACRO programming language**, which allows us to **write our own SAS procedures**.

9.2 SAS Macro Program

When you write a program that will be run **repeatedly**, you may want to seriously consider using **macros**, because:

- **MARCROS allow centralized changes**:
  You can make a change in one location, and SAS will cascade that change throughout your program.
- **MACROS promote code reuse**:
  You can write a section of code once and reuse it many times, either within the same program or across different programs.
- **MARCROS enable data-driven programming**:
  SAS can decide what actions to take based on actual data values.

In general, macro code takes **longer to write and debug** than standard SAS code. Therefore, macros are usually **not recommended** for programs that will only be run a few times.

However, if you find yourself writing **similar code repeatedly**, macros can significantly improve efficiency and reduce errors.

Macros can help in several ways:

1. You can make a **single small change**, and SAS will propagate that change throughout your program.
2. You can **reuse code blocks**, avoiding duplication.
3. You can build **data-driven programs**, allowing SAS to adapt automatically to different datasets or inputs.

9.3 How Macros Work

When you submit a standard SAS program, SAS **compiles and executes it immediately**. When you write macro code, there is an **additional step**:

- SAS first sends MARCO statements to the **macro processor**.
- The macro processor **resolves the macro code**, generating standard SAS code.
- SAS then compiles and executes the generated code.

Because you are writing a program that **writes another program**, this process is sometimes referred to as **meta-programming**.

9.4 Designing Your Own Macros

To design effective SAS macros, we need to understand two fundamental concepts:

9.4.1 Macros vs. Macro Variables

As you construct macro programs, you will work with two basic building blocks:

- **MACROS** v.s.
- **MACRO variables**

We will study each of these in detail in the next section.

9.5 Macros vs. Macro Variables

To design your own SAS macros, we need to clearly distinguish between **macros** and **macro variables**.
These are the two basic building blocks of SAS macro programming.

9.5.1 How the Macro Processor Works

Conceptually, the workflow looks like this:

- You write **macro code**
- The **macro processor** resolves it
- The output is **ordinary SAS code**, which is then compiled and executed

---

9.5.2 Naming Conventions

You can tell macros and macro variables apart by their prefixes:

- **Macro variables** start with an ampersand: `&`
- **Macros** start with a percent sign: `%`

Examples: - `&alpha`, `&n` - `%myMacro`, `%DO`, `%IF`

9.5.3 Macro Variables

A **macro variable** is similar to a data variable, but with important differences:

- It does **not belong to a dataset**
- It has **only one value**
- That value is **always character**
- The value is **substituted directly into your program**

A macro variable's value can be: - A variable name - A number - Text - Any string you want substituted into your SAS code

---

9.5.4 Macros

A **macro** is a larger unit of code that can contain:

- DATA steps
- PROC steps
- Macro logic such as
  `%IF-%THEN-%ELSE`, `%DO-%END`

Macros often—but not always—use macro variables.

9.6 Think Globally and Locally: Scope of Macro Variables

Macro variables come in two scopes:

- **Local macro variables**
- **Global macro variables**

9.6.1 Local Macro Variables

- Defined **inside a macro**
- Exist **only within that macro**
- Cannot be referenced outside the macro

9.6.2 Global Macro Variables

- Defined in **open code** (outside any macro)
- Can be used **anywhere in the program**

9.6.3 Common Mistakes to Avoid

- Using a local macro variable outside its macro
- Accidentally creating **both local and global macro variables with the same name**

Keeping scope in mind will save you a lot of debugging time.

9.7 You May Quote Me on That

The macro processor **does not resolve macro references inside single quotes**.

- `'&var'` → macro not resolved
- `"&var"` → macro resolved

Rule of thumb:
Use **double quotes** when quoted strings contain macro variables.

9.8 Substituting Text with %LET

The `%LET` statement assigns a value to a macro variable.

```
%LET alpha = 0.05;

%MACRO macro_name(param1, param2, ..., paramk);
   /* macro code */
%MEND macro_name;
```

9.9 3. A Simple Example of Writing SAS Macros

Before writing our own macros, let us first understand how **macro variables** work through a simple example.

9.9.1 Example 1: Global Macro Variable

SAS provides a built-in **global macro variable** called `SYSDATE`, which represents the system date.

Suppose we want to print the system date automatically in the title of a SAS report every time the report is generated. The advantage is that we do **not** need to manually update the date in the code.

We use the built-in SAS dataset `CARS` from the `SASHELP` library.

```
PROC PRINT DATA=SASHELP.CARS;
  WHERE MAKE = 'Audi' AND TYPE = 'Sports';
  TITLE "Sales as of &SYSDAY &SYSDATE";
RUN;
```

Example 1 (Continued): Local Macro Variables

Macro variables are referenced in SAS statements using the ampersand (&) character.

Now suppose we want our program to be more flexible. Instead of hard-coding the car MAKE and TYPE, we define macro variables so that we can easily change them without modifying multiple lines of code.

```
%LET MAKE_NAME = Audi;
%LET TYPE_NAME = Sports;

PROC PRINT DATA=SASHELP.CARS;
  WHERE MAKE = "&MAKE_NAME" AND TYPE = "&TYPE_NAME";
  TITLE "Sales as of &SYSDAY &SYSDATE";
RUN;
```

Reusing the Same Program with Different Values

Now, suppose we want to view Audi vehicles of type Wagon. We only need to change one line of code.

```
%LET MAKE_NAME = Audi;
%LET TYPE_NAME = Wagon;

PROC PRINT DATA=SASHELP.CARS;
  WHERE MAKE = "&MAKE_NAME" AND TYPE = "&TYPE_NAME";
  TITLE "Sales as of &SYSDAY &SYSDATE";
RUN;
```

9.9.2 Example 1 (Continued): Writing a Macro Program

So far, we have used macro variables to make our SAS programs more flexible.
Now we take one more step forward and define a **macro program**, which is a reusable block of SAS code.

9.9.2.1 Defining a Macro

The following program defines a macro called `show_result`.
This macro takes two arguments: - `make_`: the car manufacturer - `type_`: the vehicle type

```
%MACRO show_result(make_, type_);

PROC PRINT DATA=SASHELP.CARS;
  WHERE MAKE = "&make_" AND TYPE = "&type_";
  TITLE "Sales as of &SYSDAY &SYSDATE";
RUN;

%MEND;
```

Calling a Macro Once a macro is defined, it can be called just like a function.

```
%show_result(BMW, SUV);
```

Why Use Macro Programs? Macro programs allow you to: • Avoid rewriting the same code repeatedly • Centralize logic in one place • Make programs easier to maintain and extend • Write data-driven and parameterized SAS programs

This idea becomes especially powerful when we need to repeat the same analysis for many datasets or parameter values.

4. Revisit the One-Sample Variance Test

As discussed earlier, SAS does not provide a built-in procedure for conducting one-sample variance inference (confidence intervals or hypothesis tests).

However, SAS allows us to overcome this limitation by writing our own macro procedures.

Fortunately, a one-sample variance inference macro is commonly used and can be written in a general form. In this course, we will use a macro named vartest.sas, which is shared in the course materials.

In the next section, we will: • Examine the structure of the variance-test macro • Understand how macro parameters control the analysis • Apply the macro to real data examples

---

10 Topic 9: Writing SAS Macro Programs

10.1 One-Sample Variance Test and Confidence Interval

STAT 8678 — SAS Programming
Spring 2026

10.2 Learning Objectives

By the end of this lecture, you should be able to:

- Understand **why SAS macros are needed** for certain inference problems
- Distinguish between **macros** and **macro variables**
- Explain how the **SAS macro processor** works
- Write and interpret a **simple SAS macro**
- Use a macro to perform **one-sample variance inference**, including:
  - Hypothesis testing
  - Confidence intervals

10.3 1. Motivation: Why Care About Variance?

In many applications, **variability** is more important than the mean.

Examples include:

- **Manufacturing**: Large variance in product dimensions indicates unstable quality.
- **Environmental monitoring**: High variance may signal abnormal conditions.
- **Education or training**: Inconsistent outcomes may be more concerning than average performance.

We denote the population variance by σ^2.

10.3.1 Example 1: SCUBA Diving Depths

An instructor records dive depths for students:

- The instructor believes the **standard deviation is 3 feet**.
- The assistant believes the variability is **less than 3 feet**.

The hypotheses are:

$$H_0 : \sigma^2 = 3^2 \qquad \text{vs.} \qquad H_1 : \sigma^2 < 3^2$$

10.3.2 Example 2: Manufacturing Quality

- Pipe lengths were historically normally distributed.
- The standard deviation was **1.1 cm**.
- A random sample of $n = 30$ pipes is collected.
- The goal is to construct a **95% confidence interval for** σ^2.

10.4 2. Statistical Theory Behind the Test

If the data are normally distributed, the following pivotal quantity holds:

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$$

where:

- n is the sample size
- s^2 is the sample variance

This result allows us to construct both:

- **Hypothesis tests**
- **Confidence intervals** for σ^2 and σ

10.4.1 Confidence Interval for Variance

A $(1-\alpha) \times 100\%$ confidence interval for σ^2 is:

$$\left[\frac{(n-1)s^2}{\chi^2_{1-\alpha/2}}, \frac{(n-1)s^2}{\chi^2_{\alpha/2}} \right]$$

A confidence interval for σ is obtained by taking square roots of the bounds.

10.5 3. The Problem in SAS

There is **no built-in SAS procedure** for one-sample variance testing or confidence interval construction.

- No `PROC VARTEST`
- No direct equivalent to `PROC TTEST` for variance

Solution: Write our own procedure using **SAS macros**.

10.6 4. What Is a SAS Macro?

A **macro** is a program that **writes SAS code**.

Key idea:

> You are writing a program that generates another program.

This is sometimes called **meta-programming**.

10.6.1 Why Use Macros?

Macros allow you to:

- Make changes in **one location** that affect the entire program
- **Reuse code** without copy-and-paste
- Write **data-driven** programs
- Automate repetitive analyses

10.7 5. How the SAS Macro Processor Works

Conceptually:

Macro statements ↓ Macro processor ↓ Standard SAS code

Execution steps:

1. SAS sends macro code to the **macro processor**
2. Macros are **resolved**
3. The resulting SAS code is compiled and executed

10.8 6. Macros vs. Macro Variables

10.8.1 Macro Variables

- Begin with `&`
- Always **character values**
- Do **not** belong to a dataset
- Used for **text substitution**

Example:

```
%LET alpha = 0.05;
```

Macros • Begin with % • Can contain: • DATA steps • PROC steps • Macro logic (%IF, %DO, %END) • Often use macro variables internally

Example:

%MACRO mymacro(); PROC PRINT DATA=example; RUN; %MEND;

7. Scope: Global vs. Local Macro Variables

Macro variables can be: • Global: defined outside macros • Local: defined inside macros

Rules: • Global macro variables can be used anywhere • Local macro variables can only be used inside the macro that defines them

Common mistakes include: • Using a local macro variable outside its macro • Accidentally creating local and global variables with the same name

8. Substituting Text with %LET

```
%vartest(
  version = 1.1,
  data    = mydata,
  var     = response,
  alpha   = 0.05,
  sigma0  = 1.1
);
```

11 χ^2 Goodness of Fit Test

References