Production forecasting in semiconductor manufacturing

Mario Ricardo Cruz Duarte – Project Report

Technische Universität Dresden
Master'sprogram in ComputationalLogic
Nöthnitzer Str. 46, 01187 Dresden

SYSTEMA GmbH
Manfred-von-Ardenne-Ring 6,
01099 Dresden, Germany

Table of Contents

List of figures 4
Abstract....7
Motivation....8
Introduction....9
Production of semiconductors....9
Oxidation....10
Photolithography....11
Etching....12
Implantation....12
Deposition....13
Semiconductor manufacturing tools....14
Key Performance Indicator....14
Work-in-process....15
Cycle time....15
Raw process time....15
Tool types....17
Single wafer tools....17
Batch tools....17
Cluster tools....17
Factory simulation....18
Industry 4.0....18
The fourth industrial revolution....19
Holistic Information Model....21
Holistic Information Model Atomics....22
Holistic Information Model Fundamentals....22
Holistic Information Model Fractals....23
Hypothesis....24
Atomics....25
Fundamentals....26
Fractals....27
Production forecasting - and forecasting in general....29
Forecasting....29
Time series....30
The difference between discrete time and continuous time....30
Seasonality....31
Trend....32
Stationary time series....32
Autocovariance....33
Correlation....36
Standard deviation....38
Linear regression....41
Naïve forecasting....47
Mean forecasting....47
Drift forecasting....47

ARMA....48
Autoregression....48
Moving Average....52
ARIMA....53
Auto ARIMA function....57
Granger causal....58
VAR....60
MARS....60
An holistic approach to production forecasting....62
Computer Integrated Manufacturing....62
Holistic Production Forecasting model....64
Holistic Production Forecasting model v2....65
Data analysis....65
Data forecast....68
Forecast using VAR....69
RMSE....71
Forecast using MARS....72
Conclusions and future work....76
Sources....77

List of figures

Figure 1.1: Silicon rod and silicon discs (left), a finished wafer from Global Foundries Dresden with IC ready to be packed and shipped (Right).
Figure 1.2 IC sliced from a finished wafer and used in a DIP switch.
Figure 1.3 shows the steps to create a finished wafer.
Figure 1.4. A silicon wafer with silicon dioxide layer grown on top of it
Figure 1.5. Photoresist is grown on the silicon dioxide layer, the photoresist is then “burned“ with the IC mask pattern by applying intense UV light to it.
Figure 1.6. A wafer with a layer of dioxide and a photoresist, where the latter follows a pattern of an IC design.
Figure 1.7. Wafer with silicon dioxide layer on it, together with photoresist layer.
Figure 1.8. Wafer with silicon dioxide layer on it. No photoresist
Figure 1.9. An NPN transistor, where current flows from the source to the drain, but only if the gate allows it. The gate allows for such a flow when an electrical current is applied to the gate.
Figure 1.10. A wafer with several layers of silicon on top of it. Depending on the design, the silicon areas might have different electrical properties.
Figure 1.11. Strip of ICs.
Figure 1.12. A table obtained from [A-34] which shows the KPIs in semiconductor manufacturing.
Figure 1.13. A hammer tool and its operations
Figure 1.14. A cluster tool.
Figure 1.15. The 4 industrial revolution stages (so far)
Figure 1.16: Technological *causals* that generated the industry 4.0 trend
Figure 1.17. Template for the HIM
Figure 1.18. Fundamentals
Figure 1.19. Fractals
Figure 1.20. Atomics as seen by RI
Figure 1.21. Fundamentals as seen by RI
Figure 1.22. Fractals as seen in an MS-SQL database
Figure 1.23. WIP obtained from Fractals
Figure 2.1. Time series for Fatal Traffic Injuries among Iranian Drivers; 2004–2010
Figure 2.2. A continuous and discrete signal with the same measuring boundaries
Figure 2.3. A time series showing the amount of passengers for an airline. Monthly Airline Passenger Numbers 1949-1960 by Box & Jenkins
Figure 2.4. Trend of airline passengers
Figure 2.5. Sample mean
Figure 2.6. Population mean.
Figure 2.7. The variance is obtained by subtracting the value X from the mean μ and squaring the result.
Figure 2.8. Above: a time series plot of 100 random observations concealing a sine function.
Below: the sine function revealed in a correlogram produced by autocorrelation.
Figure 2.9. Covariance function
Figure 2.10. The autocorrelation function plot
Figure 2.11. Left: Air passenger’s data set. Right: ACF plot of the air passenger’s data set.
Figure 2.12. 5 numerical vectors which we will use for correlation.
Figure 2.13. Plot of the 5 vectors from figure 2.11.
Figure 2.14. Left: total positive correlation. Right: total negative correlation.
Figure 2.15. Correlation map
Figure 2.16. Correlation map using ellipses. Where, as difference between vectors appears, the blue diagonal line (correlation of 1) starts getting thicker until it becomes a light white colored circle (correlation of 0), then it flips to the left until it becomes a red diagonal line towards the left side (correlation of -1)
Figure 2.17. Dogs and their height
Figure 2.18. Dog heights with standard deviation.
Figure 2.19. A normal distribution plot, where each band has a width of 1 standard deviation.
Figure 2.20. A data set containing a relation of the MPG and weight of some cars.
Figure 2.21. Plot of the MPG by weight with trend line.
Figure 2.22. Regression formula
Figure 2.23. Summary of a linear regression model.
Figure 2.24. Residuals.
Figure 2.25. Coefficients summary.
Figure 2.26. Summary of a linear regression model.
Figure 2.27. A figure showing the 3 forecasting methods explained above.
Figure 2.28. Regression model formula.

Figure 2.29. Left: Plot of closing stocks data. Right: Closing stocks data set.
Figure 2.30. Partial Autocorrelation Function plot, where the PACF value cuts off after 1 lag.
Figure 2.31. Summary of an autoregressive function.
Figure 2.32. Residuals of the AR model for closing stock prices.
Figure 2.33. Prediction call on the AR model for the next 10 stock prices.
Figure 2.34. Forecast of closing stock prices.
Figure 2.35. This figure shows the final *X'* values with an MA forecast starting at the vertical line. The Upper/Lower lines represent the confidence level of the forecast.
Figure 2.36. Air passengers time series data, measured monthly.
Figure 2.37. PACF and ACF plots for air passenger's data.
Figure 2.38. ACF and PACF plots of the integrated time series.
Figure 2.39. ACF and PACF plots of the integrated time series.
Figure 2.40. Call to the ARIMA function in R together with prediction.
Figure 2.41. Forecast for the next 24 months of the air passengers data.
Figure 2.42. Auto ARIMA function with the air passengers data.
Figure 2.43. New ARIMA model with new order for p, d, q parameters.
Figure 2.44. Left: ARIMA model with order of (2,1,1) using auto arima function.
Right: ARIMA model with order of (2,1,2) using ACF and PACF plots to obtain parameter values.
Figure 2.45. Sum of the residuals, where the ARIMA object *fit*, the one created using the auto.arima function is much more accurate (although not by much) than the one we obtained using the PACF and ACF functions the *fit_pa*.
Figure 2.46. X is a causal of Y.
Figure 2.47. Chicken and egg data set, where the chicken column represents the chicken population per year, and the egg column represents the amount of eggs in millions of dozens.
Figure 2.48. Granger causal test on the chicken&egg data set.
Figure 3.1.The SEMATECH CIM Framework for semiconductor manufacturing.
Figure 3.2. Dynamic data processing by the HIM.
Figure 3.3. Holistic Production Forecasting model.
Figure 3.4. New Holistic Production Forecasting model, where the tools capable of performing each operation are added into the model, together with the forecast for each tool.
Figure 3.5. Fractal information for operation WIP.
Figure 3.6. Data transformation using function *dcast*.
Figure 3.7. No missing values.
Figure 3.8. Plot of WIP for operations.
Figure 3.9. Correlation file for tool ASM_A2.
Figure 3.10. Linear model concept for our correlation and causal analysis.
Figure 3.11. Call to VAR with 16 regressors.
Figure 3.12. Predict the next 10 hours with a confidence interval of 95%.
Figure 3.13. Forecast of next 10 hours using VAR model.
Figure 3.14. Forecast for tool ASM_A2 using VAR.
Figure 3.15. Call to RMSE function in RStudio, where the predicted data is on the second column of the aux_asm table, and the observed data is on the first column of the same table.
Figure 3.16. Call for standard deviation on observed values.
Figure 3.17. Call to the earth function.
Figure 3.18. Know the future to make predictions.
Figure 3.19. Predict on the predictors using VAR.
Figure 3.20. Call on predict function with newdata parameter.
Figure 3.21. Forecast using MARS.
Figure 3.22. RMSE of predicted values using MARS compared with observed data.
Figure 3.23. (Above) Real WIP for 500 hours. Forecasted WIP for 500 hours (below)
Figure 3.24. Comparison between real and forecasted using a correlation function.

"An unsophisticated forecaster uses statistics as a drunken man uses lamp-posts - for support rather than for illumination."

--After Andrew Lang

Abstract

Semiconductor manufacturers are continuously ramping up production capacity and intensively searching for ways to optimize its usage. One important tool for production optimization and planning is the forecasting of future production performance. The current state-of-the-art for forecasting production is fab simulation, where different production scenarios are simulated in order to predict possible outcomes. This however consumes large amounts of time and computing resources and does not offer a continuously updated real-time representation of production forecasting. As the industry moves into new challenges like high diversification of products and services, production must be as flexible and agile as possible which requires the continuous creation and update of accurate forecasts.

In order to achieve this goal, production forecasting must replace complex and heavyweight simulations with simpler lightweight statistical forecast algorithms. While these methods are faster and more agile, they work in a different way than simulations. While a simulation can cover whole production scenarios, statistical forecast models have a more specific scope and typically focus on a single measure, e.g. a key performance indicator. As a consequence, multiple forecast models must be used in combination to achieve not only real-time capabilities but also coverage for whole production scenarios. The goal of this thesis is to analyze and select forecasting methods for production performance indicators and develop concepts for a multi-model forecasting system.

Motivation

Population keeps growing and threatens to continue to do so indefinitely, this increases the demand for factories to ramp up production, increase the output and quality of their products while still trying to maintain profits. It is imperative for companies to evolve and adapt to new technologies in order to find a way to satisfy this demand.

One of the many techniques that can be used to help beat this challenge is big data [3], which for over the past 20 years has been a key element with which manufacturers are able to gather massive amounts of data from within their factory site and use advanced analytics in that data to find key performance indicators (KPIs) which would help reduce costs, increase yield, quality and forecast output dates [4].

Most of the times, like the bad guy in a film, the KPIs hide in plain sight, the operation managers take previously isolated data sets into the pool of data and analyze them, often finding new and important information that can help improve the final yield in a production line [4].

One of the most commonly used techniques in the industry lately is the Internet of Things (IoT), where devices share and store data with each other, while also allowing for wireless factory management which helps keep human intervention to a minimum. This can be further optimized by interconnecting devices using sensors and actuators which kicks things off into a psi-fi like state where we could have fully automated factories, smart vehicles and even smart cities. IoT is estimated to consist of up to 50 billion objects in 2020 [5]

As we move forward into a new age of industry, we find ourselves within the well-known trend of Industry 4.0, this, together with big data and IoT can be exploited into completing our original challenge, a holistic approach to production forecasting [1] [2].

The structure of this thesis is as follows, first we will introduce some basic concepts about semiconductor manufacturing in chapter 1, this will be done in order to make the thesis easier to read together with chapter 2 where we will introduce the basics for statistical analysis and forecast in order to acclimate the reader to the concepts used in the solution of the challenge at hand, in chapter 3 we will explain the solution for the current challenge as well as animplementation of a new concept for holistic production forecasting, we will end this thesis in chapter 4 by giving our conclusions and future work.

Introduction

In this section, we will introduce the current state of the art regarding semiconductor manufacturing, together with some definitions which will make for an easier reading and understanding of the thesis. There exist no requirements for the understating of the thesis. Everything found within it will help the reader understand the topics with simple definitions, examples and sources for further reading.

The production of semiconductors[A-1][A-1.1][A-2.2][A-2.3]is one of the most complex operations that we as humans have come up with. In Layman's terms, it starts by taking sand and turning it into silicon, then through a series of steps and regulations, the silicon is turned into a wafer disc (Fig. 1.1 Left). From there, the wafer goes through several hundreds of processes such as polishing, circuit printing via UV light testing and eventually a finished product (Fig. 1.1 Right). This whole process can take up to 8 weeks and is done in a room thousands of times cleaner than the average hospital operation room.

Figure 1.1: Silicon rod and silicon discs (left), a finished wafer from Global foundries Dresden with IC ready to be packed and shipped (Right).

The finished wafer disc is then sliced, allowing for the single units which have passed all the regulations, to be used in a final product

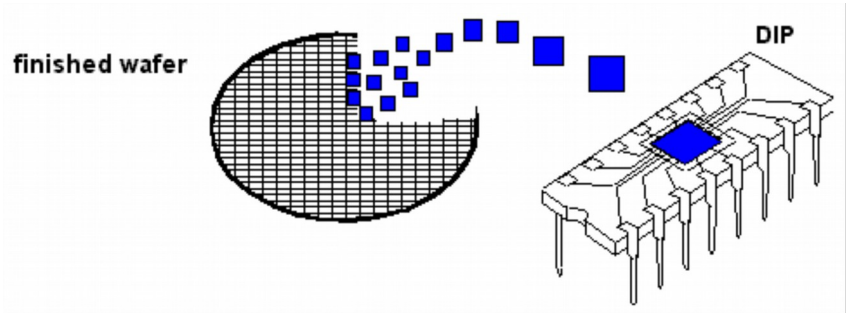


Figure 1.2 IC sliced from a finished wafer and used in a DIP switch [IC-1]

The process to create a single wafer of IC can be generalized in figure 1.3, following a reentrant plan, where sequences of steps are repeated as many times as needed, based on the task that needs to be done

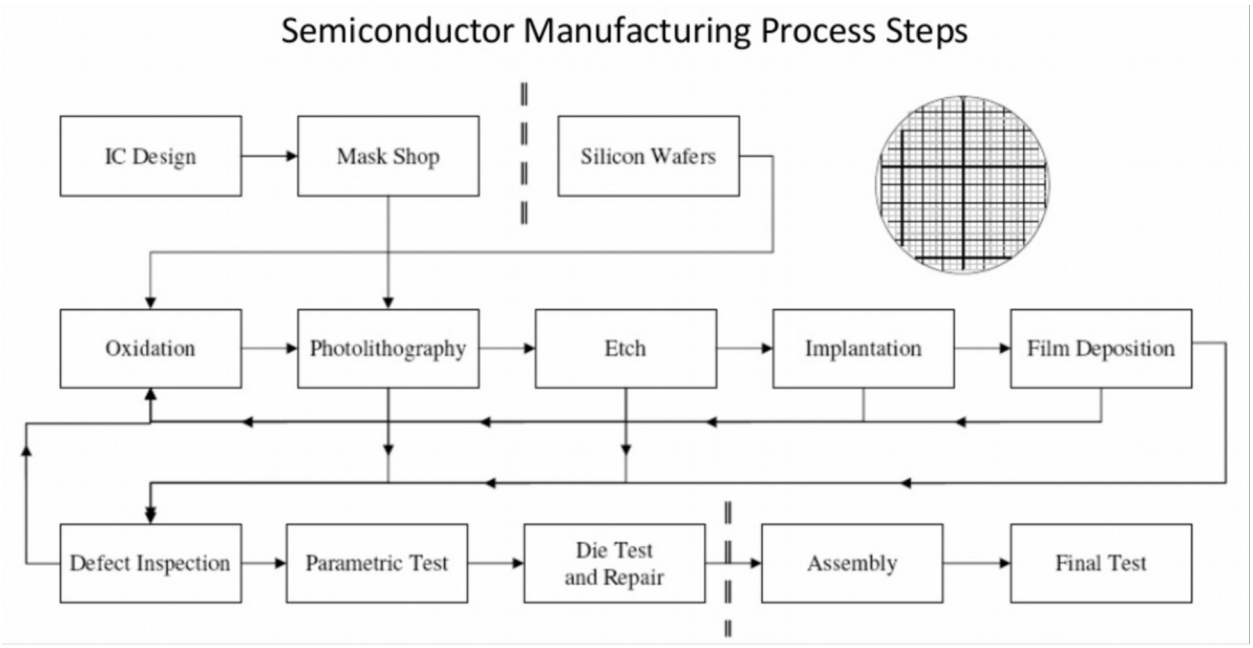


Figure 1.3.Steps to create a finished wafer. [A-2.3]

Based on the specifications of a client, an IC design is created which is used in the photolithography step which eventually leads to a finalized integrated circuit. In the next list, we will specify the steps in detail.

- **Oxidation:** Growth of thin crystalline overlayer called epitaxial film. There exist different ways to perform such a task, such as *thermal oxidation*, where a very thin layer of silicon dioxide is placed on top of the wafer.

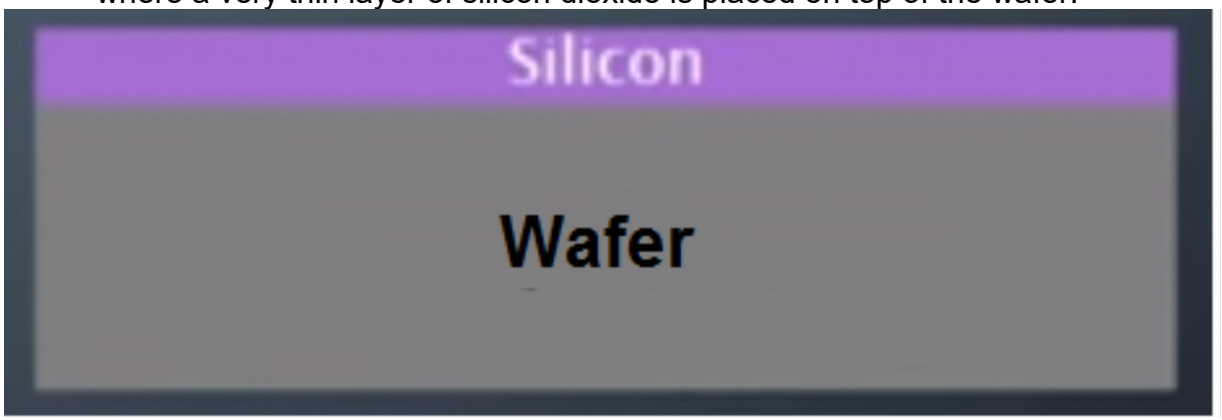


Figure 1.4. A silicon wafer with silicon dioxide layer grown on top of it

Photolithography: "Burn" an IC design on the photoresist surface. By placing a layer of "photoresist" which is a UV light sensitive material, on top of the silicon dioxide and using the client's specifications, a circuit pattern is burned on top of the wafer.

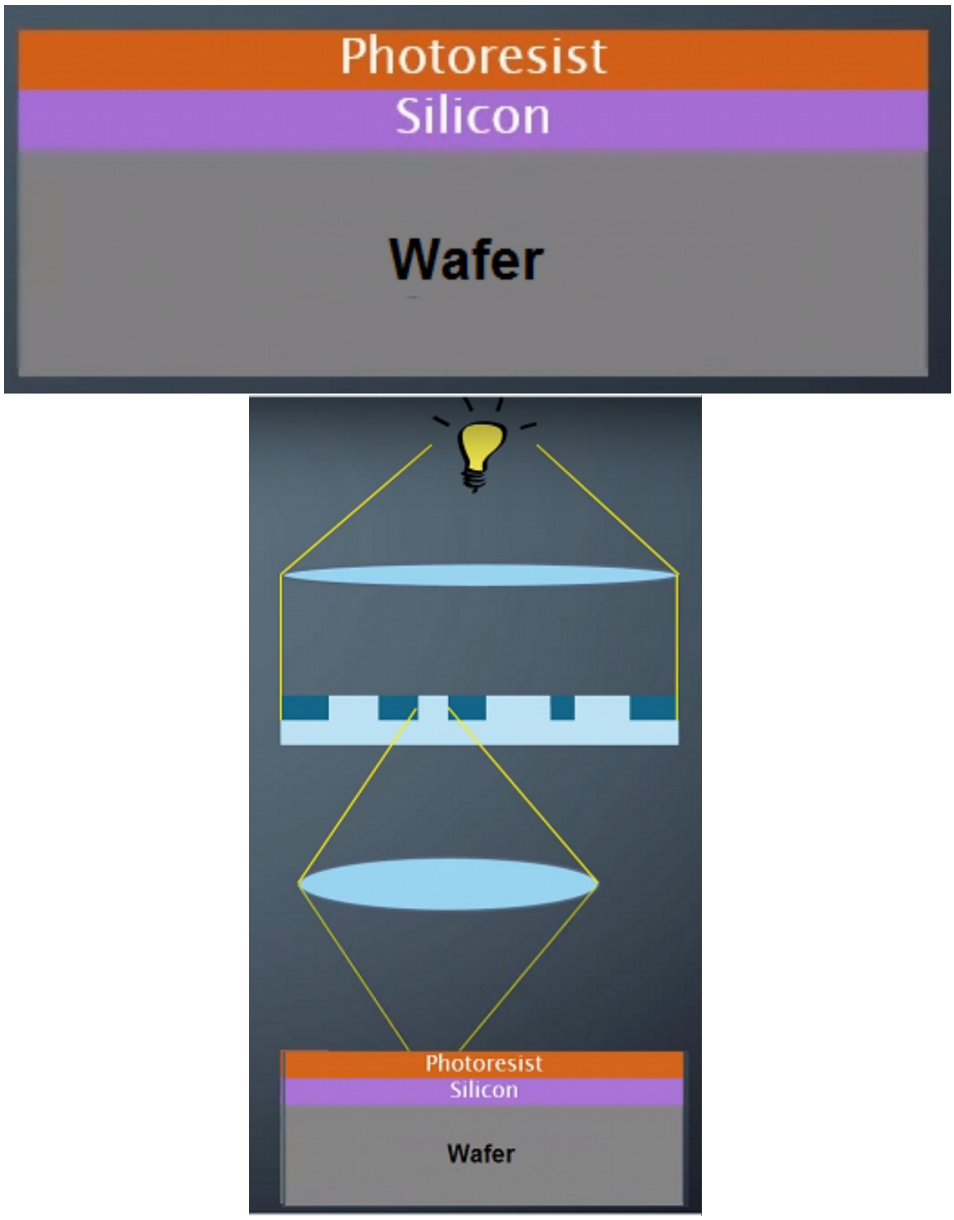


Figure 1.5. Photoresist is grown on the silicon dioxide layer, the photoresist is then "burned" with the IC mask pattern by applying intense UV light to it.

A mask that resembles the IC geometry is placed above the photoresit covered wafer. UV light of extreme intensity is directed downward to the mask and on to the wafer. This causes a chemical change on the photoresist which has been exposed to the UV light using the same pattern as the mask. The remaining photoresist is then washed away leaving a wafer looking like figure 1.6

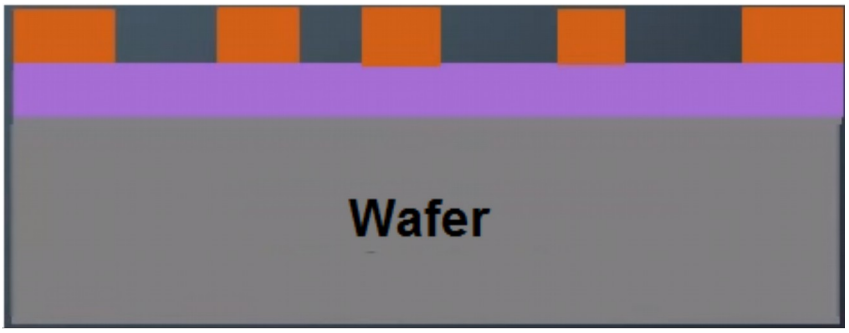


Figure 1.6. A wafer with a layer of dioxide and a photoresist, where the latter follows a pattern of an IC design.

Etch: Remove oxide following photoresist pattern. By applying a chemical agent, which does not affect the photoresist, the silicon dioxide is removed following the pattern of the photoresist. Figure 1.7 shows how this looks

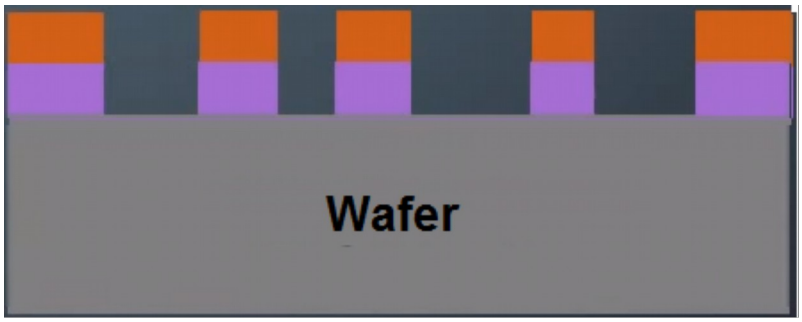


Figure 1.7. Wafer with silicon dioxide layer on it, together with photoresist layer.

Once the photoresist is no longer needed, and no errors have been made when removing the silicon dioxide, the photoresist is removed using a liquid resist stripper that modifies the resist so it no longer adheres to the dioxide. See figure 1.8

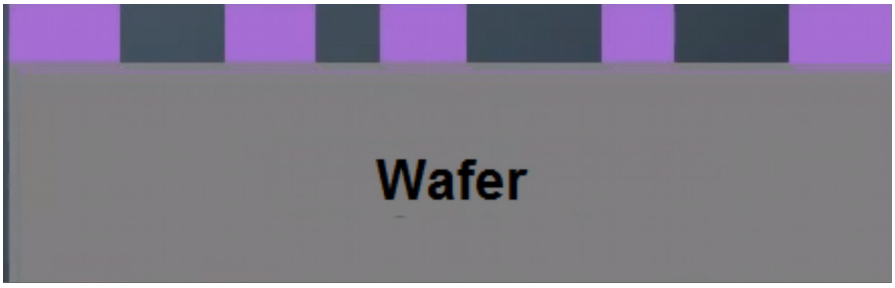


Figure 1.8. Wafer with silicon dioxide layer on it. No photoresist

Implantation: Modifying electrical properties of specific areas. This isdone in order to modify some areas of the wafer by using ion implantation. Which, based on the IC design, might require to have specific electrical properties such as a gate, where the electrical current can flow from one side to another. Figure 1.9 shows an example of this.

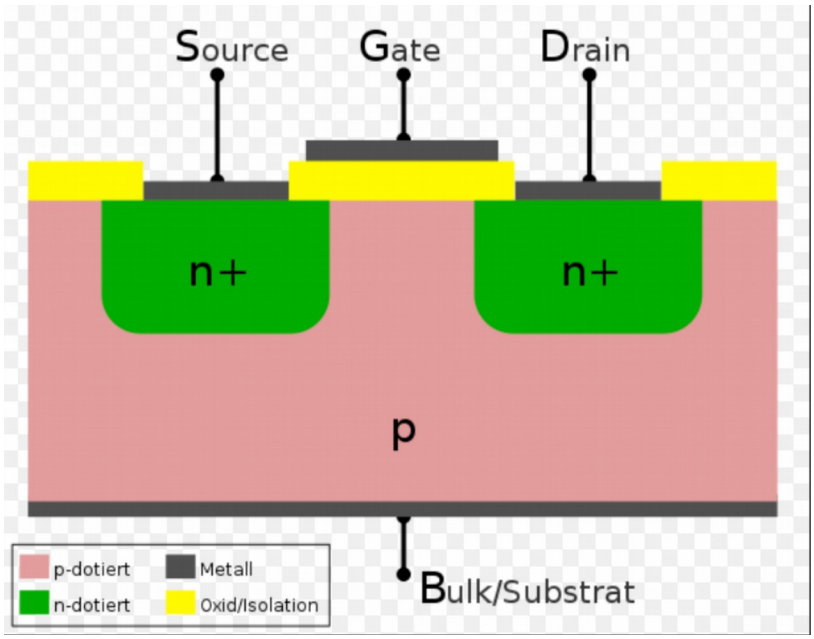


Figure 1.9. An NPN transistor, where current flows from the source to the drain, but only if the gate allows it. The gate allows for such a flow when an electrical current is applied to the gate.

Deposition: Material "growth". Oxide, dielectric (isolate layers). Depending on the recipe, a layer of dielectric (which isolates one layer from another) is placed on the wafer, or more oxide is grown on the wafer and the process starts all over again. In the end, only silicon with specific properties is left on the wafer. Figure 1.10 shows such a case.

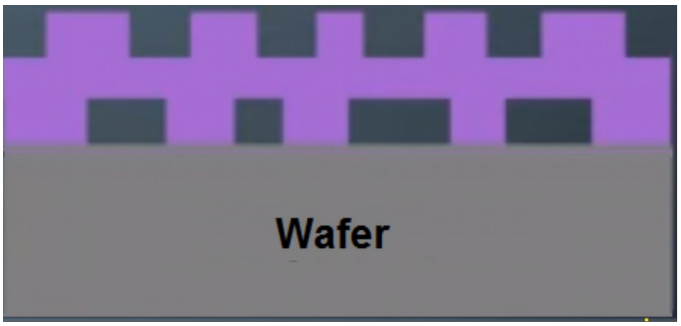


Figure 1.10. A wafer with several layers of silicon on top of it. Depending on the design, the silicon areas might have different electrical properties.

The reader should now have a general understanding of how a wafer goes from a simple silicon disc into a complex bed for integrated circuits, figure 1.11 shows a set of finished ICs stripped from a wafer.

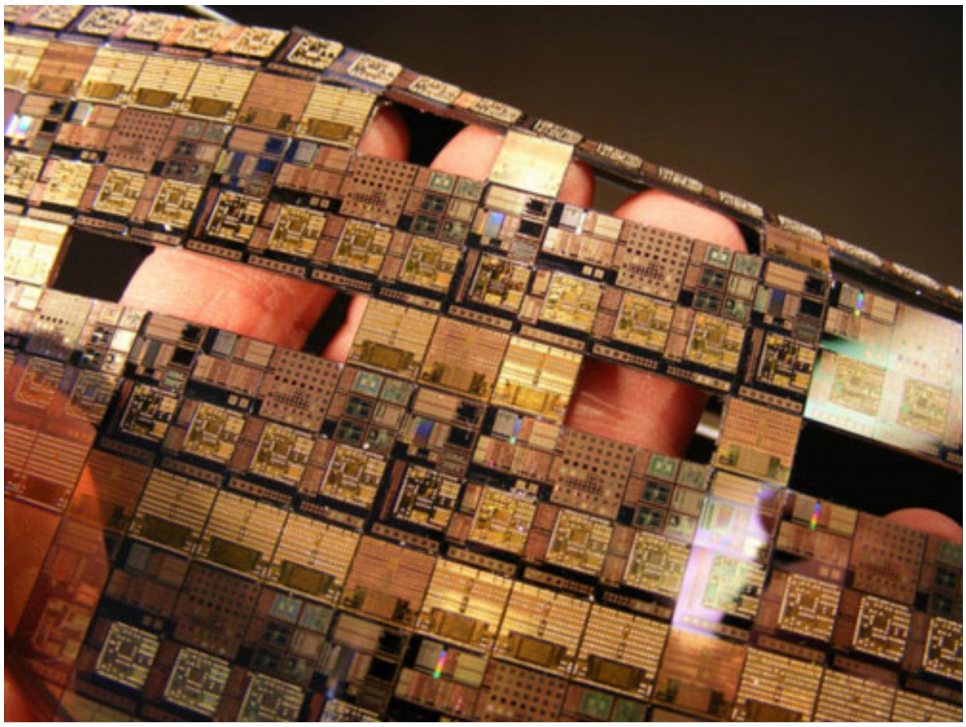


Figure 1.11. Strip of ICs.

Semiconductor manufacturing tools

In order for such a complex process to be successful, we are going to need machines, and lots of them. Given that this is a micrometric or even Nano metric scale [IBM-5nm] process which uses a reentrant path of operations, humans are out of the equation (but not completely), since the tediousness of the tasks would be likely to generate errors.

We turn our eyes towards automated machines which are capable of executing all of the operations the wafers need with minimal errors, and we keep the human operators ready to check for faults on the process.

However, before we move forward, we need to clear up some definitions which we will use later.

KPI (Key Performance Indicator)

A quantifiable measure used by a company to gauge its performance over time. An example for this could be coffee. Imagine a company that runs smoothly, however, one day the boss decides to ban coffee altogether from the company. It would be expected for the performance (and general mood) to go down, the boss would then allow for small amounts of coffee to be integrated into the company, this would increase performance over time (as well as mood). In semiconductor manufacturing (as well as in any other manufacturing process) however, KPIs are used to gauge the performance of the process of creating a finished wafer. There exist many different KPIs in semiconductor manufacturing, take the following figure 1.12 as an example.

Table 1: KPIs of Process Scheduling.

Measurement	Key Performance Index (KPI)
Time-based measures	Mean and variance of flow time per job or operation Mean waiting time Machine idle time
Work-in-process measures	Average number of jobs in queue Value of work-in-process
Due date-related measures	Mean tardiness Proportion of jobs tardy Conditional mean tardiness Number of jobs tardy Maximum lateness
Cost-based measures	Cost of idle machines Cost of carrying work-in-process inventory Total cost per job Average value added in queue

Figure 1.12. A table obtained from [A-34]which shows the KPIs in semiconductor manufacturing.

Let us look at figure 1.12 and the KPI definition above and ponder on them a bit. If a semiconductor company is trying to measure its performance based on time, they would ideally use the Time-based measures KPIs, trying to calculate how long machines are idle, how long they wait, or how each job or operation can alter the amount of time it takes to finish a job or an operation.

Let us define some important KPIs.

WIP (Work-in-process)

Inventory that has begun the manufacturing process, materials which no longer can be categorized as raw, but are not yet a finished product. In the case of semiconductor manufacturing, the WIP are all those wafers which are currently being processed by one of the tools in the factory plant.

Cycle time

Measures the speed or time it takes for manufacturing to produce a given product from the time the order is released into production line until the time it is a finished product. This time includes waiting times, idle times, processing times, etc. With an ideal cycle time to be the same as the raw process time.

Raw Process time

The time it takes for a product to go from start to finish without interruptions. Think of it as a single product being made in a factory, without waiting times or idle times, without tools being idle or down, the raw process time is the *perfect* processing time.
Using these definitions, we can move on to the tools, but first let's explain what tools and operations are in a semiconductor environment, with a simple example.

Imagine you have a hammer, this hammer is your tool. Without diverging too far into philosophical dimensions, the hammer can only perform 2 operations. Figure 1.13 for reference

Hammer down a nail
Claw a nail out

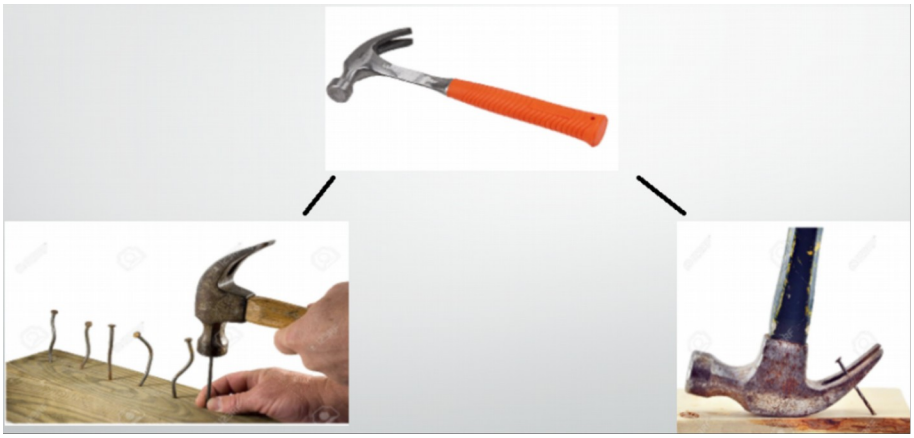


Figure 1.13. A hammer tool and its operations.

This same principle applies to the tools inside a semiconductor factory, where we have dozens of tools capable of executing many different operations. Often times, these operations can intersect between them.

By means of an example, the operations executed by Tool A and tool B could be.

Operations of Tool A = [OpA, OpB, *OpC*]
Operations of Tool B = [*OpC*, OpD, OpE]

This could mean that if a wafer needs *OpC* performed on it, and if Tool A is busy or down, then Tool B could take care of the wafer, and vice versa.

Knowing this, we need to understand the tool types inside a semiconductor factory.

The tools used in semiconductor manufacturing could be separated into 3 types

Tool Types

Single wafer tools: These tools take single wafers and perform operations on them. When working on a single wafer we can effectively reduce errors, thus increasing the quality of the product at the cost of WIP. For critical processes, a significant shift towards single wafer processing has become prevalent. These critical process steps include both etch and deposition processing where the process step is key to the performance of the device and to yield. [A-4] [A-4.1]

Batch tools: These tools take a couple of wafers at the same time and perform operations on them. Let's consider the case of thermal oxidation furnaces. The cost of running the same process temperature with the required process gas is **the same whether one wafer is processed or 25**, making the decision of batch process clear. The results from the deposition are consistent and of high quality and are not possible at single wafer processing level. [A-4] [A-4.1]

Cluster tools: Cluster tools are a combination of single and batch tools. Take figure 1.14 for reference.

When it comes to cluster tools, the idea is that a batch of wafers enters the tool at the Load Lock section, then a robot arm grabs single wafers and sends them to be processed individually on different chambers.

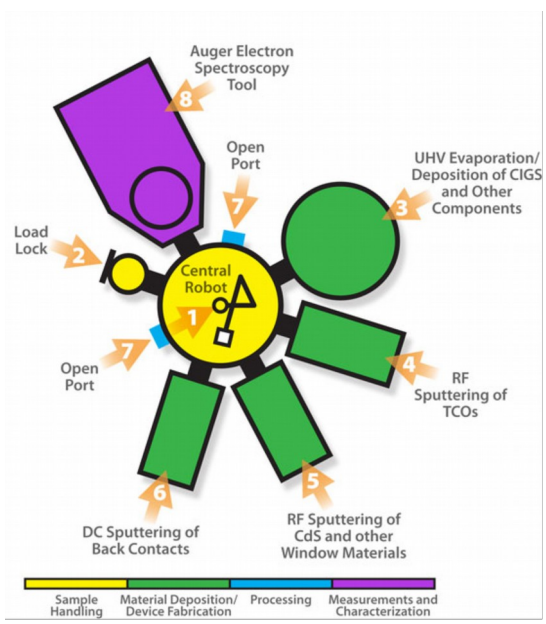


Figure 1.14. A cluster tool.

Recall the main steps for processing a wafer: oxidation, photolithography, etch, implantation, deposition. There exist many different tools which are capable of

executing all of these operations on the wafer lots. And as stated above, some tools might have operations which intersect between them.

It is clear that the process to create semiconductors is not a simple one. Errors can occur on the wafers when tools are performing operations on them. The tools themselves could go down for maintenance or simply shut down for other reasons. And with the global semiconductor industry's worth at over $300 billion USD (as of January 2016) [SI-W], manufacturers are always trying to improve their performance.

Factory Simulation

Factory simulation is the computer-based modeling of a real production system. Simulation allows organizations in the industry to analyze and experiment with their processes in a virtual setting, reducing the time and cost requirements associated with physical testing. Production, equipment, and personnel can be assessed within a simulation model, resulting in information that can preserve or improve value at the lowest possible cost [9].

The need for efficiency in today's factories has never been greater, with equipment and labor costs continuing to rise each year. Successful companies need to ensure that the costs associated with time, equipment and investments are being considered and optimized. At its core, factory simulation is an inexpensive, risk-free way to put your facility to the test, ensuring that you are meeting production goals and quality standards at the lowest possible cost. Simulation also provides a means to test and implement principles of lean manufacturing and Six Sigma [21]. And unlike spreadsheet-based analysis and forecasting, factory simulation software offers a quick and efficient means to adjust parameters and re-simulate, saving valuable time and hastening results [9].However, these simulations are expensive and very time consuming. But most important, they do not offer a **real-time forecast** simulation of the events.

Industry 4.0

The first industrial revolution, which focused on the use of steam powered engines, set the ball of industrialization in motion. It went for a good 70 years (1760 to 1820~1840). With the power of steam, societies began growing and increasing their standard of living. Iron production, textiles, chemical manufacturing, were some of the areas which were mainly affected by this first era.

The second industrial revolution (1870-1914), characterized by the industrialization of processes all over the world, but primarily in Britain, Germany and the US. Electricity, petroleum and steel processing amongst others, are often cited as the ones responsible for everyday household items such as washing machines, or the internal combustion engine, which led to the first automobiles and the first airplane on 1903 by the Wright brothers. Together with the basics of an assembly line, this led to mass production in different areas.

The third industrial revolution, often called the Digital Revolution, together with the mastery of the assembly line by Henry Ford, turned the tide and led to automation and great inventions such as computers. This stage of industry, which dates back to the 1980s is still in effect.

The fourth industrial revolution (Industry 4.0) will introduce a new era of automated production and data exchange in factories. Under some definitions, the 4.0 in its name refers to the fact that it represents the fourth major disruption in modern manufacturing, following the lean revolution, increased outsourcing, and the first wave of automation. An alternative definition takes a longer-term view of manufacturing history, claiming that the first three disruptions involved the use of steam power, the discovery of electricity, and the introduction of computers in the workplace. In either case, Industry 4.0 represents a major advance that will change how we manufacture goods [5].

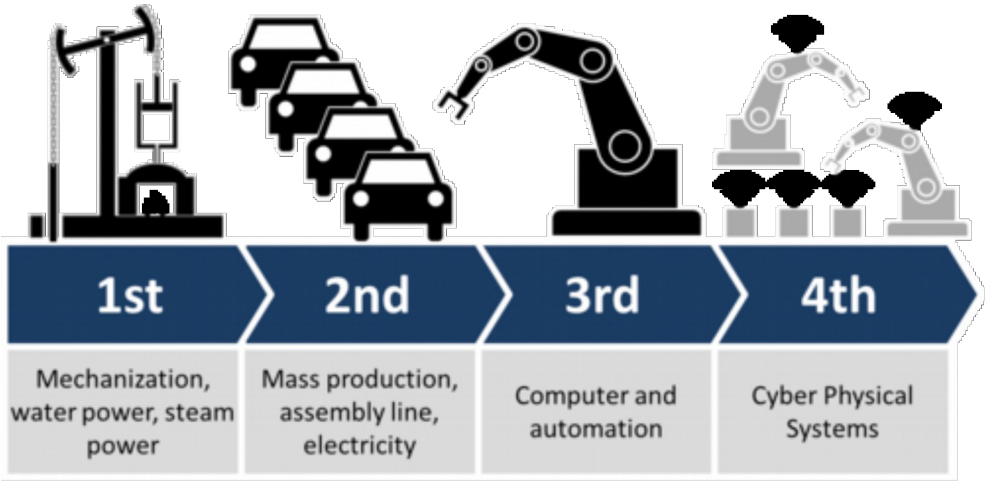


Figure 1.15The 4 eras of the industrial revolution. [Chris-Roser]

Industry 4.0 is a fairly recent innovation, and it wouldn't be possible without some recent technological advances: increased data volumes, computational power, and connectivity; the emergence of big data, analytics, and automated knowledge work; better human-machine interactions, such as those involving collaborative robots; and advanced production methods, including 3-D printing [5].

The following figure points towards the technologies responsible for the Industry 4.0

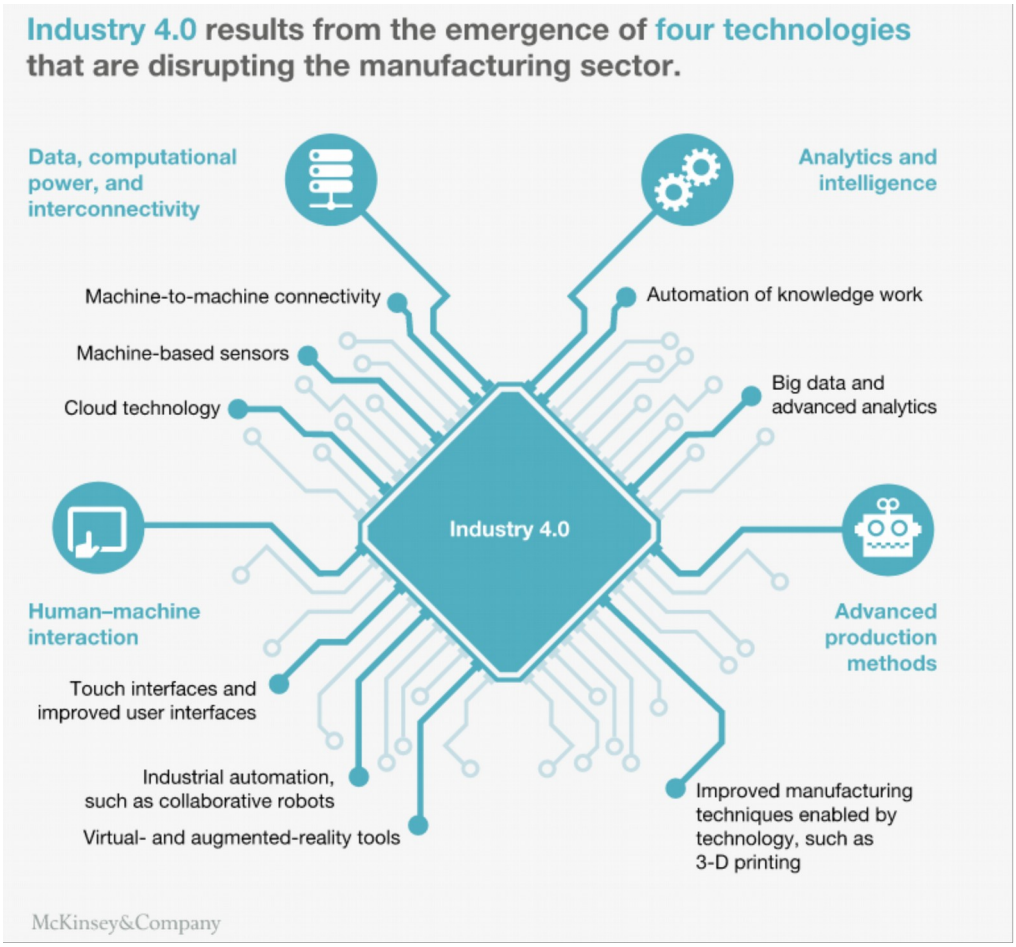


Figure 1.16: Technological *causals* that generated the industry 4.0 trend

Although Industry 4.0 techniques are relatively new to front-end factories and almost unknown at back-end factories, we've already seen some impressive results. The availability of low-cost automation options and advanced robotics is making it easier for chip manufacturers to reduce human error or accelerate production. Likewise, the use of sensor-enabled equipment and big data analytics lets semiconductor companies predict when plant equipment may need to be repaired or replaced ahead of any breakdowns. And better connectivity among products and machines gives managers more visibility into production steps, which helps them detect and address potential errors earlier in the process [5].

The industry 4.0 trend together with its *causals* will be the ship with which we travel towards a real-time solution for production forecasting in semiconductor manufacturing, as presented on this thesis.

Holistic Production

The Holistic Production Model (HIM) [1]is based on the idea that every single component in a production line is responsible for the overall production time. The HIM consists of 3 key data structures, **Atomics, Fundamentals and Fractals.** Figure 1.17 shows the template of the HIM.

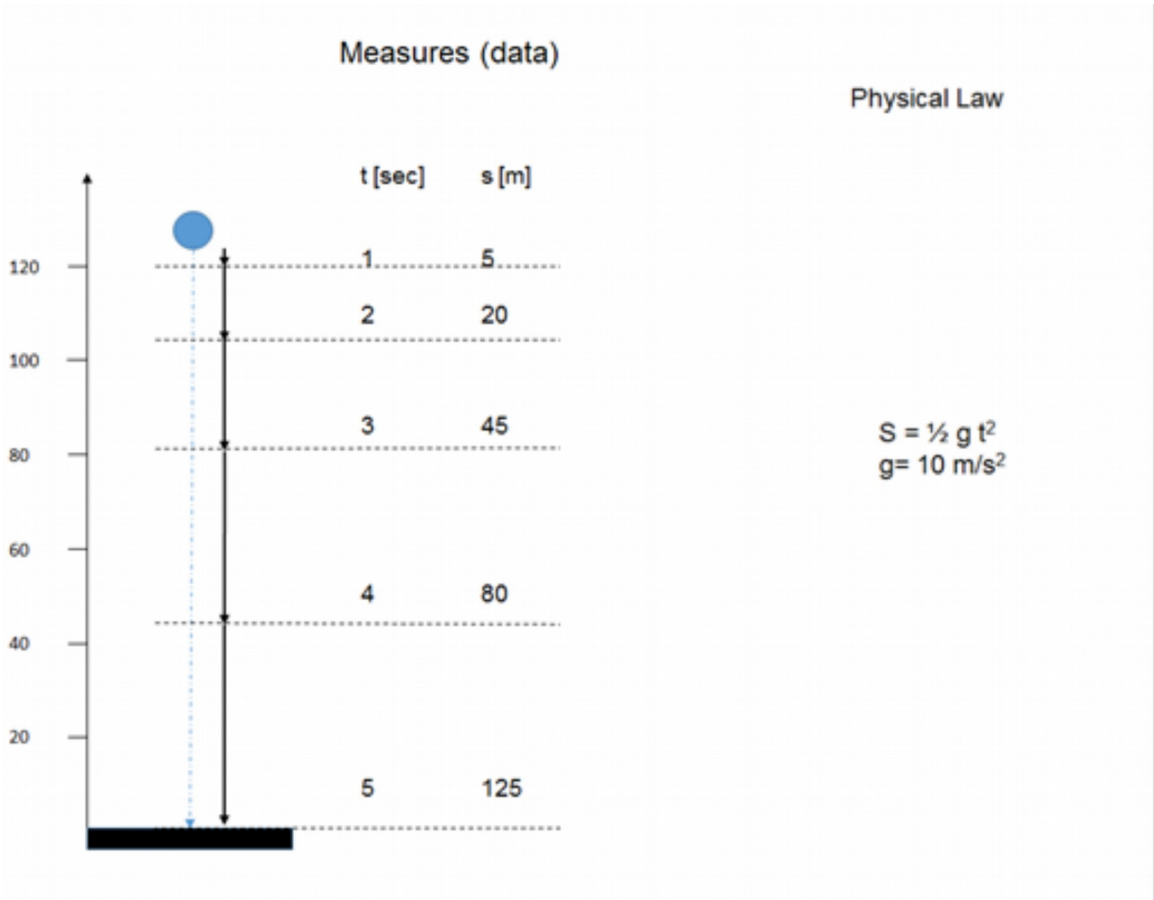


Figure 1.17. Template for the HIM

Let us explain what is happening in figure 1.17. We have a ball placed at a height of 125 meters. The ball takes 5 seconds to fall from the highest point all the way to the ground. With some tweeked parameters for gravity and an S for distance traveled.

As trivial and simple as the example may seem, this can be extended to a production line, where the ball is the initial “empty” wafer, then the ball travels through the air in a similar fashion as a wafer moves through the production line’s processes until finally reaching its final point at the ground, a finished wafer.

HIM Atomics

The atomics represent any event that is in any way considered within the domain model and hold information for the Fundamental and Fractal components.

HIM Fundamentals

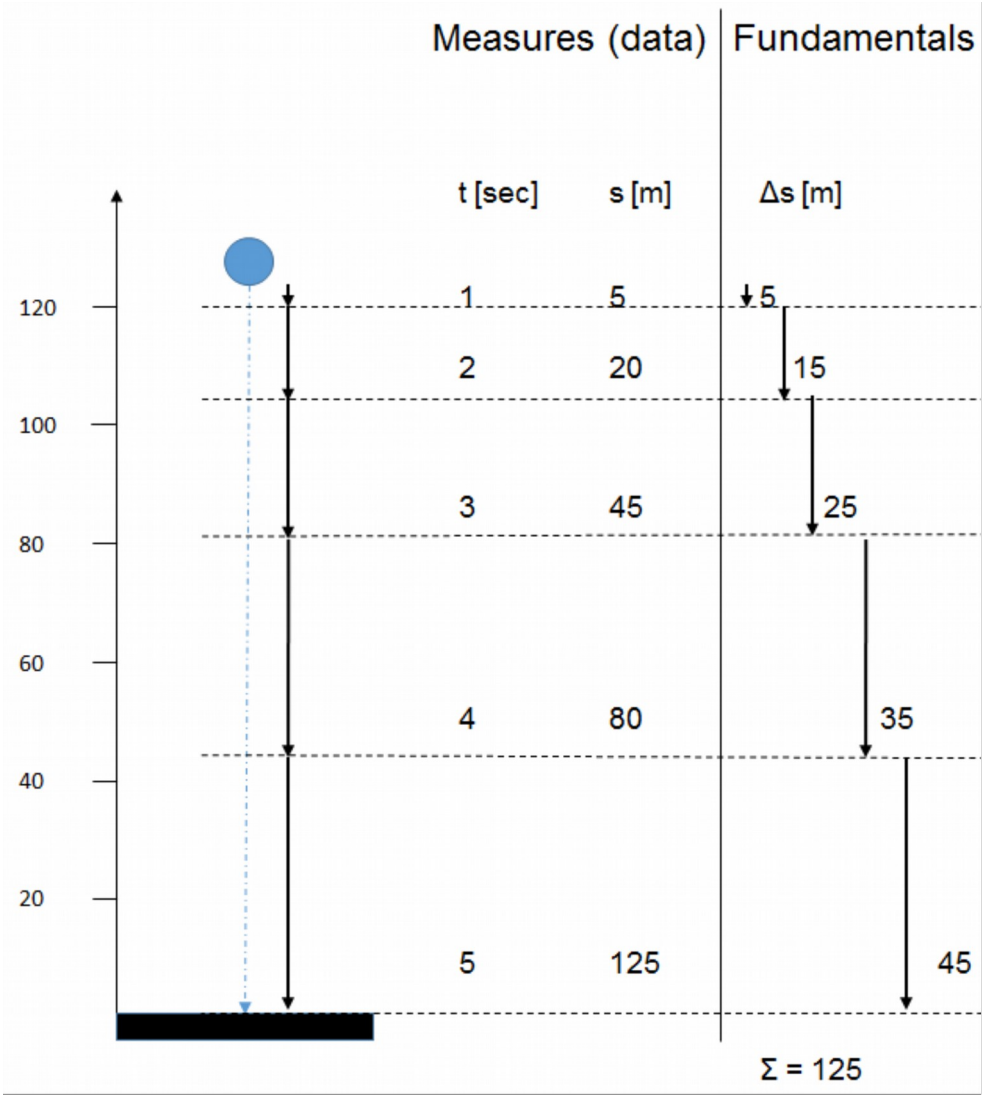


Figure 1.18. Fundamentals

Figure 1.18 shows the HIM Fundamentals, in the case of the ball, the fundamentals are the distance traveled by the ball at specific times. In the case of a wafer, the processing time of a wafer or the duration a lot was assigned to a tool.

HIM Fractals

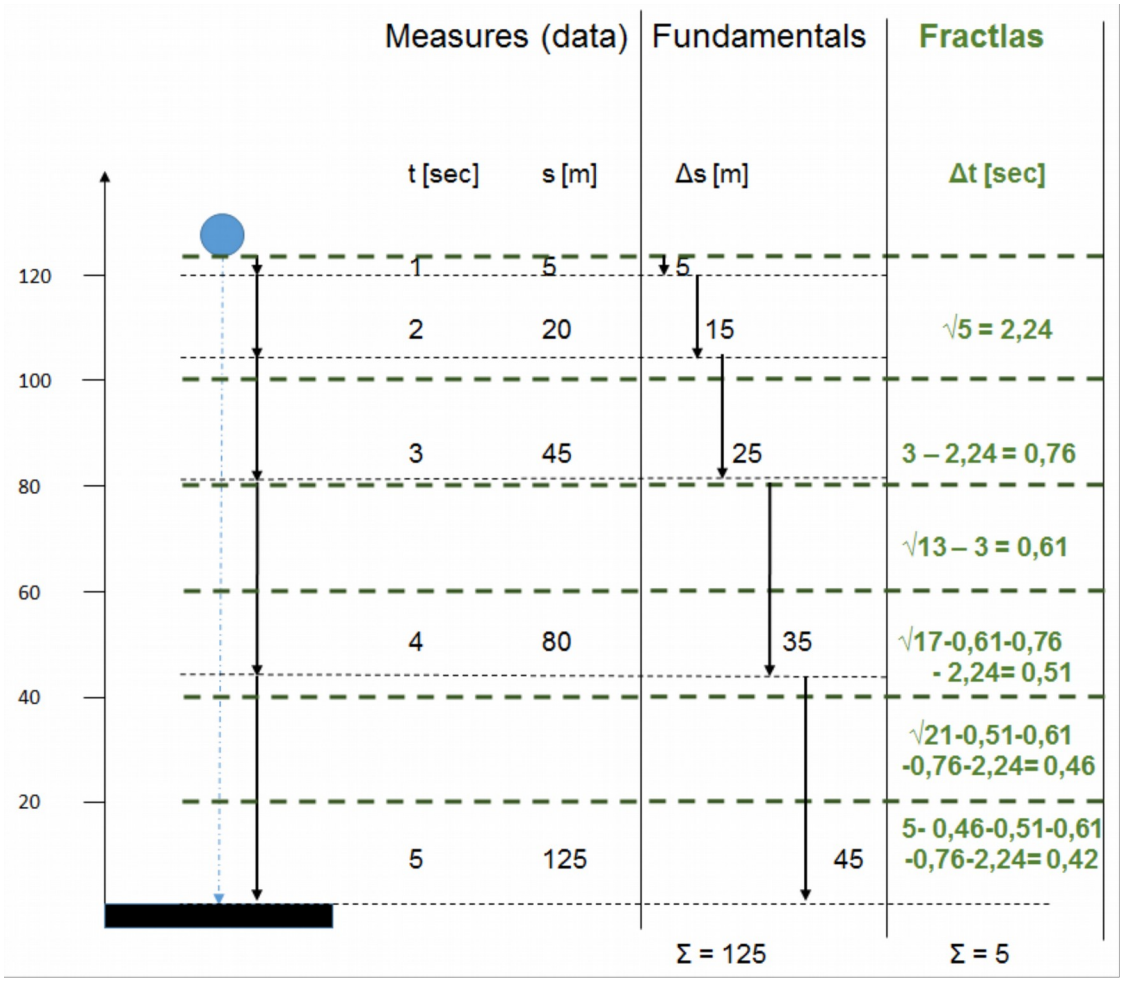


Figure 1.19. Fractals

Figure 1.19 shows the HIM Fractals, in the blue ball example, the fractals are used to represent the amount of time the ball stayed in a specific area of the fall. In the case of a wafer, the fractals can display the time a lot has used a tool or an operation, but not only that. The fractals are also capable of showing the WIP per tool or per operation. Recall that the WIP is the amount of work-in-process, so by using the HIM Fractals, we could obtain this valuable KPI, the WIP. The fractal is calculated by the S equation from figure 1.17 by clearing the equation in terms of *t*.

Hypothesis

By now, the reader must have a general idea on how a semiconductor factory works, how there exist tools which take wafers and perform operations on them, by batches or in single form. The reader should also have some basic idea on what simulation of a factory is. Let us make a connection from the Industry 4.0 and the HIM model.

Recall that the industry 4.0 trend is led by 4 techniques:

1. Data, computational power and interconnectivity

Analytics and intelligence
Advanced production methods
Human-machine interaction

The HIM model, as the name suggests uses a Holistic approach at its core. As defined by a dictionary, the word *holistic* means:

- Characterized by the belief that the parts of something are intimately interconnected and explicable only by reference to the whole.

Taking this into consideration, and the general idea from industry 4.0. A system has been designed which, by means of sensors and a highly optimized database, is capable of capturing every single movement within a semiconductor factory. From the moment a product lot is sent into the assembly line, all the way until it has been finished processing.

Using a **real-time information system** (RI) that takes all of this data and displays it in a neat and clean way, we can go back into HIM's 3 data structures, atomics, fundamentals and fractals.

Atomics

Lot Lookup

D06483

History

Transaction			Duration Hours			Operation		
Timestamp	Transaction Type	State	Total	Process	Waiting	Operation	Process ...	Job
22.02.2016 11:49:54	ProductionStart	InProduction	0,00000	0,00000	0,00000	6951	IMP1	16316465
	OperationStart	InOperation						
22.02.2016 11:53:42	EquipmentStart	InEquipment	0,06333		0,06333			
	ProcessStart	InProcess						
22.02.2016 11:53:45	ProcessEnd	InEquipment	0,06416	0,00083				
22.02.2016 11:53:51	EquipmentEnd	InOperation	0,06583		0,06500			
	OperationEnd	InProduction						
22.02.2016 11:54:00	OperationStart	InOperation	0,06833		0,06750	8112	BEL1	16316614
22.02.2016 12:38:00	EquipmentStart	InEquipment	0,80166		0,80083			
	ProcessStart	InProcess						
22.02.2016 12:43:35	ProcessEnd	InEquipment	0,89472					

Figure 1.20. Atomics as seen by RI

As stated above, the atomics show the events within the domain model. In figure 1.20 we see how, for a specific lot, we have all the transactions started on it. How the lot went into production, into operations, into equipment, etc. Recall that the atomics hold information for the fundamentals and fractals.

Fundamentals

Figure 1.21. Fundamentals as seen by RI

The fundamentals is the first aggregation tier, it collates the information of several events into information components. As seen on figure 1.21, for a specific tool we are capable of analyzing the lots that were being processed by said tool, as well as the time that the lots spent on that tool. The color code at the bottom and on top shows when a tool is on a productive, down or standby state, as well as when a lot is on equipment and when it is in process.

Fractals

	TimelineId	TransactionIdStart	Lot	DurationHours	Operation	UnitType
735	2016-03-14 21:00:00	7292428	D06483	0.32222	6401	Wafer
736	2016-03-14 22:00:00	7292428	D06483	1.00000	6401	Wafer
737	2016-03-14 23:00:00	7292428	D06483	0.15083	6401	Wafer
738	2016-03-14 23:00:00	7294463	D06483	0.84778	4111	Wafer
739	2016-03-15 00:00:00	7294463	D06483	1.00000	4111	Wafer
740	2016-03-15 01:00:00	7294463	D06483	1.00000	4111	Wafer
741	2016-03-15 02:00:00	7294463	D06483	1.00000	4111	Wafer
742	2016-03-15 03:00:00	7294463	D06483	1.00000	4111	Wafer
743	2016-03-15 04:00:00	7294463	D06483	1.00000	4111	Wafer
744	2016-03-15 05:00:00	7294463	D06483	1.00000	4111	Wafer

Figure 1.22. Fractals as seen in an MS-SQL database

Figure 1.22 shows the “inOperation“ fractals for one lot. For this instance, the lot in question has been assigned to specific operations with a specific *DurationHours*value which represents the amount of time the lot spent in that operation. This is only half of it though. Because as expined above, the fractals are also capable of returning the WIP.

	TimelineId	Operation	OperationWIP
1	2016-03-15 00:00:00	4111	4.39361
2	2016-03-15 00:00:00	5111	12.85833
3	2016-03-15 00:00:00	6401	12.79416
4	2016-03-15 00:00:00	6551	10.61834
5	2016-03-15 00:00:00	7724	3.51028
6	2016-03-15 01:00:00	4111	5.00000
7	2016-03-15 01:00:00	5111	15.00333
8	2016-03-15 01:00:00	6401	12.66860
9	2016-03-15 01:00:00	6551	4.82222
10	2016-03-15 01:00:00	7724	2.09027
11	2016-03-15 02:00:00	4111	4.82250
12	2016-03-15 02:00:00	5111	13.61751

Figure 1.23. WIP obtained from Fractals.

Figure 1.23 shows the WIP by an operation ordered by time. By modifying the SQL query a little bit, we aggregate the units which were processed by either a tool, or an operation. We can read this information as: “Row 6, at 1 AM, operation 4111 processed 5 lots”.

This aggregation comes from the entire work done by an operation or a tool. In the case of an operation, the HIM takes all events into consideration. This would mean that all through 1 AM, operation 4111 processed 5 lots.

With this knowledge in our hands, we can ask ourselves some questions, such as:

How could we use a model such as the Holistic Information Model to perform production forecasting?
What is production forecasting?
How can we create a system that is capable of predicting the WIP on different tools and/or operations? Is it reliable and accurate or even possible?

These questions will be answered in chapter 2. But for now, let's recap what we have learned.

We have seen (in general terms) how a semiconductor factory works to make their products. We also saw what a key performance indicator is. We have seen that the work-in-process is a KPI which represents the amount of units being processed by a tool or an operation, amongst other definitions, we have seen the types of tools that exist within a semiconductor factory and how they may differ and offer improvements over other tool types. We also saw what the industry 4.0 is all about, together with the Holistic Information Model, which tries to gather all data from an environment in order to be able to understand said environment more.

The chapters ahead include statistical definitions. No real statistical background is required since the definitions will be explained as simple as possible with the help of examples, as needed.

The next chapters will also include code used by some statistical analysis tools. The code will be available at the end of the chapter the code is used, unless it is code generated by the author, in which case the reader needs to only ask for it. Naturally sources to definitions, topics, code obtained from the web and anything of the sort will be available at the end of this thesis.

Production forecasting – and forecasting in general

"Forecasting is the art of saying what will happen, and then explaining why it didn't!"

--Anonymous (communicated by BalajiRajagopalan)

The converse is also true: saying what WON'T happen, and then explaining why it DID! See the large El Niño of 1997 for a recent example.

Recall the 3 questions planted by the hypothesis at the end of chapter 1
How could we use a model such as the Holistic Information Model to perform production forecasting?
What is production forecasting?
How can we create a system that is capable of predicting the WIP on different tools and/or operations? Is it reliable or accurate or even possible?

These will be, from now on, our **3 hypothesis questions.** We will answer these questions (without any specific order) as we move through this chapter.
As mentioned before, we will often use code and an analysis tool to explain some definitions as well as show examples, but mostly to try and answer our **3 questions.**

The program we will use is **RStudio**[RSTUDIO], this is a free to use tool, although paid versions exist.

But first, let's talk about forecasting.

Forecasting

Forecasting is one of the most used techniques today by any type of service out there, going from simple fast food ordering [17] [18] or weather forecast, all the way to finding the point in time a machine will break down based on its historical behavior [MBe-S], and everything in between. Forecasting on itself, is nothing but a technique to make a prediction on a specific outcome or result based on an analysis of previously known data as well as on external influences.

While the term forecast can vary depending on the context upon which it is used, the general concession remains that forecast is nothing more than a way to make future predictions based on past knowledge.

However, we must first get acclimated with some statistical terminology, this will be done in order for the reader to get a good grip on the data that will be presented here, as well as the examples and the answer to our **3 hypothesis questions.**

Time series

A time series [Ts-1] [Ts-2] [Ts-3]is a series of data points measured in *discrete time*. Such data may be collected at regular time intervals, such as hourly, monthly, annually. Time series are used in statistics, econometrics, mathematical finance, weather forecasting and many other different applications. Figure 2.1 contains a time series.

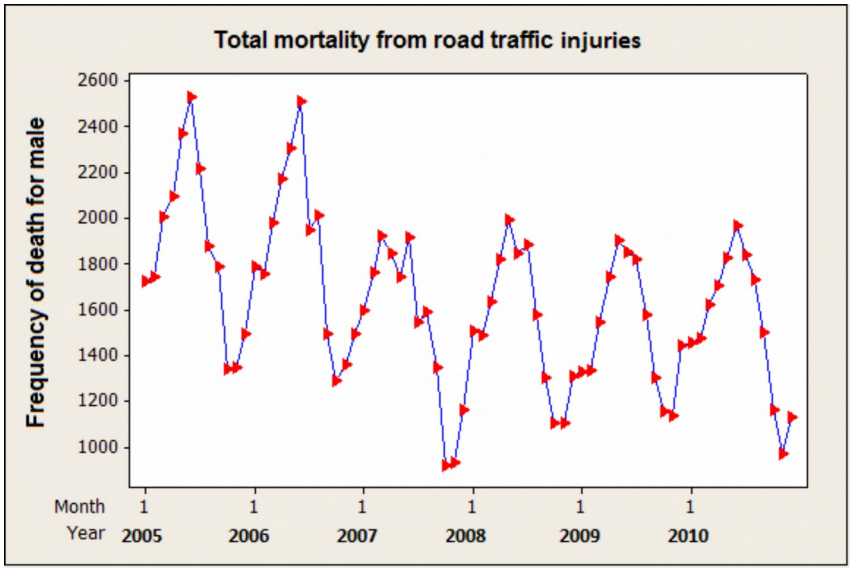


Figure 2.1. Time series for Fatal Traffic Injuries among Iranian Drivers; 2004–2010.

Here we have a time series displaying the amount of deaths per month due to traffic injuries. The data is measured each month through 6 years. It is also very clear that a seasonal pattern appears within the data, where a likelihood of dying due to a car accident injury increases during the middle of the year. Also a trend that implies that deaths are slowly decreasing, which is a good thing.
Depending on the frequency of the data (hourly, quarterly, annually) different patterns emerge in the data set which forms the component to be modeled.

The difference between discrete time and continuous time

Figure 2.2 for reference. A discrete signal is a time series consisting of data points in sequence. Unlike a continuous signal, a discrete signal is measured at same time intervals, as seen on figure 1.22, the discrete signal has a time interval of 1, meanwhile, a continuous signal can measure data points at any time between a specific interval. If one is trying to obtain information about a specific data point in a discrete signal, and this data point is found between measurements, interpolation will have to be applied, while a continuous signal is capable of capturing information at any point in time between its time domain. It is important to note that a continuous time series can be transformed into a discrete time series.

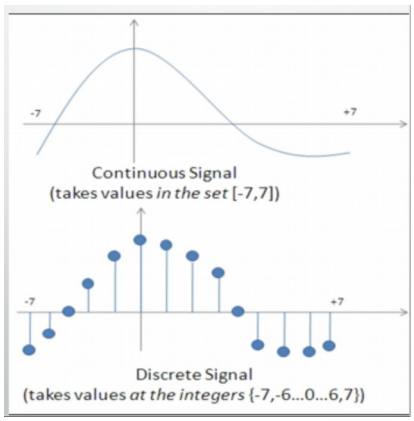


Figure 2.2. A continuous and discrete signal with the same measuring boundries.

Seasonality

Seasonality [A-5][A-6][A-7], as seen on figure 2.1, we have a seasonal component to our time series, one could speculate that fatalities peak between the months of March and July every year. This is known as a seasonality pattern. Let's look at another seasonality example on figure 2.3.

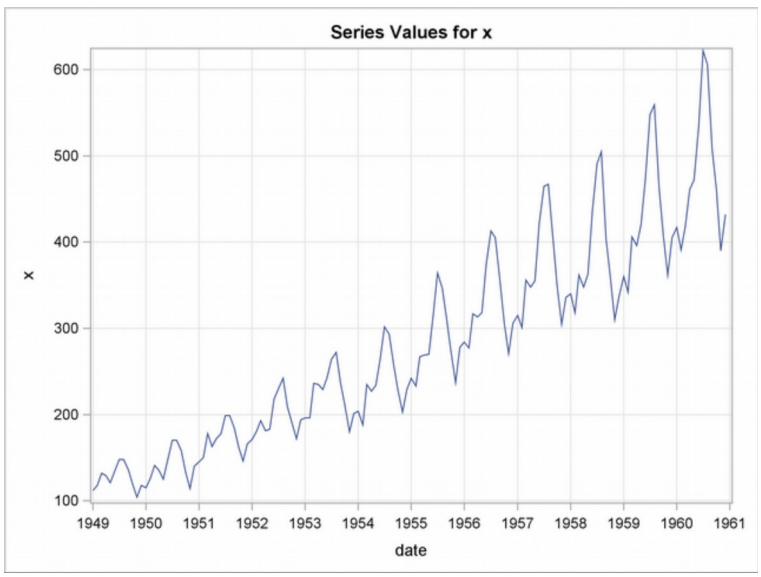


Figure 2.3. A time series showing the amount of passengers for an airline. Monthly Airline Passenger Numbers 1949-1960 by Box & Jenkins.

In figure 2.3 we have a discrete time series example of the amount of monthly passengers an airline tends to for about 11 years. This data also shows a seasonality pattern where the amount of passengers seems to peak from June to August, this could be explained because of the holiday season, and people tend to travel more during these months. The data also shows an increasing trend.

Trend

A trend [A-5][A-6][A-7] is simply a pattern which is followed by (in this case) a time series. In figure 2.4 we can see how the airline passengers trend (red line) is slowly raising, we could expect the next year to have a pattern that follows the same seasonality, but with an increased amount of passengers, as shown by the trend.

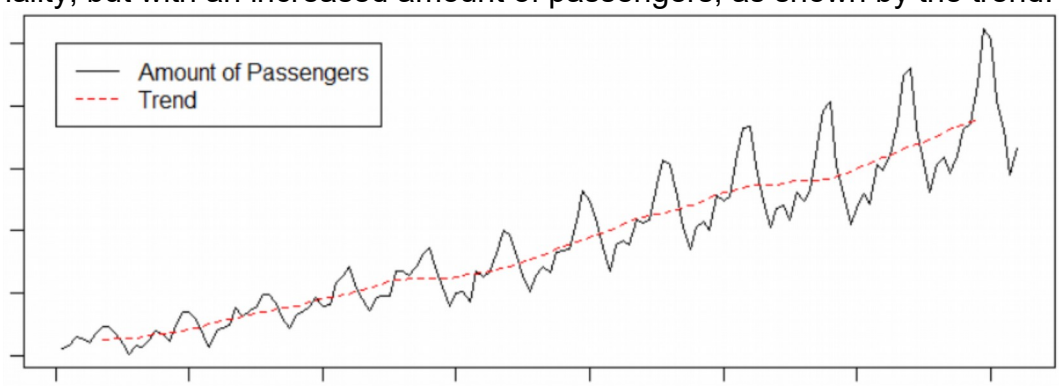


Figure 2.4. Trend of airline passengers.

Stationary time series

A stationary time series [A-Sta][A-Sta-2]is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. Let's break all of these down.
Depending on the context, the definition of ***mean***could be different, in statistics however, we will use the mean (or average) of a set of numerical observations, from here we can break it down into two categories. The mean of a sample and the mean of a population. In our case we will use the population mean.

- for a data sample of size *n,* its **sample mean** is defined as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Figure 2.5. Sample mean.

for a data population of size *N,* the **population mean** is:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Figure 2.6. Population mean.

Where *xi* is the value of an observation at time *I.*

The ***variance*** is a measure of how far an observation is from the mean. The variance is calculated with the formula from figure 2.7:

$$\mathrm{Var}(X) = \mathrm{E}\left[(X-\mu)^2\right].$$

Figure 2.7. The variance is obtained by subtracting the value **X** from the mean **μ** and squaring the result.

The variance leads us to another very useful statistical definition, the *standard deviation*, which will be defined further ahead in this chapter.
Going back into a stationary time series, the ***autocorrelation*** is a measure of similarity between a time series and a lagged version of itself. The main use of autocorrelation is to find repeating patterns such as a periodic signal, hidden between the noise of the time series. Figure 2.8 shows an example of this.

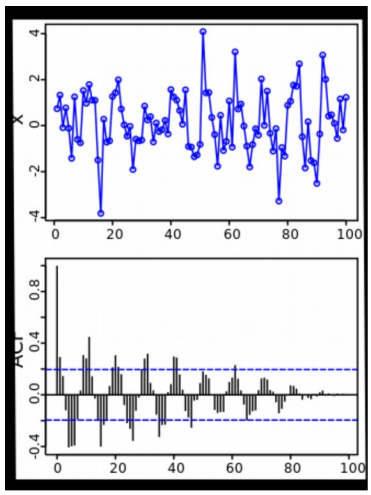


Figure 2.8. Above: a time series plot of 100 random observations concealing a sine function. Below: the sine function revealed in a correlogram produced by autocorrelation.

The bottom plot of figure 2.8 is obtained through by using the *ACF* function in Rstudio, we will not go too deep into Rstudio yet, but let us explain how the function works.

Imagine you have a time series for stocks called ***St*** with values from stocks, measured at discrete times, with ***St = {100, 110, 112, 115, …}***, where *t* represents a point in time *t***.** The prices change over time. Let's now define a time series at a time point *t-1* ***St-1 = {110, 112, 115, …}.*** In order to calculate the autocorrelation we need to first define the **Autocovariance**.

The autocovariance is the value obtained by multiplying the time series variance at a point *t* by a lagged version of the variance of itself at time *t-1.* The covariance is obtained by using the formula from figure 2.9

$$\mathrm{cov}(X,Y) = \mathrm{E}\left[(X-\mathrm{E}[X])(Y-\mathrm{E}[Y])\right]$$

Figure 2.9. Covariance function.

Where ***E[X]*** and ***E[Y]*** is the mean of ***X*** and ***Y*** respectively, ***Y*** is the lagged series at *t-1*, and ***X*** is the series at time t.

The autocorrelation function is a substitute for quantifying the relationship between a time series with its past data. The ACF is the division of the autocovariance divided by the variance. Recall from figure 2.8 we obtain the autocovariance of the series **S** at *t* with the series **S** at *t-1,* thus in order to obtain the autocorrelation value, we need to divide this autocovariance by the variance of the time series. Figure 2.9.1 shows the autocorrelation formula.

$$r_k = \frac{\sum_{i=1}^{k} (x_k - \mu_k)(x_{k-i} - \mu_{k-i})}{\mu}$$

Figure 2.9.1. Autocorrelation formula

Where *k* is the lag, μ_k is the mean at time *k,* and $x_k$is an observation at time *k.* The idea is to sum the multiplication of the differences between observation and mean divided by the mean.

Due to this division, it is very common to find the first couple of values to be rather high. This is due to the first autocorrelations being made between practically the same time series, but as the lag increases and the ACF continues to measure the similarity between the lagged series, the value will decrease. The value returned by the *ACF*is between -1 and 1. This leads us to another basic but very important definition in statistics, the **correlation**, but we will not be defining this yet, the important thing that the reader needs to know is that the correlation is a measure of similarity between two variables, while the autocorrelation is the measure of similarity found within one single variable, in this case a time series.

Recall from figure 2.8 the ACF plot, let us analyze it here on figure 2.10.

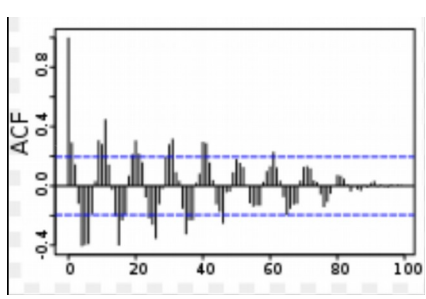


Figure 2.10. The autocorrelation function plot.

Here we see the correlation measure ranging from -1 to 1, where a value of 1 implies total positive correlation between the series, a value of 0 implies no linear correlation

and a value of -1 implies total negative correlation. We will explain this further once we reach the definition of **correlation.**

The idea behind a stationary time series is that its statistical properties such as mean, variance and autocorrelation are all constant over time, this means that a stationary time series is not dependent on any other variable other than itself. This is why it can be possible to forecast a stationary time series with itself alone, this will be of great help when trying to answer our **3 hypothesis questions.**

Let us, for the sake of comparison, analyze a non-stationary time series, namely the air passenger's time series from figures 2.3 and 2.4.

Recall that this time series was a discrete series with time intervals of 1 month. Let's use Rstudio to obtain the acf plot of the air passenger's data set with figure 2.11

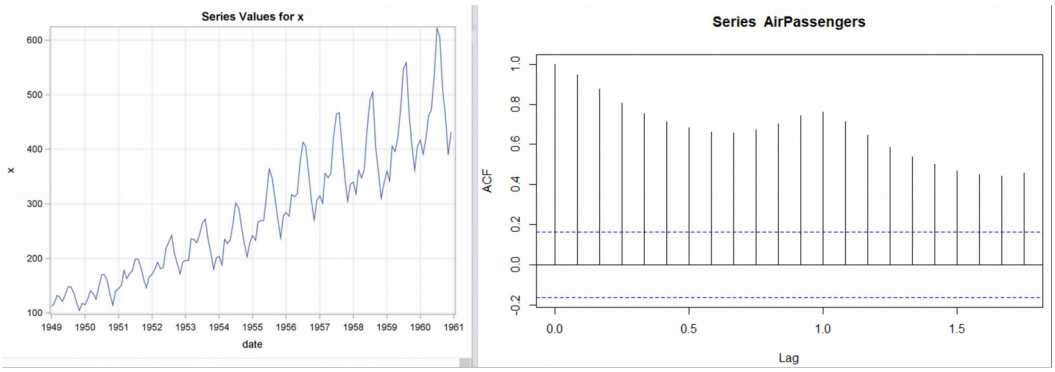


Figure 2.11. Left: Air passenger's data set. Right: ACF plot of the air passenger's data set.

There is something of note here, which we have already explained before, the amount of passengers is dependent on the time, with an increase in passengers around the holiday season (months between June and August). The ACF plot returns high autocorrelation found within the air passengers data, this is because this data set is not stationary.

So, what can we conclude from this?
Mainly that a stationary time series is one that can be forecasted with itself, a non-stationary time series is one that is dependent on other variables, in the case of the air passenger's data, time.

This entire process is very tedious and sometimes slow if the data is not in the proper state. However, we can come up with a very simple way of finding out whether or not a series is stationary or not. This can be done by simply looking for seasonality in the data. If we look closely to the air passenger's data and the data of the top plot from figure 2.8, a very clear seasonality pattern is found in the air passenger's data while the time series from 2.8 is not very seasonal.

What we want to look for in an ACF plot is the decrease in correlation, as seen on figure 2.10.

Let's recall two very important definitions which we saw above, **correlation** and **standard deviation.**

Correlation

The correlation [A-Stephen][A-10] [C-1] is a measure of similarity between two variables, or in our case, time series. Just like the autocorrelation, the correlation ranges from -1 to 1. There exist many different ways to calculate the correlation, there are some methods which return a value of 0 when two variables are positively correlated and slowly increase as correlation between them disappears, whereas the ACF or correlation return a value of 1 when two variables are positively correlated. Let us look at some examples for correlation. Picture you have 5 different vectors with the data from figure 2.12.

	Vector.A	Vector.B	Vector.C	Vector.D	Vector.E
1	0.009953966	-0.01586103	-0.01622703	0.01677471	-0.02134260
2	-0.230181145	0.20330379	-0.68532184	0.14705071	-0.12226900
3	-0.552905273	-0.03403964	-0.51135631	-0.33052491	-0.23908857
4	-0.089739322	-0.08276864	-0.41120913	-0.30101166	1.56018599
5	-1.986059137	-0.25221762	-0.36904453	-0.58561940	0.54590376
6	-1.635875161	2.74131045	-0.05841131	-1.45882583	0.07848098
7	0.525846706	-1.13464366	-0.06701484	-1.43199022	-0.55705712
8	-0.913511821	0.68837478	0.37641204	-0.86174643	2.06550717
9	-1.538179621	0.81433038	1.63993904	-1.41478931	1.80273829
10	0.817957993	-0.42656051	2.77338024	-0.12329182	1.31688375

Figure 2.12. 5 numerical vectors which we will use for correlation.

One of the best tips that the reader can take, is to plot the data, you will always find interesting patterns between your data plots which could hint towards specific results, this might not always be the case but it is good practice. Figure 2.13 shows the vectors above in a plot.

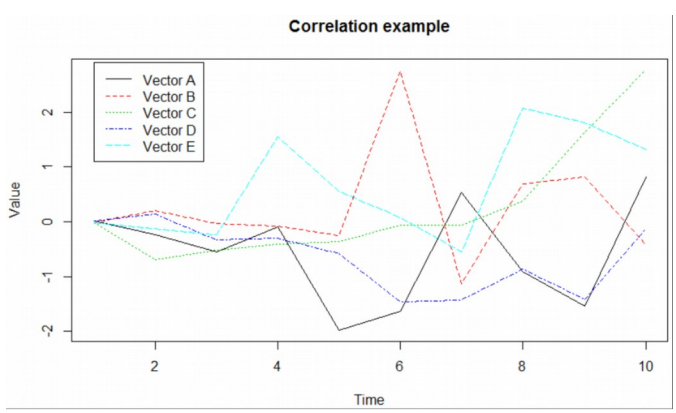


Figure 2.13. Plot of the 5 vectors from figure 2.11.

Remember that we are trying to find a correlation (similarities) between the vectors. At first glance, there does seem to be some similarity between Vector C and Vector E. They both seem to have an upwards trend despite not having the exact same values.
By using RStudio we can create a neat graphic which contains a correlation map. Figure 2.14 will make it easier to understand the correlation map.

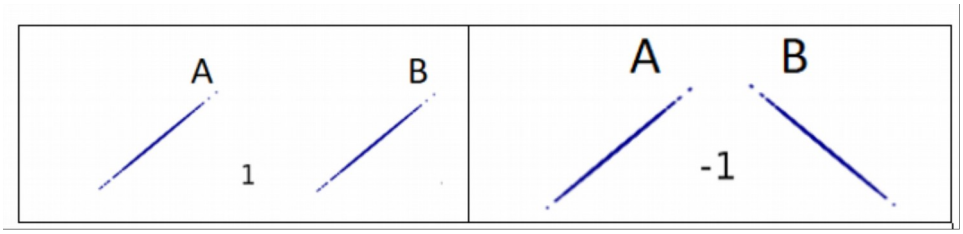


Figure 2.14. Left: total positive correlation. Right: total negative correlation.

By looking at the left side of figure 2.14, we see a correlation value of 1. This means that when we compare two vectors (A and B), if they are exactly the same, we will get a value of 1. On the right side of figure 2.13 we have the opposite case, where the vectors are the same but mirrored or inverse. On the left case (correlation of 1) we could simply assume that, for each increase in vector A, there exists a similar increase in vector B. On the other hand (correlation of -1) we could assume that, for each increase in vector A, there exists a similar decrease in vector B. Let's now get the correlation map of our 5 vectors from the example above to make this clearer.

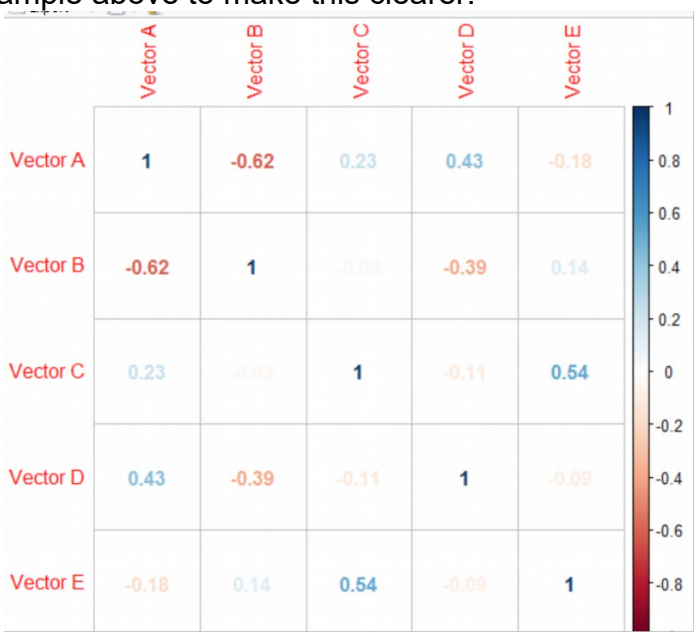


Figure 2.15. Correlation map.

Figure 2.15 is our correlation map, where we have plotted the value of the correlation following a similar example as the one seen on figure 2.14.

How can we read this correlation map? Notice that when we try to find the correlation of vector A with itself, we end up with perfect correlation, value of 1. We can look on and find that vector A and B have a negative correlation, we can understand this as, each

time vector A increases, vector B decreases. Another important thing is our idea from the original plot of the vectors on figure 2.12, recall that we said that there might have existed a correlation between vector C and vector E, since they were following the same upward trend. We see that indeed, there is a positive correlation between them. Notice also those barely visible correlation numbers between vector C and vector B. This is closely to 0, meaning no correlation at all.

There exist other methods of plotting the correlation map, a personal favorite is the one which uses ellipses (figure 2.16).

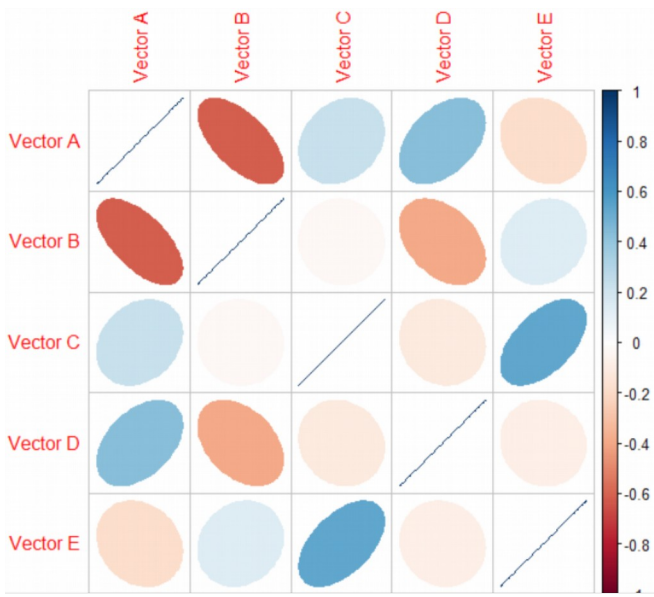


Figure 2.16. Correlation map using ellipses. Where, as difference between vectors appears, the blue diagonal line (correlation of 1) starts getting thicker until it becomes a light white colored circle (correlation of 0), then it flips to the left until it becomes a red diagonal line towards the left side (correlation of -1)

We have now learned what the correlation measure is. This will be extremely helpful when answering our **3 hypothesis questions.**

Standard deviation

The standard deviation, denoted in most literature with the symbol σ (sigma), is a measure of data dispersion in a data set. When analyzing a data set, but mainly when making forecasts, the smaller the value of the standard deviation, the better. We will see an example ahead, but first, let's look at the two different ways to calculate the standard deviation, the sample standard deviation and the population standard deviation.

Samplestandarddeviation

$$\sigma = \sqrt{\frac{\sum (X-\mu)^2}{N-1}}$$

Populationstandarddeviation

$$\sigma = \sqrt{\frac{\sum (X-\mu)^2}{N}}$$

The square sum of the differences between observed values and mean. Where X is an observation and μ is the mean of the data set. Let us now look at an example on standard deviation.

Imagine you are trying to find how disperse the height of 5 dogs is. Figure 2.17 has the visual representation of the data.

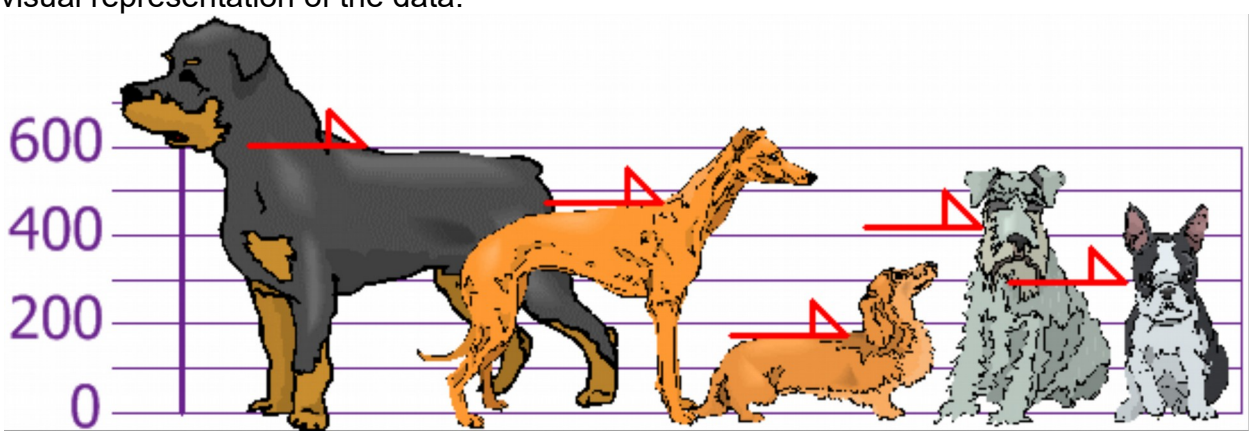


Figure 2.17. Dogs and their height.

The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm. The first step would be to find the mean of the heights, this is fairly easy, by simply doing a summation of the heights over the number of dogs we have.

$$\text{Mean} = \frac{600 + 470 + 170 + 430 + 300}{5} = \frac{1970}{5} = 394$$

Then we need to calculate the variance, which might be a bit more work, but simple nonetheless. The variance is the square difference between observed height and mean divided by the data sample.

$$\begin{aligned}\text{Variance: } \sigma^2 &= \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5} \\ &= \frac{42,436 + 5,776 + 50,176 + 1,296 + 8,836}{5} \\ &= \frac{108,520}{5} = 21,704\end{aligned}$$

Now we are ready to get the standard deviation σ, which is, if you have not noticed, the square root of the variance.

$$\sigma = \sqrt{21.704}$$
$$= 147{,}32...$$
$$= 147 \text{ (to the nearest mm)}$$

Thus we end up with $\sigma = 147\,mm$, but, what can we infer from this? Figure 2.18 will answer the question.

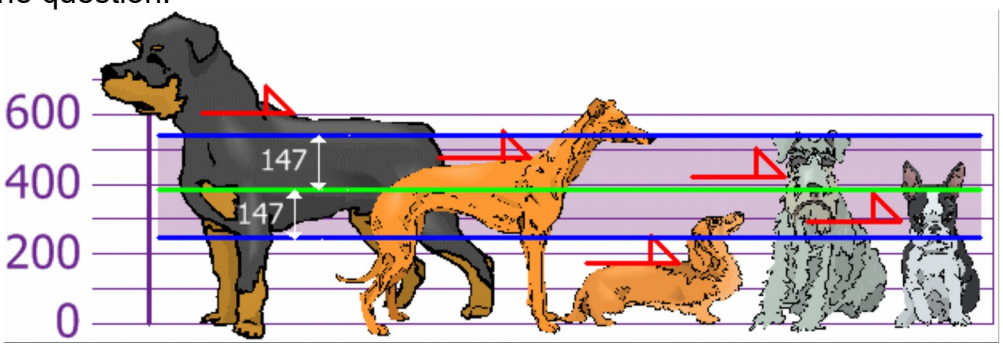


Figure 2.18. Dog heights with standard deviation.

The σ just tells us how disperse the dog heights are from the mean height. Another way to look at it is the *certainty of uncertainty,* how likely are we to find the height of a dog from the mean. Ideally, we would want our data to stay close together, this would imply a small σ value. When making forecasts, we will always try to stick between 2 standard deviations. Figure 2.19 for details.

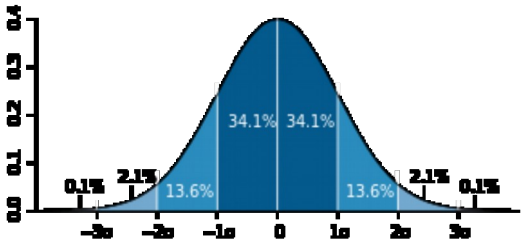


Figure 2.19. A normal distribution plot, where each band has a width of 1 standard deviation.

What does the concept “2 standard deviations” mean?
By looking at figure 2.19, we can think of the σ as a data catcher, and if we are trying to capture as much data as possible, we would want to increase this σ in order to expect the unexpected. Looking back at our dog height example, where we only use 1 standard deviation together with the normal distribution plot, we are currently catching about 68.2% of the height data, but what if we want more?

When using 2 standard deviations we increase our *capture zone* up to 95%! This means that if a new dog comes in, and we are using 2 standard deviations, we are very likely to catch the new dog's height between our boundaries with an accuracy of 95%.

It is worth mentioning, that we got the standard deviation from a population (the 5 dogs are the only dogs we are interested in), not from a sample. When we do standard deviation of a sample we just do a little change on the calculation of the variance above. If these 5 dogs were a sample from a larger set, we would do the variance division by 4 (N-1) instead of 5. The other calculations remain the same.

We will now dive into the realm of regression. We will start simple and use examples as needed. We will find some very important definitions in regression which will be extremely helpful when looking into forecasting methods.

Linear regression

Linear regression [A-11][A-16][A-12] is a statistical approach to model the relationships between a dependent variable and one or more independent variables. For a simple example, imagine you want to know how many miles per gallon (MPG) you can travel, based on the weight of your vehicle, logically, the lighter the vehicle the easier it would be to move around and you would also have a higher MPG value. The dependent variable would be the MPG and the independent variable would be the weight. Let's try and make this a bit more graphical. Figure 2.20 for reference.

	MPG	Weight
1	25.0	2800
2	26.0	2600
3	28.0	2700
4	31.0	2700
5	26.0	3100
6	25.0	3400
7	27.0	3500
8	24.0	3900
9	20.0	4600
10	22.5	4250

Figure 2.20. A data set containing a relation of the MPG and weight of some cars.

Remember that it's always a good idea to plot data to try and find patterns in it. Let's do that. Figure 2.21 is the plot.

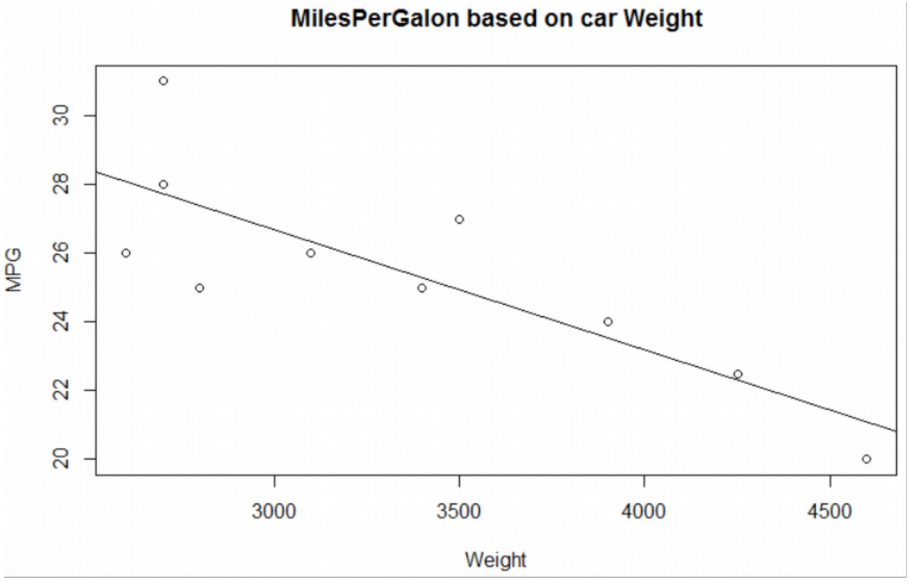


Figure 2.21. Plot of the MPG by weight with trend line.

Looking at our plot, we can see a fairly logical pattern, one that says ”the heavier the car is, the less MPG it will be able to achieve”. We also would like to know what the *rate* for this decrease in MPG really is. So let us look into a very popular regression formula in figure 2.22.

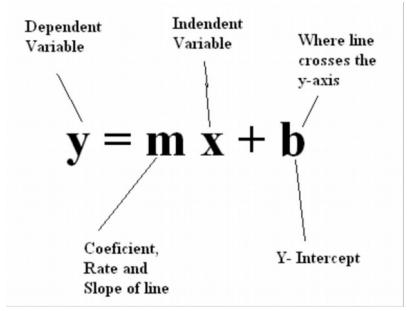


Figure 2.22. Regression formula

Let’s dissect the formula from figure 2.22.

The *y* is the dependent variable (aka. Response variable), in our case, the MPG. The *x* is the independent variable (aka. Predictor), in our case the weight. The *m* is the coefficient or *rate of change* of *y* per unit of *x.* The *b* is the Y-intercept, this is just the point where the lines cross. Using Rstudio, let’s build a linear regression model to find out the values of *m* and *b.*

Figure 2.23 contains our summary of the linear regression model.

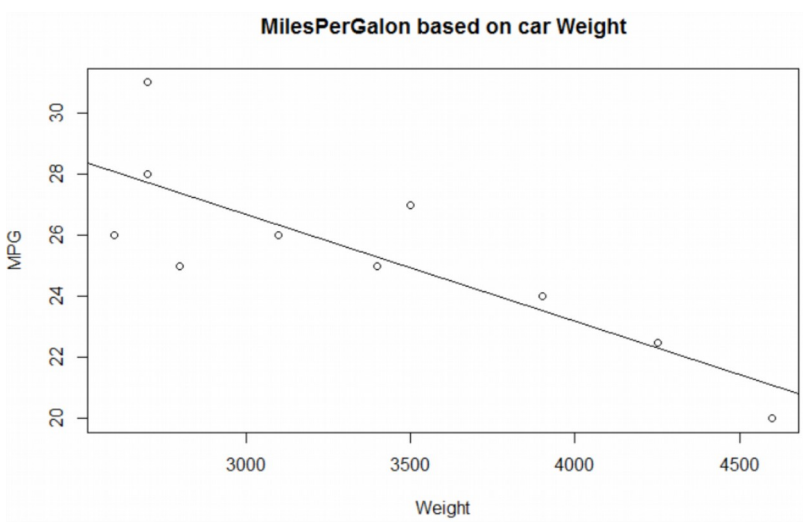
```
> summary(lm(car$MPG~car$Weight))

Call:
lm(formula = car$MPG ~ car$Weight)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.3887 -0.9109 -0.0582  0.4058  3.2619 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.1697132  2.9526571  12.589 1.49e-06 ***
car$Weight  -0.0034932  0.0008631  -4.047   0.0037 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.825 on 8 degrees of freedom
Multiple R-squared:  0.6719,	Adjusted R-squared:  0.6309 
F-statistic: 16.38 on 1 and 8 DF,  p-value: 0.003698
```

Figure 2.23. Summary of a linear regression model.

Figure 2.23 opens an entire new world of definitions, but we will go through them one by one, by the end of it, the reader will be able to perform linear regression analysis on any time series.

Call: The call outputs a formula, this is very simple to understand, basically saying y ~ x or Y is dependent on X, in our case, the MPG is dependent on the weight.

Residuals: The residuals are the difference between the observed data (real) and the predicted data. To understand this better let’s look at a plot of our data.



On this plot we see the MPG according to the weight of a car, the line passing through them is called a *trend line (aka. Regression Line, Least squares line, Best fit line, Least Squares Regression Line)*, the line is placed in a way that it fits the data as best as possible, by minimizing the error between observed and predicted. It is also used to predict future and unknown values.

The accuracy of the prediction depends on the quality of the data. Figure 2.24 below shows an example which can be used to better understand the residuals

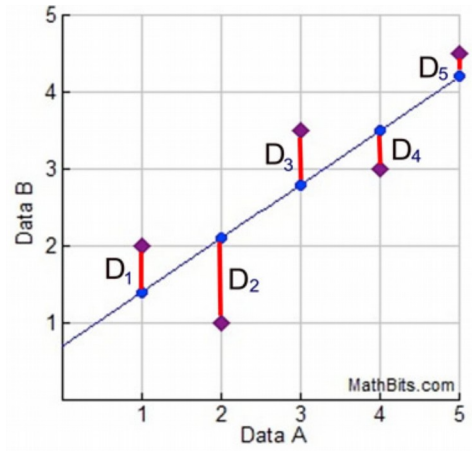


Figure 2.24. Residuals.

The red line segments represent the distances between the y-values of the actual MPG data. While the small blue dots on the trend line are the predicted y-values. The lengths of the red line segments are called **residuals**. Thus each pair of observed and predicted value has a residual. The accuracy of the forecast tends to increase if the residuals are closer to 0. [A-15]

As an example, let's look at the value obtained by the first observation minus the value obtained by the trend line at the same weight by using our regression formula from 2.21 using the information from the summary of the linear model.

Observed: 25 MPG at 2800 pounds of weight.

Predicted: $y = mx + b\,; wit\ h\, x = 2800, m = -0.0034, b = 37.1697$

$$y = -0{,}0034 * 2800 + 37{,}1697$$

$$y = 27{,}6497$$

Thus the residual for the first data point is 25 – 27,6497 = -2,6497. The same process is repeated for each data point. If we would like to know how many MPG we would achieve on a new vehicle, all we would have to do is enter the vehicle weight into the regression formula and obtain the MPG. Note that the accuracy of the predictions depends mainly on the residuals, if the residuals are small then our predictions on new vehicles will be very accurate.

Coefficients: Let us now see the coefficients. We will break this down into parts since they also offer interesting data on their own. On our summary for the linear model we have 2 coefficients, the intercept and the weight.

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.1697132  2.9526571  12.589 1.49e-06 ***
Weight      -0.0034932  0.0008631  -4.047   0.0037 **
```

Figure 2.25. Coefficients summary.

The **estimate** column gives a value to the intercept and the weight (independent variable), the intercept is the expected mean value of the MPG (dependent variable) when the weight is 0. Another way of seeing this, although a bit illogical is: When a car has a weight of 0 pounds, the MPG will be 37.16. The estimate for the weight is -0.0034. This negative number affects the MPG per unit of weight.

We read this as: *We will lose 0.0034 MPG for each pound of weight the car has.*

The **standard error** column simply shows the accuracy of the predictions made with the regression line (trend line). The smaller the Std. Error, the better. [A-14]

The **t-value** column is obtained by dividing the coefficient by the standard error. The t-value can be thought of as a measure of the precision with which the regression coefficient is measured. If the t-value is higher than 2 (or smaller than -2) we take it as a high importance value and thus useful. [A-14]

The **p-value** column is used to accept or reject the null-hypothesis [A-9]. The null-hypothesis basically states: X does not cause Y, or Y is not dependent of X. If the p-value is smaller than 0.05, we reject the null-hypothesis. In our example the p-value is definitely smaller than 0.05, thus we can reject the null-hypothesis and infer that indeed X causes Y or Y is dependent of X. [A-23] [A-21]

Remaining measures

```
Residual standard error: 1.825 on 8 degrees of freedom
Multiple R-squared:  0.6719,    Adjusted R-squared:  0.6309
F-statistic: 16.38 on 1 and 8 DF,  p-value: 0.003698
```

Some of these values are easily understood while one in particular is very complicated and not the focus of this thesis, but we will still try to give a “short” answer to it since they are all important.

Residual std. error on degrees of freedom: While degrees of freedom are not something one might *need* to perform statistical analysis, some people (research statisticians or statistical theory students) might want to learn about them, so let's try to explain them with a fun example from [A-17]:

- Imagine you're a fun-loving person who loves to wear hats. You couldn't care less what a degree of freedom is. You believe that variety is the spice of life. Unfortunately, you have constraints. You have only 7 hats. Yet you want to wear a different hat every day of the week. On the first day, you can wear any of the 7 hats. On the second day, you can choose from the 6 remaining hats, on day 3 you can choose from 5 hats, and so on. When day 6 rolls around, you still have a choice between 2 hats that you haven't worn yet that week. But after you choose your hat for day 6, you have no choice for the hat that you wear on Day 7. You must wear the one remaining hat. You had 7-1 = 6 days of "hat" freedom—in which the hat you wore could vary! That's kind of the idea behind degrees of freedom in statistics. Degrees of freedom are often broadly defined as the number of "observations" (pieces of information) in the data that are free to vary when estimating statistical parameters.

R-squared: In order to explain the multiple r-square and the adjusted r-square we need to define the r-squared. The R-squared is the value representing how close the observed data is to the fitted data obtained by the trend line. Ranging from 0 to 1, with a higher value increasing the accuracy of the prediction. There exists a possibility of negative R-squared values, these negative values are the result of a model containing terms which do not help to predict the response variable.

Multiple R-squared: The accuracy of the model. It can be seen as the percentage of the response variable variation that can be explained using the linear model. This is a highly important value and in our example it can be read as: ***The accuracy of the linear model to predict the MPG using the weight as predictor is 67.19%.***

Adjusted R-squared: The adjusted R-square is a modified version of the R-square to adjust for the number of predictors in the model. The adjusted R-square increases if new terms improve the model more than would be expected by chance, it decreases when a predictor improves the model by less than expected by chance. [A-18] [A-19]

F-statistic: Very similar to the p-value. Used to test whether or not a variable has statistical significance [20]. This is used to reject or accept the null-hypothesis. Where an f-value of 1 would accept the null-hypothesis, and a high value would reject it. It can range from 0 to a rather large number. It can be calculated by dividing the sum of the squares by the mean error sum of squares. [A-21] [A-22]

Now that we understand what our linear model summary is telling us, let us look at it one more time on figure 2.26.

```
> summary(lm(car$MPG~car$Weight))

Call:
lm(formula = car$MPG ~ car$Weight)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.3887 -0.9109 -0.0582  0.4058  3.2619 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.1697132  2.9526571  12.589 1.49e-06 ***
car$Weight  -0.0034932  0.0008631  -4.047   0.0037 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.825 on 8 degrees of freedom
Multiple R-squared:  0.6719,	Adjusted R-squared:  0.6309 
F-statistic: 16.38 on 1 and 8 DF,  p-value: 0.003698
```

Figure 2.26. Summary of a linear regression model.

We are now ready to look into some forecasting methods.

Some of the following methods are available in RStudio, a name for a library will be added if it is not by default in the program.

Naïve forecasting – (default in Rstudio)

One of the most basic forms of forecasting. It focuses on using the last known value as the next unknown value. Picture a vector ***X*** of 5 numbers we would like to forecast with ***X = [3, 1, 6, 5, 4].*** A naïve forecast on vector X would use the last known value (in this case, 4) as the next values, so if we were interested in forecasting the next 3 values, the new vector would look like ***X’ = [3, 1, 6, 5, 4, 4, 4, 4]***.

Mean Forecasting – (default in Rstudio)

Similar to Naïve, the mean forecast uses the mean value of a time series as the new values. Following our previous example from the naïve forecast, the new vector would look like ***X’ = [3, 1, 6, 5, 4, 3.8, 3.8, 3.8].*** Where the value 3.8 is the mean value of the original vector.

Drift forecasting– (default in Rstudio)

A variation of the naïve method. The drift forecast tries to use the average change of the time series to either increase or decrease the forecast. Using drift with an increase argument, the new vector would look like: ***X’ = [3, 1, 6, 5, 4, 4.25, 4.50, 4.75]***

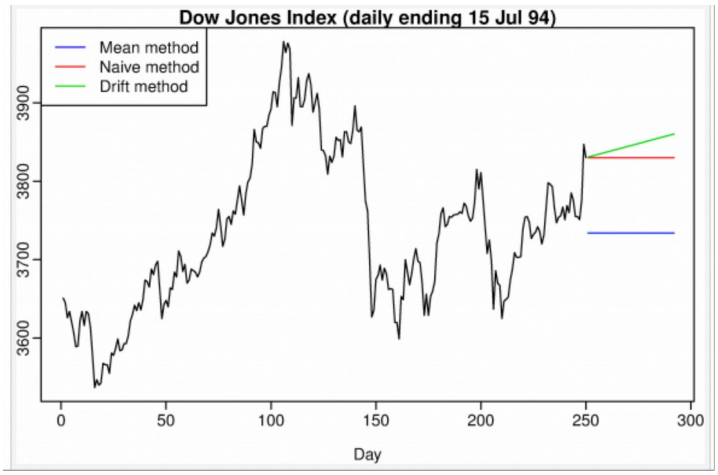


Figure 2.27. A figure showing the 3 forecasting methods explained above [A-30].

ARMA – (library tseries)

Similar to how multivariate analysis is the analysis of relationships between multiple variables (which we will see later in the thesis), univariate analysis is a quantitative analysis of only one variable. When you model a univariate time series, you are modeling time series changes that represent changes in a single variable over time.

Autoregressive Moving Average (ARMA) is a class of forecasting methods that can be used to predict future values from current and historical data. As the name implies, the ARMA models combine **autoregression techniques** (analyses that assume that previous observations are good predictors for future values and perform an autoregression analysis to forecast for those future values) and **moving average techniques** — models that measure the level of the constant time series and then update the forecast model if any changes are detected. [21]

Let's try and break down the ARMA model for easier understanding

Autoregression – (library stats)

An autoregressive model is a stochastic process in which future values are estimated based on a weighted sum of past values. An autoregressive process operates under the premise that past values have an effect on current values.

A process considered AR(1) (p = 1) is the first order process, meaning that the current value is based on the immediately preceding value. An AR(2) (p = 2) process has the current value based on the previous two values.

As an example, we might have y a measure of global temperature, with measurements observed each year. To emphasize that we have measured values over time, we use "t" as a subscript of time. Such that y_t means y measured in time period *t*. An

autoregressive model is when a value from a time series is regressed on previous values from that same time series, for example, y_t on y_{t-1} , on figure 2.28:

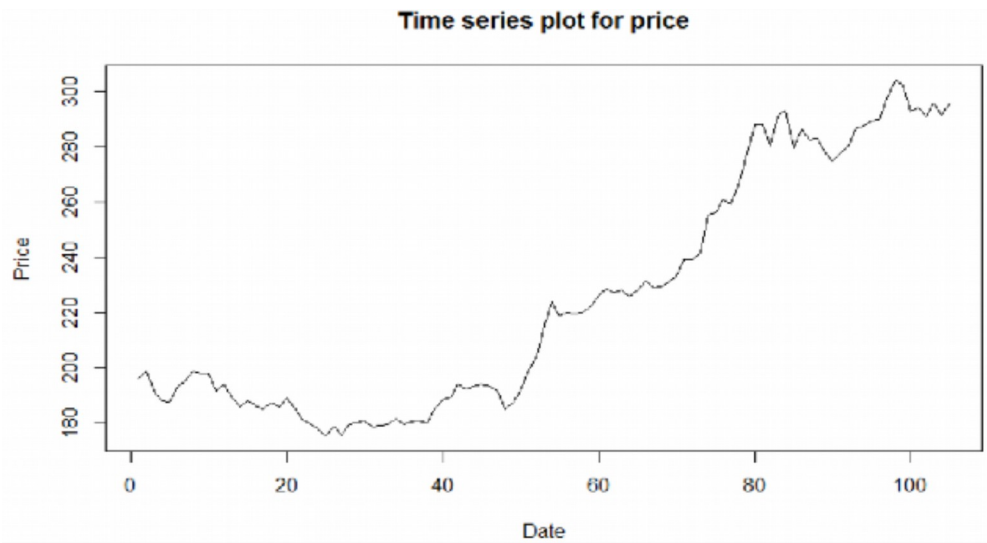
$$y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t.$$

Figure 2.28. Regression model formula.

Figure 2.28 shows an AR model with *p = 1.* Where the response variable from the previous time period becomes the predictor of the current period. The $ß_0 \wedge ß_1$ are a constant and a coefficient respectively and finally the ϵ_t is the white noise or randomness on the series.

Time to get our hands dirty with an example.

Taking a closing stocks data set from [St-Go],we will try and make an autoregressive model. Figure 2.29 for a glimpse of the data, together with a plot of the data itself.



	date	price
1	2/7/2005	196.03
2	2/8/2005	198.64
3	2/9/2005	191.58
4	2/10/2005	187.98
5	2/11/2005	187.40
6	2/14/2005	192.99
7	2/15/2005	195.23
8	2/16/2005	198.41
9	2/17/2005	197.90

Figure 2.29. Left: Plot of closing stocks data. Right: Closing stocks data set.

At first glance, and with our previous knowledge on stationary time series, this data does not seem to be stationary, meaning that the mean and variance might not be constant, however, let us do a PACF (partial autocorrelation function) on this data to find out if we are correct.

Figure 2.30 shows the ACF plot.

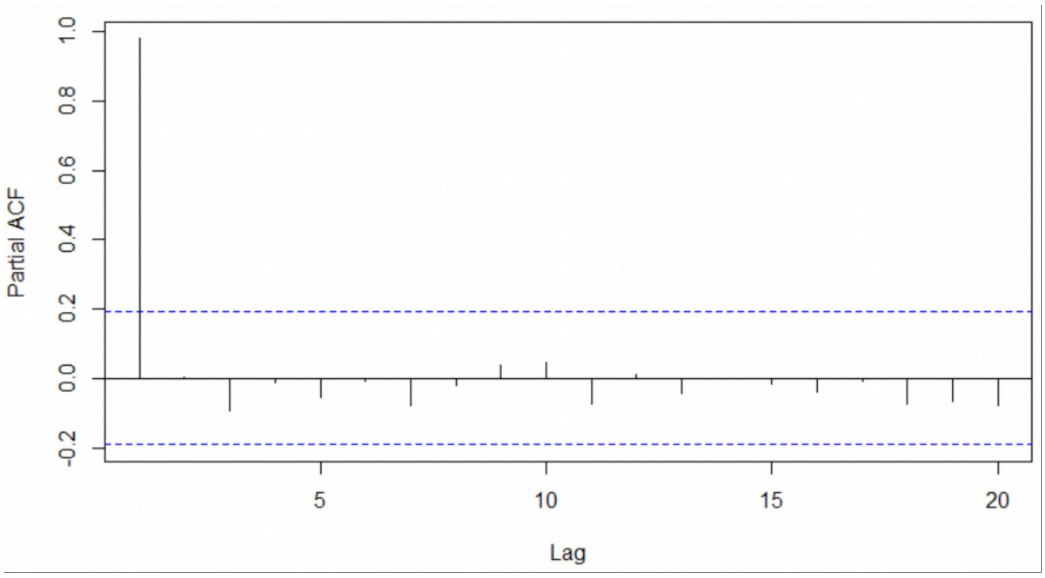


Figure 2.30. Partial Autocorrelation Function plot, where the PACF value cuts off after 1 lag.

The partial autocorrelation function (PACF) gives the partial autocorrelation of a time series with its own lagged values, controlling for the values of the time series at all shorter lags. It contrasts with the autocorrelation function (ACF), which does not control for other lags. What is important here is to note two things, the **PACF will help find the value of *p*,** while the **ACF will help find the value of *q*.** It is good practice to try both methods to find out the proper values for *p* and *q*. There exists an automatic way to get these values without the need for ACF and PACF, we will discuss this when we dive into ARIMA, but it is always a good idea to double check using ACF and PACF.

Now that we have obtained the value of *p* for the AR part, let's build a small AR model and see the summary. Figure 2.31 has a call to the AR function and a summary.

```
> ar(stocks$price, order.max = 1)

Call:
ar(x = stocks$price, order.max = 1)

Coefficients:
     1
0.9792

Order selected 1  sigma^2 estimated as  76.59
```

Figure 2.31. Summary of an autoregressive function.

We see here some terms we met before, the coefficients and a standard deviation of grade 2, with a value of 76.59 for the predicted values, not to be confused with the deviation for the forecast of future values. Notice the order.max parameter, this is equal to 1, since we have found that the proper AR value for *p* is 1 by our PACF. If we take a look at some of the residuals of the AR model, we could find some more information

about how accurate we can predict known values. Figure 2.31 for details on the residuals.

	residuals_ar
1	NA
2	2.01474128
3	-7.60109757
4	-4.28760243
5	-1.34230745
6	4.81565674
7	1.58165704
8	2.56814017
9	-1.05587040
10	-0.50645361

Figure 2.32. Residuals of the AR model for closing stock prices.

Notice the first value is *NA,* this is due to the fact that, since the model tries to regress one step each time, it eventually reaches a point of no regression and thus generates an empty value. What is important here is that the residuals are very close to 0, considering the fact that the actual data is in the ranges of 180 to about 300, this is a fairly accurate prediction. Recall that the residuals would ideally be 0 in a perfect fit.

The question we have to ask ourselves now is: can we make future predictions about stock prices?

The answer is a very loud YES! Let's do that right now. Figure 2.32 shows a prediction on the next 10 values for the closing stock prices.

```
> predict(ar(stocks$price, order.max = 1), n.ahead = 10)
$pred
Time Series:
Start = 106
End = 115
Frequency = 1
 [1] 294.0703 292.6311 291.2217 289.8416 288.4902 287.1667 285.8708 284.6017 283.3590
[10] 282.1421

$se
Time Series:
Start = 106
End = 115
Frequency = 1
 [1]  8.75181 12.24918 14.84837 16.97094 18.78238 20.36867 21.78157 23.05525 24.21378
[10] 25.27490
```

Figure 2.33. Prediction call on the AR model for the next 10 stock prices.

Let's break this down. The *predict* function is a well-defined function, it is capable of understanding what kind of model we are asking it to make predictions on, in this case, the predict function understands that it will work on an AR model. The parameter *n.ahead*simply tells the predict function to predict the next 10 values ahead. It returns the next 10 values in a column called *$pred*, together with the standard error for each value in a column called *$se.* In order to see if this prediction makes sense, let us try to plot it together with the original data. Figure 2.34 for details.

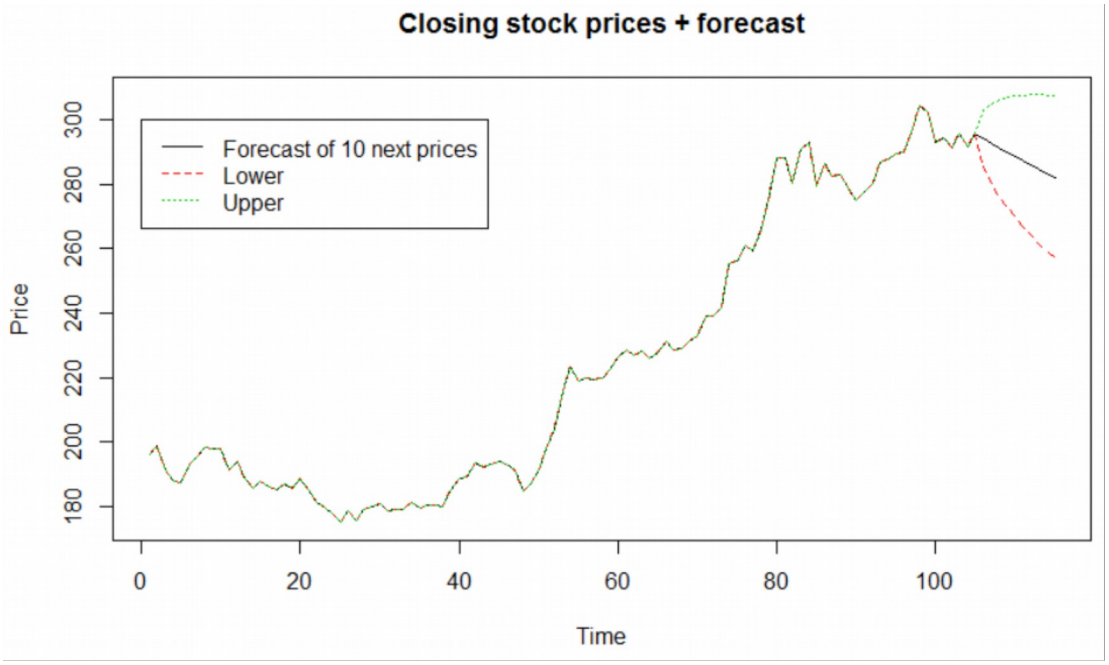


Figure 2.34. Forecast of closing stock prices.

The forecast came out looking like the drift forecast, however, that's what the upper and lower bounds are for, they represent the certainty of uncertainty, the deviation of the forecast. Let us now move on to the second part of ARMA, the MA (moving average) model.

Moving Average – (library forecast)

The MA algorithm takes a time series and a *frequency q* parameter, this is used to take samples of size qfrom the time series and generate an average from them. MA then moves to the next observation of size *q* and generates another average from the time series. After no more samples can be made, a new time series using the average samples is used for forecasting. This algorithm is most commonly used in economics and in the stock market [A-31].

Take for example a vector ***X = [3, 5, 4, 1, 3, 5, 6, 8, 4]*** with an MA *q* = 3.

MA will take the first 3 values (3, 5, 4) from **X** to calculate the average and generate a new vector ***X' = [4],*** then it will take the second 3 elements (5, 4, 1) and calculate the average. Thus, ***X' = [4, 3.33,].*** Then the next 3 elements (4, 1, 3) and ***X' = [4, 3.33,***

2.66]. Until we have a final ***X' = [4, 3.33, 2.66, 3, 4.66, 6.33, 6].*** Then we just use a *predict* function from R which takes this data and predicts the next *n* steps. Figure 2.34 for details.

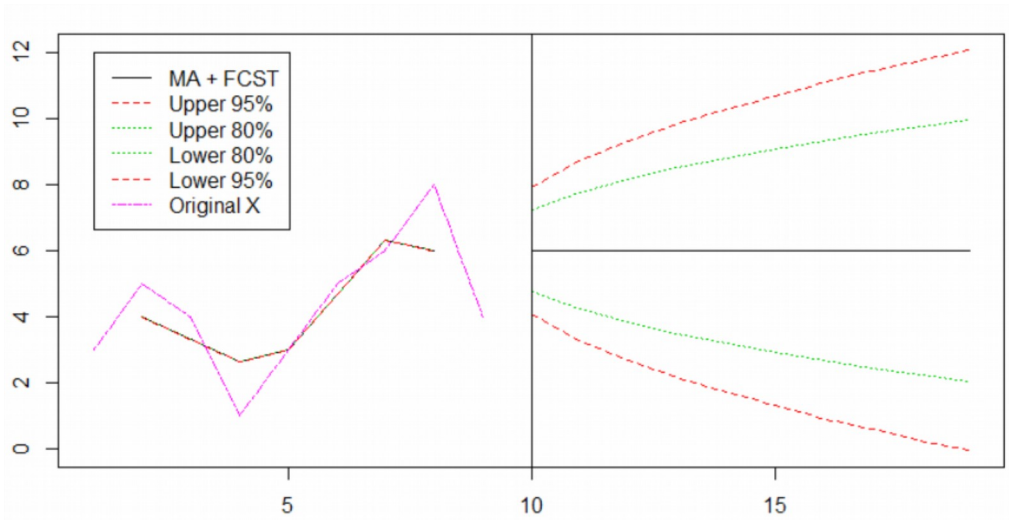


Figure 2.35. This figure shows the final ***X'*** values with an MA forecast starting at the vertical line. The Upper/Lower lines represent the confidence level of the forecast.

ARIMA – (library stats)

ARIMA (Autoregressive Integrated Moving Average) is a well-known analysis model method for forecasting time series. Often used to understand the way data behaves, but other times used to make future predictions on a time series. It is a generalization of the ARMA (Autoregressive Moving Average) model. ARIMA is often times used when the data shows evidence of non-stationarity, but by applying a simple modification using a differencing step, can turn the non-stationary data into stationary. The big difference here comes from the "I" part of ARIMA, namely the Integrated part. As seen before with ARMA, ARIMA also has the p and q values for AR and MA respectively, however, it consists of a value *d* for the I, which is the integrated part of ARIMA. Whenever we find a non-stationary time series, in order to transform it into a stationary one, we use a differencing of the time series values. Let us look at an example.

Recall the air passengers data, figure 2.36 for a reminder.

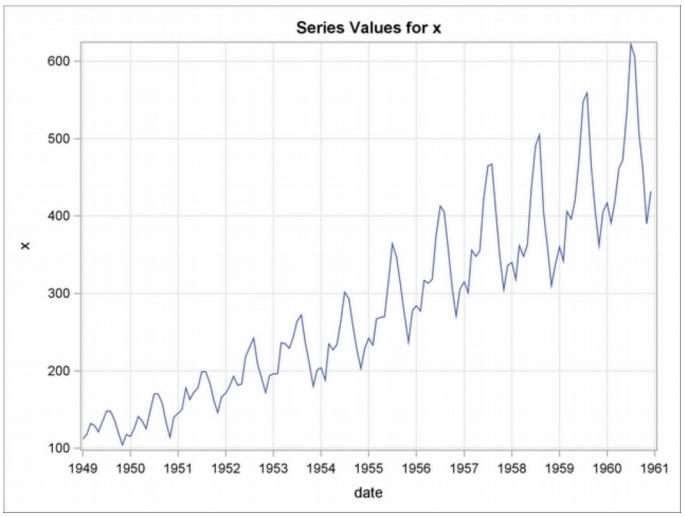


Figure 2.36. Air passengers time series data, measured monthly.

If we do both, a PACF (partial autocorrelation) and ACF (autocorrelation) plots, we will find some interesting information. Figure 2.37 for both plots.

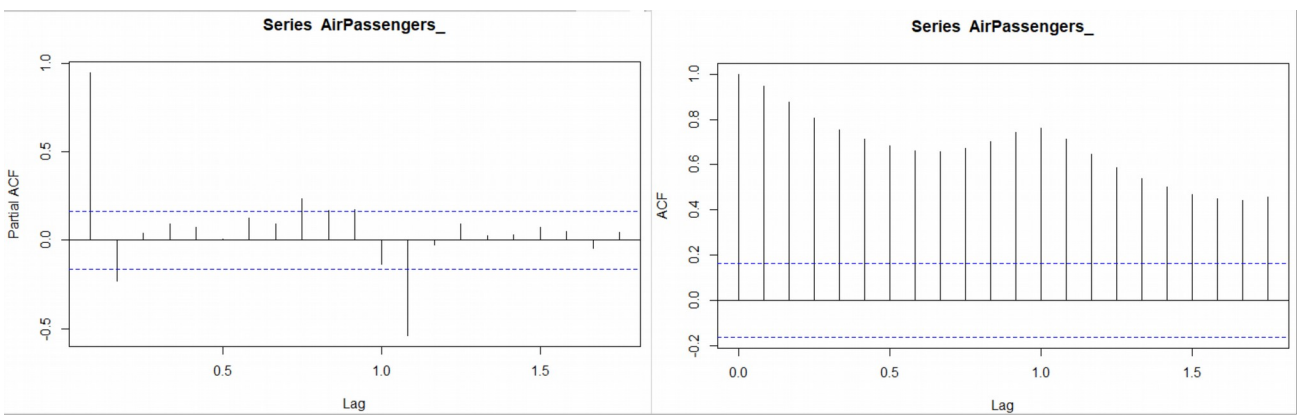


Figure 2.37. PACF and ACF plots for air passenger's data.

As seen before, the PACF is used to obtain the AR (p) value for a time series, and the ACF is used to obtain the MA (q) value for a time series. The value used is the lag value from each plot, where the lines start to settle down between the horizontal dotted line at the 0.2 and -0.2 correlation value, these data boundaries represent statistical significance. So for the p value we would use a 2, but what about q? The ACF plot does not seem to smooth down at all, this is why we need to do a differencing of the time series. The idea is to take the difference from the first value minus the second, the second minus the third and so on. This will generate a new time series using the differenced values, thus, integrating the time series. We will use the *diff* function from R (available by default) and we will plot the PACF and ACF again.

Figures 2.38and 2.39 for details.

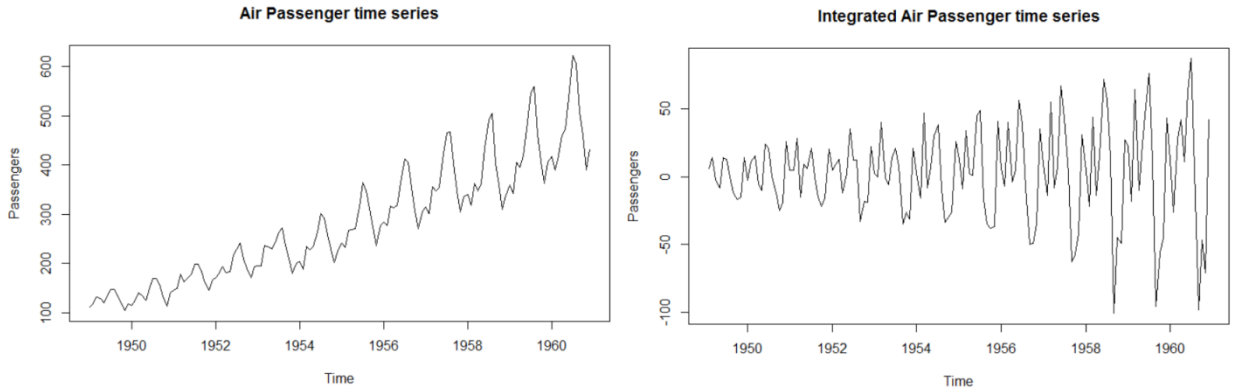


Figure 2.38. Original non-stationary air passenger data (left). Integrated stationary air passenger data (right).

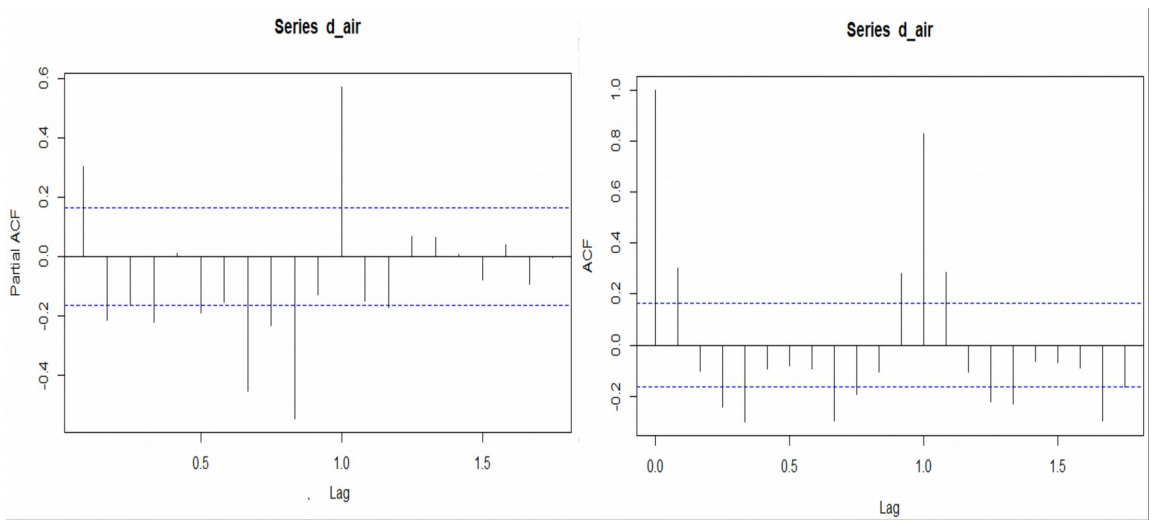


Figure 2.39. ACF and PACF plots of the integrated time series.

Now the ACF plot provides something different, the AR and MA values for p and q respectively are both 2. Since both plots have their correlation value drop between the dotted lines after the second lag. Recall that the *p* value just represents how the current value depends on the last 2 values (in this case) and the *q* value is going to be the number of elements to use for the average when creating a new series, in this case, 2.

So now let's try to make a forecast of the next 2 years with them without forgetting that we used an integration, so the ARIMA model parameters *p, d* and *q* will respectively be (2,1,2). For this forecast we will need the library *forecast* installed in Rstudio.

Before we move on to the forecast plot, let's first explain the Rstudio commands we used. Figure 2.40 for details.

```
> fit_pa <- arima(AirPassengers, order=c(2, 1, 2), seasonal = list(order=c(0, 1, 0), period = 12))
> fore_pa <- predict(fit_pa, n.ahead=24)
```

Figure 2.40. Call to the ARIMA function in R together with prediction.

ARIMA has many parameters, mainly it will need a data set and an order for *p, d* and *q* to work, however, since we are using a seasonal time series, we need to address that in the seasonal parameter, we add an object called list, where the order represents the integrated part we did before, ignoring the *p and q.* we then add the period of the time series. We then call our trusty predict function, the n.ahead parameter is calling for the next 24 measurements (months) to be predicted on the fit_pa object, the fit_pa object is an ARIMA object, the predict function knows this. Figure 2.41 shows the plot of our forecast.

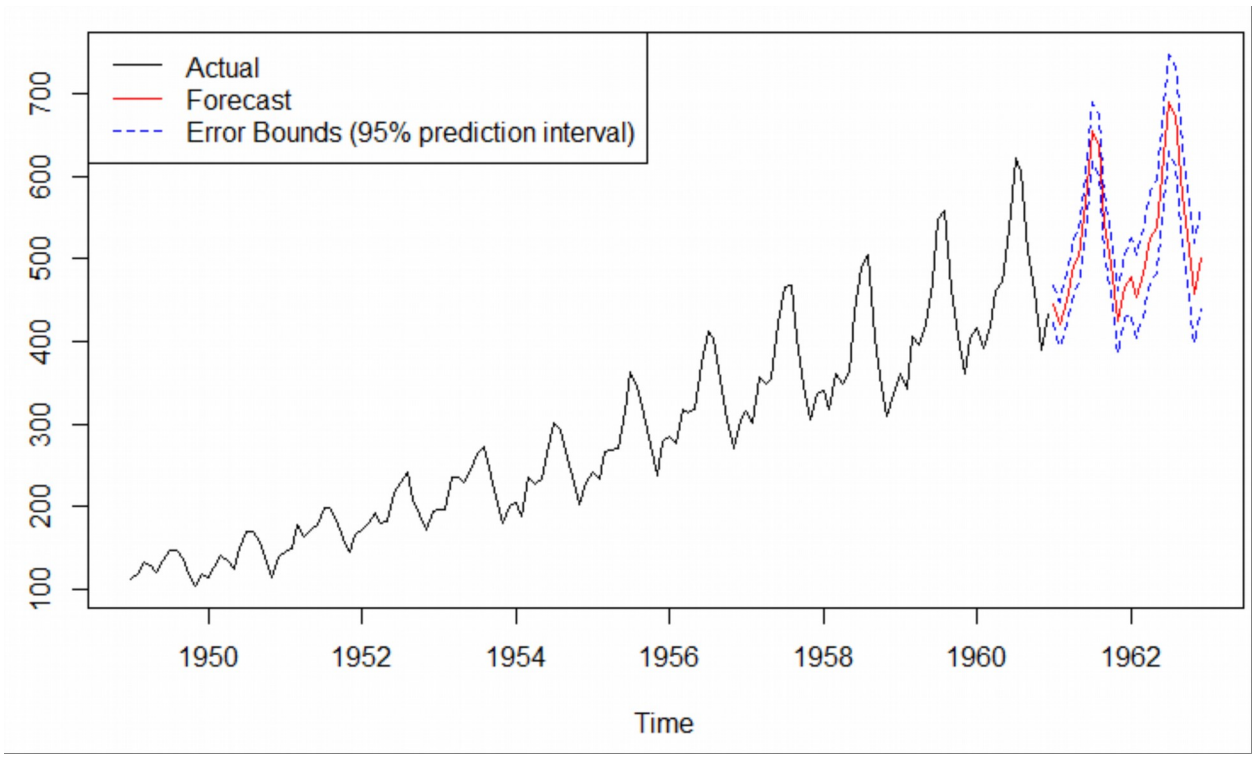


Figure 2.41. Forecast for the next 24 months of the air passengers data.

We finally did it! We now have the forecast of the next 2 years for our time series data. Notice the error bounds surrounding the forecast, this is, as said before, the certainty of uncertainty or deviation of the forecast. If in the months ahead the values do not stand exactly in the red lines of the forecast, they will surely be found within the blue dotted lines with an accuracy of 95%.

Let's recap all we had to do to perform this forecast, we first had to analyze the time series to find whether or not it was stationary, if it is, the forecast is simple, if it is not, then we need to do a little bit of a transformation on it, in the end we just had to get the proper AR and MA values for p and q respectively. However, there is an easier way of doing this, using the function *auto.arima*.

Auto.arima function – (library forecast)

This very simple function takes a time series as a parameter, this is very useful when trying to look at multiple time series inside of a loop, something that we will do in the following chapter. Let's look at an example using the air passenger's data.

```
> auto.arima(AirPassengers)
Series: AirPassengers
ARIMA(2,1,1)(0,1,0)[12]

Coefficients:
         ar1     ar2      ma1
      0.5960  0.2143  -0.9819
s.e.  0.0888  0.0880   0.0292

sigma^2 estimated as 132.3:  log likelihood=-504.92
AIC=1017.85   AICc=1018.17   BIC=1029.35
>
```

Figure 2.42. Auto ARIMA function with the air passengers data.

Let's now dissect the information obtained. The most relevant part is on the top, ***ARIMA(2,1,1)(0,1,0)[12]***, in short, we read this as: ARIMA model with *p = 2, I = 1, q = 1,* with seasonal data as (0,1,0) and frequency 12.

We can also look at the coefficients for the regression of the AR part, together with the MA coefficient. We can see that this auto.arima model returned a different value for the MA value of *q.* Let's try and forecast a model using these parameters and see what the difference is. Figure 2.39 for details on how to create the new ARIMA model. Figure 2.43 for the call on the function, figure 2.44for a comparison between both models.

```
> fit <- arima(AirPassengers, order=c(2, 1, 1), seasonal = list(order=c(0, 1, 0), period = 12))
> fore <- predict(fit, n.ahead=24)
>
```

Figure 2.43. New ARIMA model with new order for p, d, q parameters.

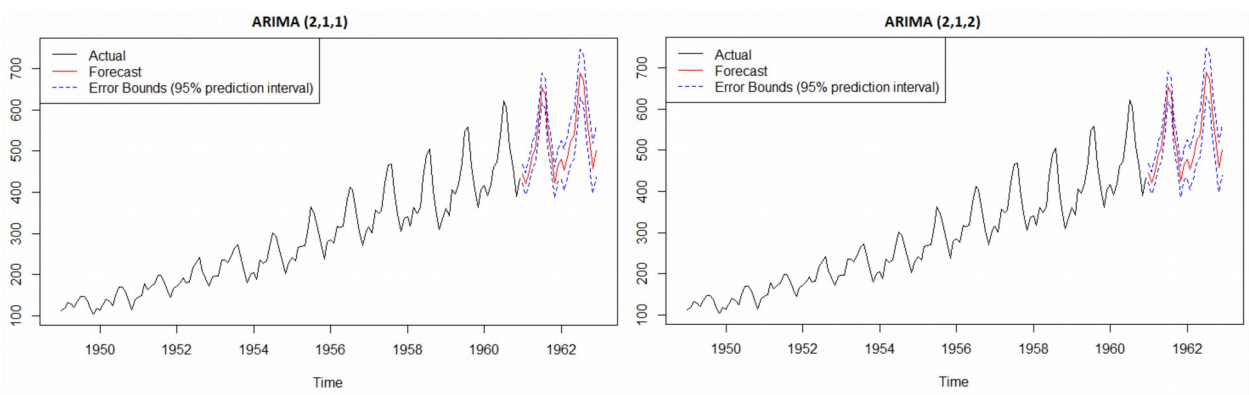


Figure 2.44. Left: ARIMA model with order of (2,1,1) using auto arima function. Right: ARIMA model with order of (2,1,2) using ACF and PACF plots to obtain parameter values.

From figure 2.44 we can see that there is not that much difference in terms of forecasting accuracy. However, we can try some already seen techniques, we can see how accurate the predictions were made on the observations by measuring the sum of the residuals, where the model with the least residuals might be the most accurate when it comes to forecast into the future. We will use simple summations from Rstudio, it is important to note that the ARIMA objects have many different properties, one of them is the residuals, recall that the residuals is the difference between observed and predicted (with a trend line or regression formula). Figure 2.45 for summation of residuals.

```
> sum(fit$residuals)
[1] 193.2912
> sum(fit_pa$residuals)
[1] 194.3998
>
```

Figure 2.45. Sum of the residuals, where the ARIMA object *fit,* the one created using the auto.arima function is much more accurate (although not by much) than the one we obtained using the PACF and ACF functions the *fit_pa*.

The reader is encouraged to try both methods, at least to try and understand which values for p and q are the correct ones for a time series.

There exist 2 more forecasting methods which will help us fulfill our task. The vector autoregression (**VAR**) model and the multivariate adaptive regression splines (**MARS**) model. Before we dive into them, let us first define one more thing, *Granger causality.*

Granger causal – (library lmtest)

Called after Clive Granger, the Granger test is used when dealing with multiple time series, for that, the Granger causal test is a statistical hypothesis test for determining if one time series can be used in order to predict another time series. Take a look at figure 2.46 for details.

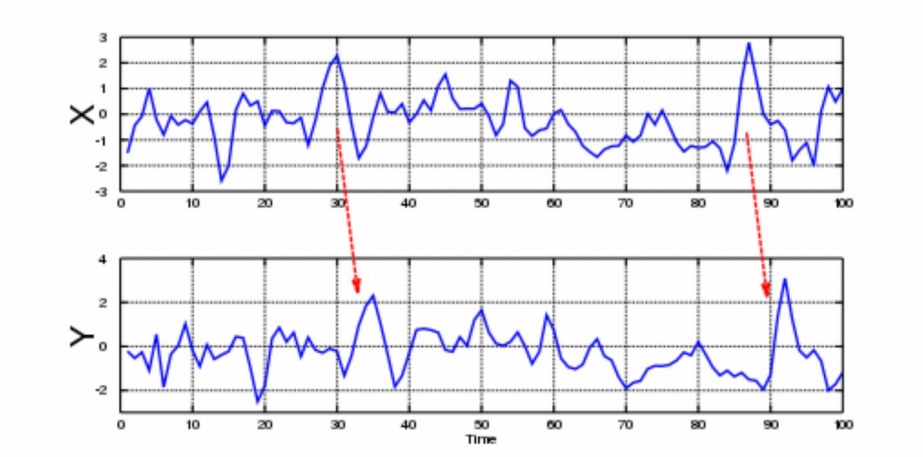


Figure 2.46. X is a causal of Y.

From figure 2.46 we can see the main idea of the Granger test, where the dependent series Y is Granger caused by the independent series X. Therefore, series X could be used to predict the values of Y. Whenever we perform a Granger test on a pair of time series, we will look for a p-value which helps us reject the null-hypothesis. The null-hypothesis states "*X is not the cause of Y",* so when we obtain a p-value smaller than 0.05, we consider it statistically significant and reject the null-hypothesis, thus stating *"X is indeed the cause of Y".*

Let's look at an example provided by the library lmtest in Rstudio, which caused what, does the chicken cause the egg, or does the egg cause the chicken? The data set that we will use is from the US chicken population and egg production. Figure 2.47 for details on the data.

	chicken	egg
1	468491	3581
2	449743	3532
3	436815	3327
4	444523	3255

Figure 2.47. Chicken and egg data set, where the chicken column represents the chicken population per year, and the egg column represents the amount of eggs in millions of dozens.

Let's now perform a Granger causal test to see which causes which came first, based on the data.

```
> grangertest(egg ~ chicken, order = 3, data = ChickEgg)
Granger causality test

Model 1: egg ~ Lags(egg, 1:3) + Lags(chicken, 1:3)
Model 2: egg ~ Lags(egg, 1:3)
  Res.Df Df      F Pr(>F)
1     44
2     47 -3 0.5916 0.6238
> ## eggs granger-cause chickens?
> grangertest(chicken ~ egg, order = 3, data = ChickEgg)
Granger causality test

Model 1: chicken ~ Lags(chicken, 1:3) + Lags(egg, 1:3)
Model 2: chicken ~ Lags(chicken, 1:3)
  Res.Df Df     F   Pr(>F)
1     44
2     47 -3 5.405 0.002966 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

Figure 2.48. Granger causal test on the chicken&egg data set.

The Granger test returns a lot of information, however we only need to focus on the p-value column, or as the Granger test calls it, the *Pr(>F)* column. We can also see the f-value column on the left, recall from the linear regression summary seen before, that we

consider both the f-value and p-value statistically significant if the f-value is greater than 2 or smaller than -2, and the p-value is smaller than 0.05. In our case, when trying to find if the chicken is causing the egg, the test fails with a p-value of 0.6238, however when we ask if the egg is causing the chicken, the test is successful with a value of 0.002966.
The main idea here, when it comes to semiconductor manufacturing, is to find which tools or operations are working together, since our time series will be made up of the lots being processed (WIP), that is to say, which tools or operations are working with the same amount of lots, we can then trace down the path that the lots travel inside the factory.
We are almost there, we just need to define our 2 forecasting methods, VAR and MARS, they have some specific requirements to work, that is why we had to go through all of the previous definitions, they will be very useful when trying to get VAR and MARS working at the top of their game. Because of that, we will hold the examples for the next chapter, since we will start working on real semiconductor data.

VAR – (library vars)

Vector autoregression tries to find the linear inter dependencies between a set of time series, because of this, VAR requires more than one time series when we make a call to it, it also needs to know the lag order to use when trying to find the inter dependencies of the series.

The most important thing that the reader needs to understand for VAR is that, it does **not** need to know the future in order to make future predictions.

MARS – (library earth)

The earth R package [MRS-D] builds regression models using the techniques in Friedman's papers "Multivariate Adaptive Regression Splines" [MARS-1] and "Fast MARS" [MARS-2]. The term "MARS" is trademarked and thus not used in the name of the package.

The most important thing that the reader needs to understand for MARS is that it **needs** to know the future in order to make future predictions.

Let's now do a recap on everything we learned so far.

We first learned about the semiconductor industry and an overview on the process to make integrated circuits. We learned about tools and operations, we also saw a very interesting concept known as holistic information model which together with the industry 4.0 could lead to great possibilities.

We then moved into the realm of statistics and learned about forecasting, time series and their properties, we learned about forecasting methods and finally applied one on a real life scenario. But what about production forecasting?

Production forecasting tries to predict the future in a production process. More specifically, the basis for any project-based resource estimate, and the same production forecast is also the basis for any business or development decision. Thus, forecasting aligns resource estimation, decision making and business planning and perhaps even operational short-range forecasts. But how does that relate to the semiconductor industry?

Recall the WIP. The WIP is a measure of how many lots are being processed at a specific moment in time by a tool or an operation. We already know what a time series looks like, it requires a discrete measure in order to work, and the information regarding the WIP comes into the database all the time, thus it is a continuous measure, however, with a small trick into the selecting query in the databse, and thanks to the malleability of the HIM, we are capable of transforming this data into a discrete series, which would then lead us to an analysis of the tools and operations inside a factory, finding the correlation between them and eventually, a production forecast model.

Two birds, one stone, and many definitions later, we answered 2 of our 3 hypothesis questions. Only one remains, which states:

- How can we create a system that is capable of predicting the WIP on different tools and/or operations? Is it reliable or accurate or even possible?

This question will be answered on chapter 3, semiconductor data analysis and model implementation. We are not out of the woods yet.

An holistic approach to production forecasting

We are now ready to get our hands dirty with some real data. However, before we move on, we need to explain the ideal model that we would like to end up with, for that let's first explain what the CIM (computer integrated model) for semiconductor manufacturing.

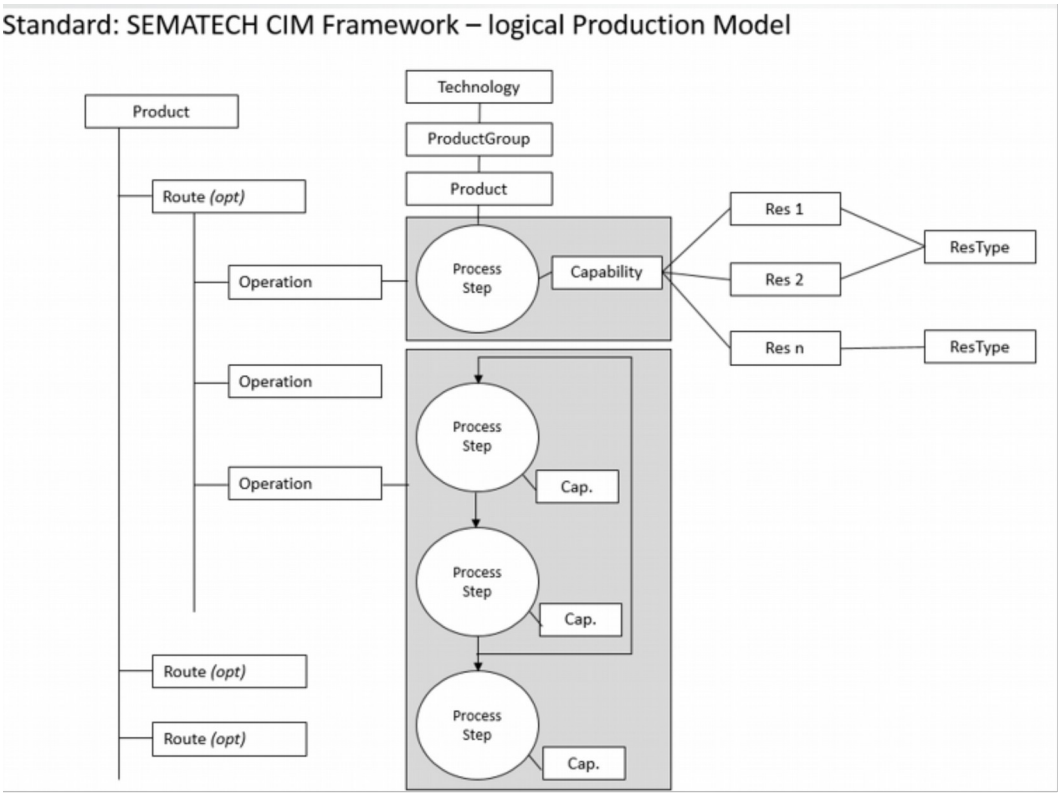


Figure 3.1. The SEMATECH CIM Framework for semiconductor manufacturing.

Figure 3.1 is basically the *flow* that products need to go through in a semiconductor factory, where we start at the top with a product, the product then follows a route and the route itself is made up of different operations which need to be performed on the product during process steps, the process steps represent the tools and the capabilities are the resources that the tools can use. Imagine a car manufacture, where a car travels through an assembly line, once it reaches a specific operation the car might need a special color of paint used on it, however, the process tool might have been used by a car which needed a different type of resources, another color, the car might now need to wait a moment at a process step while the capabilities are set for its own specifications. A similar situation happens with semiconductors.

Recall that the HIM tries to gather data from every area inside the factory, groups it into atomics, fundamentals and fractals and from there it can be displayed in very neat ways for the users to learn more about what is happening in the factory, from a very general

point using the atomics which show event data, and drill down all the way untilwe know which operation worked on which lot at what time, in what area and by which operator.

Such a system uses a very dynamic collection process, figure 3.2 for details.

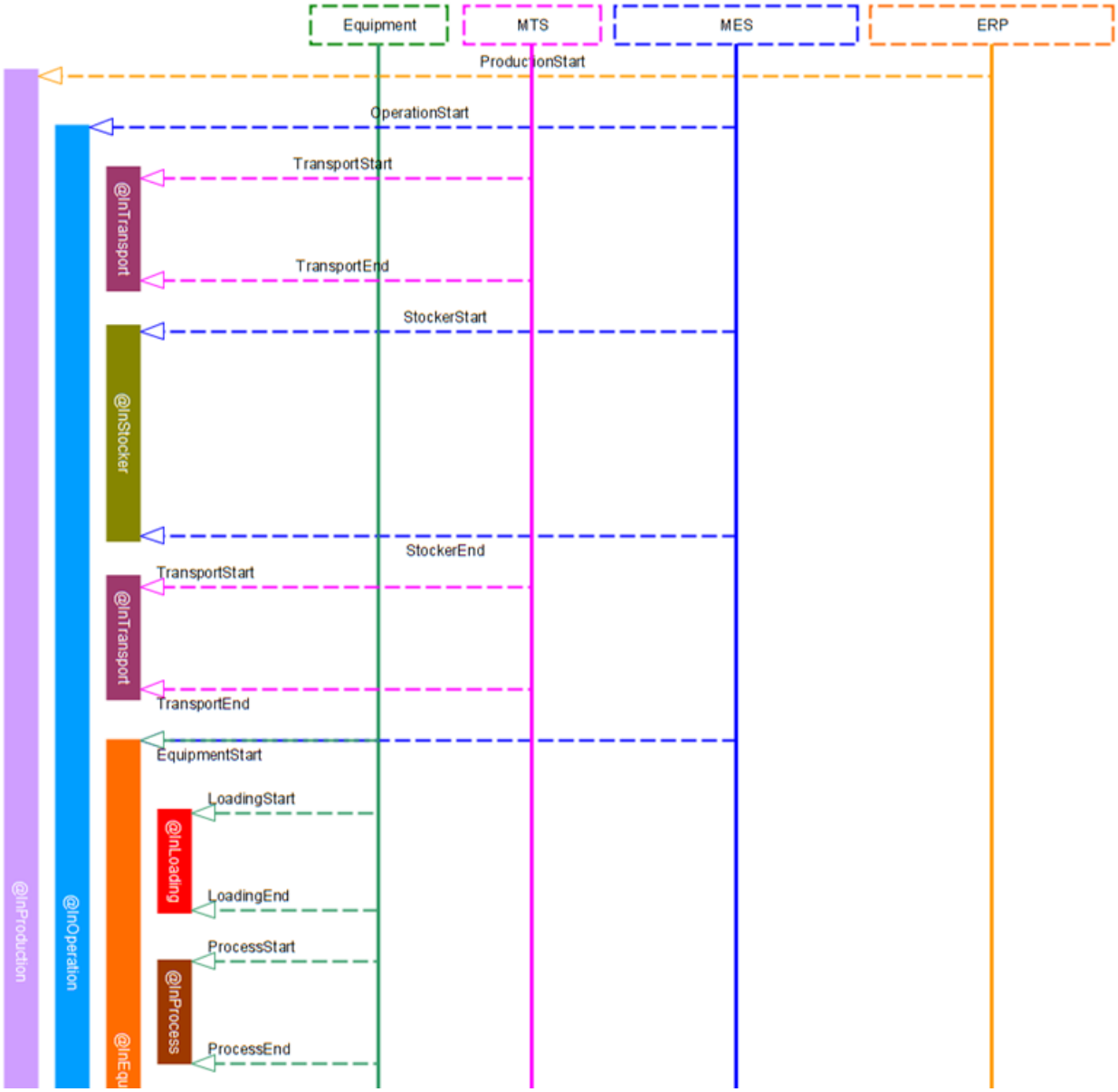


Figure 3.2. Dynamic data processing by the HIM.

Let's explain figure 3.2 in detail. The ERP (Enterprise Resource Planning) sends a message when production has begun, for some specific product. The MES (Manufacturing Execution System) notifies the system that an operation on the lot has begun, the lot the moves around the factory looking for a tool to process it. As the lot moves through the assembly line, the MTS (Manufacturing Transport System) keeps track of the movement of the lot by the addition of some metadata such as, which area of the factory the lot is at (Lithography, Etching, etc.). Each and everyevent will be captured by all systems involved. The lot continues moving until it reaches a stock area,

from there it is moved again into a tool (Equipment), process starts and the lot remains in the tool until the process ends. Once the process is done, the lot then moves towards the next operation in its route. The ERP, MES, MTS and Equipment systems are all sending information to the database in real-time, this means that whenever an event that involves this lot takes place, data is sent into the database with all the necessary information related to it, the data can then be processed by other software or accessed by users if they so desire.

All of this leads to a Holistic Production Forecasting model, figure 3.3 for details.

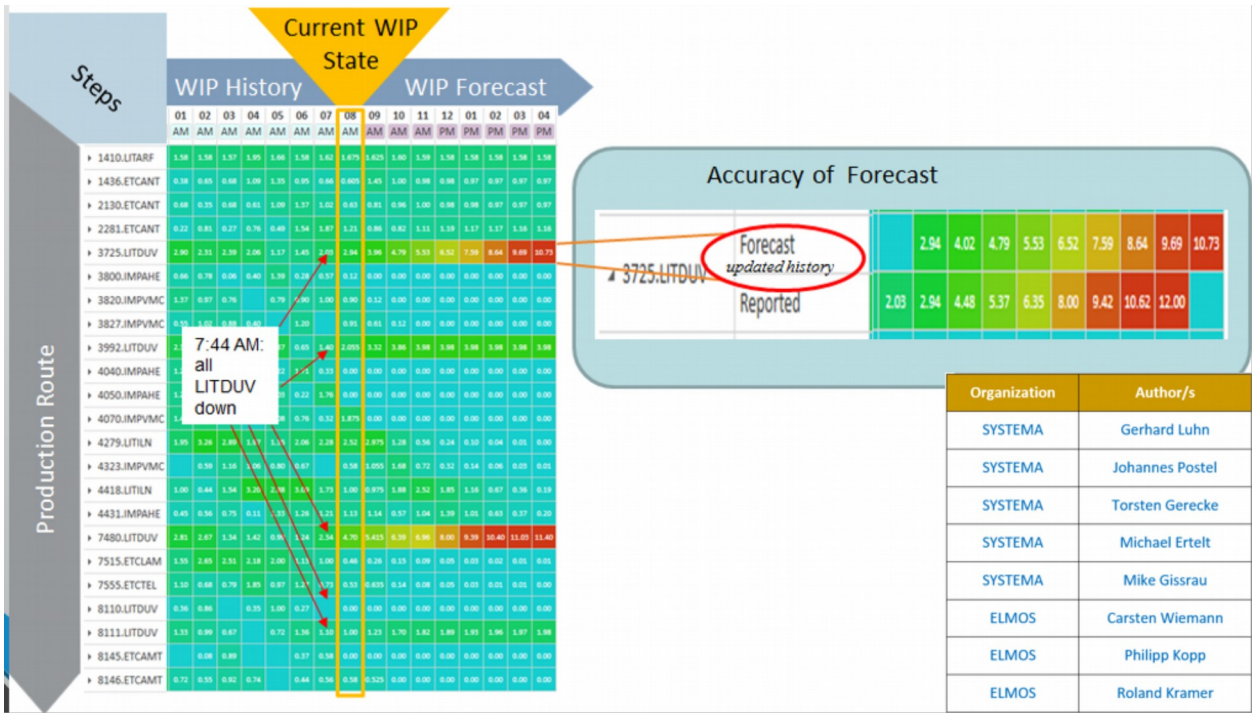


Figure 3.3. Holistic Production Forecasting model.

The HPF is the beginning of the answer for the final question we have. In the HPF we have a production route denoted by operations, each operation has a value for the WIP at a specific time, recall that we started with continuous data, but by doing the average summation of lots processed by operations and tools, we can transform the data into discrete measures, which we can use to then start forecasting the WIP per operation.

We now have our target, we just need to analyze the data from a semiconductor factory and turn it into an HPF model, however, while doing this analysis, a new idea came out and was designed by the author, this idea takes the HPF model a step further. Recall that tools can sometimes have overlapping operations. The new HPF is capable of forecasting operations, just like the old HPF model does, but the main point lies in the tool forecast, where a route still exists based on the operations that need to be performed on a product, with the extension of tools that can perform this operations and the WIP of those tools which is then forecasted. Figure 3.4 for details on the new HPF model.

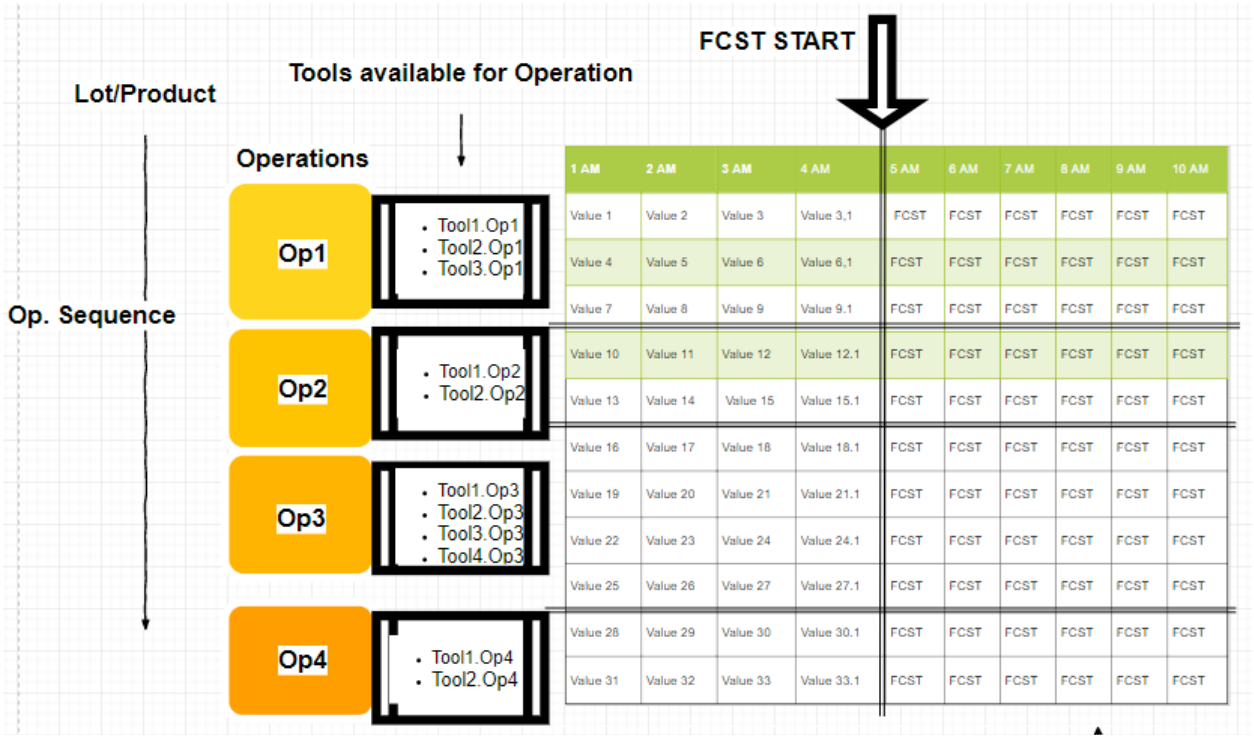


Figure 3.4. New Holistic Production Forecasting model, where the tools capable of performing each operation are added into the model, together with the forecast for each tool.

We now have our target, with the help of the definitions made in previous chapters, it is time to get our hands dirty.

Data analysis

Using a HIM approach, we are able to obtain all the data we need, however, we are only focusing on the fractals, since the fractals are the ones which provide information about the WIP. We then use 2 queries in MS-SQL to get the WIP information for tools and operations. We can save the information from the MS-SQL database into a csv file which we can then access from RStudio, or we can use RStudio to access the database by itself using the *RODBC* library, however, for the time being, we will use the former. Figure 3.5 for details.

```
Time;Operation;WIP
2017-07-01 00:00:00;A0301;1.00000
2017-07-01 00:00:00;A0302;0.85250
2017-07-01 00:00:00;A0503;0.30083
2017-07-01 00:00:00;A1303;0.60417
2017-07-01 00:00:00;A1304;0.63389
2017-07-01 00:00:00;A2101;0.54389
2017-07-01 00:00:00;A3150;0.44944
2017-07-01 00:00:00;A4106;1.00000
2017-07-01 00:00:00;A4202;2.00000
```

Figure 3.5. Fractal information for operation WIP.

Let's take a good look at the data from figure 3.5. This is the csv file from MS-SQL, we can then read this file in RStudio using the *read.csv* command (by default in RStudio). For each row, the data shows a timestamp, the name of an operation and the value of the WIP for that operation during one specific hour. It is very clear that this data is not good for forecasting as it is now, we need to shape it. Using RStudio and the library *reshape2*, we can transform this data into a proper table which we can work with. Figure 3.6 for details.

	Time	A0208	A0301	A0302	A0401	A0402
1	2017-07-01 00:00:00	NA	1.00000	0.85250	NA	NA
2	2017-07-01 01:00:00	0.04944	1.93750	1.64361	NA	NA
3	2017-07-01 02:00:00	1.00000	2.00000	2.00000	0.33694	0.78500
4	2017-07-01 03:00:00	1.00000	1.18917	2.00000	1.08139	1.00000
5	2017-07-01 04:00:00	0.21083	1.00000	2.75750	2.00000	0.19611

Figure 3.6. Data transformation using function *dcast.*

The data looks better, if not for those NA values that appear. These come from the operation at a specific time having nothing to work on, we can use a simple fix in RStudio which transforms these missing values into 0. Figure 3.7 for details.

	Time	A0208	A0301	A0302	A0401	A0402
1	2017-07-01 00:00:00	0.00000	1.00000	0.85250	0.00000	0.00000
2	2017-07-01 01:00:00	0.04944	1.93750	1.64361	0.00000	0.00000
3	2017-07-01 02:00:00	1.00000	2.00000	2.00000	0.33694	0.78500
4	2017-07-01 03:00:00	1.00000	1.18917	2.00000	1.08139	1.00000
5	2017-07-01 04:00:00	0.21083	1.00000	2.75750	2.00000	0.19611

Figure 3.7. No missing values.

Let us try and plot our data to see how it looks like.

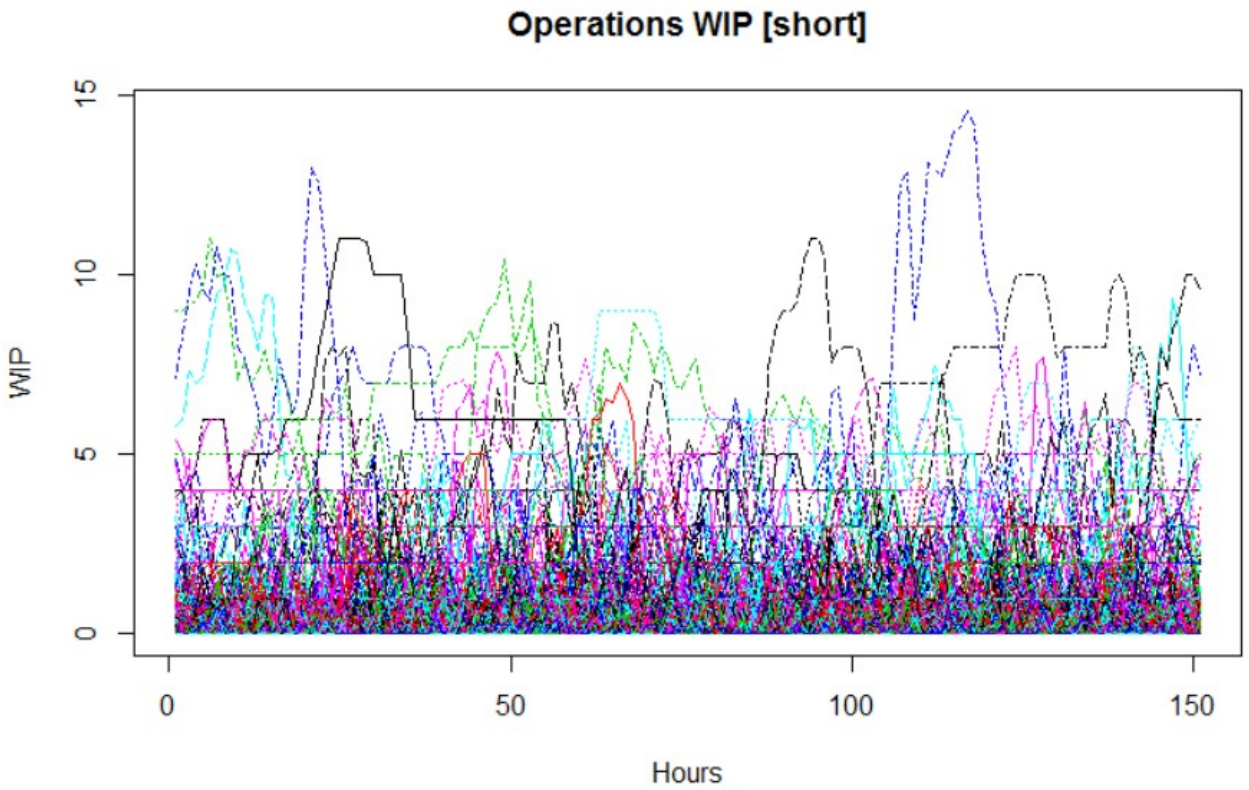


Figure 3.8. Plot of WIP for operations.

The plot looks messy, as is expected since we have 438 operations being plotted.

In this case, like in many others, we might find some duplicated information, we need to remove it because of two main reasons.

1. Duplicated data will affect the results of our data analysis, especially on linear models.

2. Whatever results we obtain for one of the tools/operations which are duplicated can be applied to the others.

One might ask, why do we get duplicated data in such a complex system? There exist situations where tools or operations are constantly working with the same lots, this leads to duplicated data.

Once duplicates are removed, we can start our data analysis. We loop through the tools and operations performing 2 functions on them, which we already know, a correlation and a Granger causal test.

We do this to try and associate which tools and operations might be correlated to some other tools and operations, also to try and find out which ones are the possible causals

for them as well. We then use RStudio to save this data as a csv file. Let's see an example from tools to find out how this would look. Figure 3.9 for details.

```
"ASM_A2","AUTO_CL_undot_1","LAM_4400_Poly_1","NF_1_D","RUD_AUTOEL_1"
0.61417,494140,140917,0,0.37222
0,471750,257555,0,0.30278
0,342083,186916,0,0.14945
0,173056,1e+05,0,0.65722
0,227999,0.27972,0,0.19889
0,299194,0,0,0.0375
0,415222,0.95917,0,0.17167
0,240805,0.73028,0,0.11694
```

Figure 3.9. Correlation file for tool ASM_A2.

Figure 3.9 gives an example of our data analysis, where we focus on one tool, namely ASM_A2 and find out which other tools are correlated to it. We do the same when it comes to Granger causals, in our case the result came out as the correlation. Note that this might not always be the case, as seen on the analysis of a different factory [XB-ELS].

Depending on the amount of data we have, this process can take sometimes even more than 10 hours to complete, but once it is done, we are ready to move to the forecasting part.

Data Forecast

We have completed our data correlation and causal tests, we now have, for each tool and each operation, two files one which contains the correlation results and one which contains the Granger causal results.

However we find ourselves at a cross-roads. In semiconductor manufacturing, just like in many other factories, it might happen that a tool or an operation works alone, often times these tools or operations are the first ones among the assembly line. This means that they depend on the behavior of themselves alone. As we have seen on stationary time series, ARIMA might be the best course of action for trying to forecast future values on them.

This is where things get a little bit messy, we now need to loop through each of the tools and operations, then we need to read their respective csv files and create a forecast for each of them (4 forecasts in total), we then need to select one of these forecasts as the most accurate one.

Let's break that down, we focus on tool ASM_A2, this tool has 2 files

- ASM_A2_correlationtools.csv
- ASM_A2_grangercausals.csv

We need to perform 2 forecasts on each of them, one using VAR and one using MARS.

However, whenever a tool or an operation has no causals and no other tool or operation associated to it by means of correlation, we need to apply an ARIMA forecast on it.

RStudio offers the use of scripts, where we can create some code and use it for later, and if it is well-defined, we can use it on pretty much any type of data. By creating such a script we can forecast on the data. Making the assumption that a tool or an operation does indeed have Granger causalsor some other tool or operation associated to it by correlation, we will start with a simple VAR forecast.

Forecast using VAR

VAR, as stated before, tries to find interdependencies between a set of time series (minimum of 2). Given that for the case of tool ASM_A2 we got the same file in both cases (correlation and Granger test), we will try and use this as our focus for now, but note that this works for any other cases. Also note that our files are created in a way that, the first column is the tool which we are focusing on when doing the correlation and Granger causal test, this will help when looping through all of them by making it easier to play with the data. Figure 3.10 for details on this concept.

	ASM_A2	AUTO_CL_undot_1	LAM_4400_Poly_1	NF_1_D	RUD_AUTOEL_1
1	6.14170e-01	4.941400e+05	1.40917e+05	0.00000	3.72220e-01
2	0.00000e+00	4.717500e+05	2.57555e+05	0.00000	3.02780e-01
3	0.00000e+00	3.420830e+05	1.86916e+05	0.00000	1.49450e-01
4	0.00000e+00	1.730560e+05	1.00000e+05	0.00000	6.57220e-01
5	0.00000e+00	2.279990e+05	2.79720e-01	0.00000	1.98890e-01

Dependent/response variable (y)

Independent variables (x1, x2...) Predictors

Figure 3.10. Linear model concept for our correlation and causal analysis.

Let's now look at the parameters required for VAR to work properly, as well as the key difference between it and MARS, which will be useful when working with MARS.

VAR takes a set of time series, a lag number and a type which defines the type of deterministic regressors to use, in our case, the type parameter will be set to “none”. The lag will be a number representing the amount of steps back that VAR will look into when finding the interdependencies between the set of time series, please note that the more regressors we use, the longer it will take to create a VAR model and the accuracy will vary.

Let us now look at a call for the VAR model using our ASM_A2 data. Figure 3.11 for details.

```
> ASM_VAR <- VAR(ASM_[1:1000,], p = 16, type = "none")
>
```

Figure 3.11. Call to VAR with 16 regressors.

Now let’s explain what happened. The first parameter is the data set from figure 3.10, however it is not the entire set, we arbitrarily select the first 1000 hours as a way of training the model, and since we expect to forecast the next 10 hours of WIP for the tool we are interested in (ASM_A2), we will be able to compare this forecast with the real data for the tool, we will explain this later when dealing with forecast accuracy.

Now that we have our VAR model, let’s call our handy predict function using this VAR model. Figure 3.12 for details.

```
> pred_ASM <- predict(ASM_VAR, n.ahead = 10, order = 0.95)
>
```

Figure 3.12. Predict the next 10 hours with a confidence interval of 95%.

The object *pred_ASM*contains our VAR forecast for the 5 variables used (response ASM_A2, and 4 predictors). Figure 3.13 shows how this forecast looks.

```
> pred_ASM
$ASM_A2
          fcst        lower    upper       CI
 [1,] 430824.7  262253.4318 599395.9 168571.2
 [2,] 409722.6  164117.1123 655328.0 245605.5
 [3,] 389772.7   96621.4978 682923.9 293151.2
 [4,] 327376.5    -822.8879 655575.9 328199.4
 [5,] 277617.4  -75301.4423 630536.2 352918.8
 [6,] 230852.8 -140618.8015 602324.5 371471.6
 [7,] 194025.1 -189295.5093 577345.8 383320.6
 [8,] 212756.4 -178965.2627 604478.1 391721.7
 [9,] 197154.9 -201538.5641 595848.3 398693.4
[10,] 180191.3 -224064.1639 584446.7 404255.4

$LAM_4400_Poly_1
            fcst     lower    upper       CI
 [1,]  16083.842 -167540.1 199707.8 183624.0
 [2,]  -3537.027 -263765.7 256691.6 260228.6
 [3,]  54360.469 -253511.5 362232.4 307871.9
 [4,]  89081.541 -250357.6 428520.7 339439.1
 [5,] 108687.445 -251471.6 468846.5 360159.1
 [6,] 116044.565 -259126.0 491215.2 375170.6
 [7,] 115899.399 -270698.6 502497.4 386598.0
 [8,] 107884.227 -287293.0 503061.5 395177.2
 [9,] 111763.564 -292693.5 516220.6 404457.1
[10,] 105812.146 -307740.0 519364.2 413552.1

$AUTO_CL_undot_1
          fcst      lower    upper       CI
 [1,] 320954.3   14761.51 627147.1 306192.8
 [2,] 353291.5  -36941.92 743524.9 390233.4
 [3,] 297334.2 -131605.19 726273.6 428939.4
 [4,] 322486.8 -133390.27 778363.9 455877.1
 [5,] 338351.9 -139722.20 816426.0 478074.1
 [6,] 340498.5 -152675.18 833672.3 493173.7
 [7,] 332539.6 -169568.80 834647.9 502108.4
 [8,] 297838.7 -214012.24 809689.7 511851.0
 [9,] 243194.8 -276851.89 763241.5 520046.7
[10,] 212878.4 -311849.85 737606.6 524728.2

$NF_1_D
            fcst     lower    upper       CI
 [1,]  1310.4031 -11361.06 13981.86 12671.46
 [2,]  2372.2910 -12179.87 16924.45 14552.16
 [3,]   405.4471 -14154.19 14965.08 14559.64
 [4,]  -156.4128 -14722.63 14409.81 14566.22
 [5,]  1632.1712 -12966.74 16231.08 14598.91
 [6,]  2372.4234 -12259.54 17004.39 14631.97
 [7,]  1230.9134 -13410.03 15871.86 14640.95
 [8,]  -351.9084 -15032.24 14328.43 14680.34
 [9,] -3123.6883 -17814.72 11567.34 14691.03
[10,] -1655.4317 -16358.98 13048.11 14703.54
```

Figure 3.13. Forecast of next 10 hours using VAR model.

This however, is not very reliable for all of these tools. The reason is that the predictors are the predictors only for ASM_A2, thus the ASM_A2 can be forecasted using the interdependencies obtained by its predictor tools, the same cannot be said about the predictors themselves, we can't be sure that the forecast on tool NF_1_D is reliable, since VAR tries to find interdependencies between the variables used and create a forecast, a forecast is done for the ASM_A2 using its actual predictors, meanwhile, tool NF_1_D might have a different set of predictors, thus the forecast for this tool could not be reliable.

Let us now compare the results of the forecast with the results of the actual observed data from the ASM_A2 tool. Figure 3.14 for a plot of the forecast using VAR.

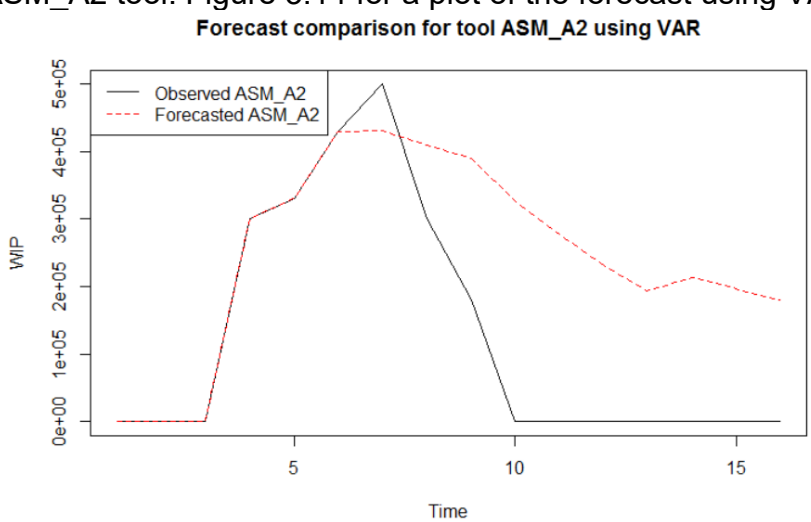


Figure 3.14. Forecast for tool ASM_A2 using VAR.

The forecast may not appear so good, so we need to measure how accurate this was, let us now introduce the RMSE (root mean squared error) measure for forecasting accuracy.

RMSE – (library caret)

The RMSE (aka. Root mean squared error or root mean squared deviation) measure is a value representing the sample standard deviation. When the predicted values are further apart from the observed values, the RMSE rises. Ideally we would have an RMSE of 0 [A-28] [A-29] [A-33]. In RSudio, the RMSE function takes as parameters the predicted values and the observations, it then tries to calculate the difference between observed and predicted in a residuals manner (as seen on the linear model). RMSE uses the following formula.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}.$$

Let's now use RMSE to measure the accuracy of our forecast. Figure 3.15 for RMSE call.

```
> RMSE(aux_asm[1001:1010,2], aux_asm[1001:1010,1])
[1] 212722.6
>
```

Figure 3.15. Call to RMSE function in RStudio, where the predicted data is on the second column of the aux_asm table, and the observed data is on the first column of the same table.

So this is a rather large number, recall that we said that ideally we would have a value of 0 for a perfect forecast, but in reality things are not always as pretty, however, let's remember another term we defined above, the standard deviation. *The certainty of uncertainty,* if we obtain the standard deviation of the observed data, and our forecast finds itself between these boundaries, we can actually call this an accurate forecast, note that we will be using 2 standard deviations. Figure 3.16 for a call of the standard deviation function.

```
> sd(aux_asm[1001:1010,1])
[1] 175171
>
```

Figure 3.16. Call for standard deviation on observed values.

Now our forecast does not look so bad, given that 1 standard deviation gives a value of 175171 units, and our RMSE gave a value of 212722.6, if we use 2 standard deviations (350342 standard deviation order 2) our forecast will find itself perfectly between the boundaries of the standard deviation of our focus tool, the ASM_A2. This was a good forecast indeed. Let's now move into the big one, MARS.

Forecast using MARS

The earth function (MARS) is the one we use to create our MARS models, this function takes a couple of parameters to work, it is also capable of creating a VAR model on its own. MARS will also force our friendly function *predict* to add a new parameter into its call. Let's first see what the earth function needs.

The earth (MARS) function requires a set of predictors, a response variable or a set of responses. This is what makes the MARS models different, they are capable of creating models for multiple response variables with minimum effort. However, for our purposes, we will try and focus on one tool at a time.

Let's see now how the call of the earth function looks like. Figure 3.17 for details.

```
> mars_ASM <- earth(x = ASM_[1:1000,-1], y = ASM_[1:1000,1])
>
```

Figure 3.17. Call to the earth function.

For the inexperienced RStudio reader, the call from figure 3.17 is read like this. "We call the earth model and assign the predictors as the first one thousand rows of data without the first column, recall that the first column is our focus tool. In the predictors set we add the same amount of rows but only the first column, namely our ASM_A2".

We can then move to the predict function, however, in order for MARS to know the future, it needs to know the future first. What does that mean? MARS needs to know the future value of the predictors in order to make the prediction on the response. Sketch figure 3.18 for details.

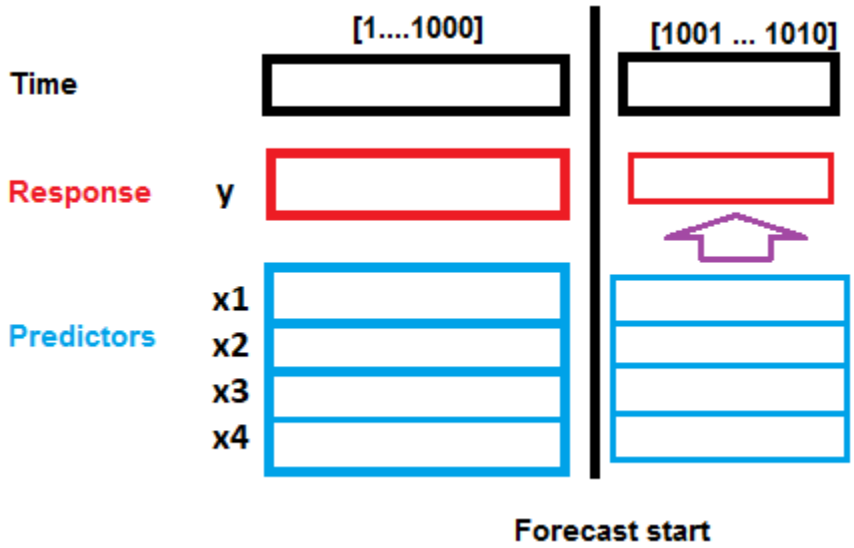


Figure 3.18. Know the future to make predictions.

Let's explain figure 3.18. We have a time frame of 1000 hours, we have a response variable[y] (red) and a set of predictors [x1,x2,x3,x4](light blue), we then start the forecast on the next 10 hours, however, in order to know the value of the response variable we first need to forecast the next 10 hours of the predictors. We can read the causal file of each of these predictors, we can create a VAR forecast on each of them (as seen above) or we can use ARIMA.

Depending on the situation, the forecast of the predictors could be done using ARIMA or VAR, if one of the predictors is a stationary time series, we will use ARIMA, if the predictor has some causals or correlated time series to it, we use VAR, for our case, each of the predictors has other time series assigned to them, thus VAR is the way to go.

Recall that we have already made a forecast for the predictors using VAR, but that forecast is not reliable since it was not made with the predictors of the predictors in mind. By looping through the set of predictors and their files, we can create 2 new forecasts using VAR, one for the correlation file of each predictor and one for the Granger causal test of each predictor. Using the RMSE measure we can decide which forecast is the most reliable. We save this new information in a table and use it to predict the response from our earth (MARS) model. Figure 3.19 for details on this concept.

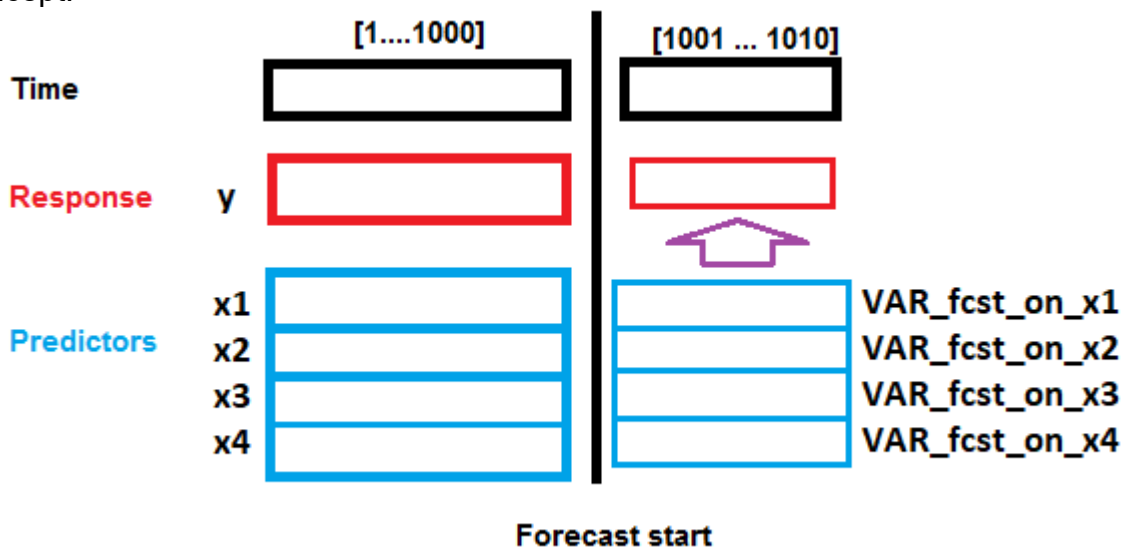


Figure 3.19. Predict on the predictors using VAR.

In order to make a prediction, we use our handy *predict*function which understands that we are using an earth object, we then add a new parameter called *newdata*, this parameter contains the future data of the predictors and will be used to forecast the responseFigure 3.20 for the call.

```
> pred_mars_ASM <- predict(mars_ASM, newdata = ASM_[1:1010,-1])
>
```

Figure 3.20. Call on predict function with newdata parameter.

Let's look at our results. Figure 3.21 for details.

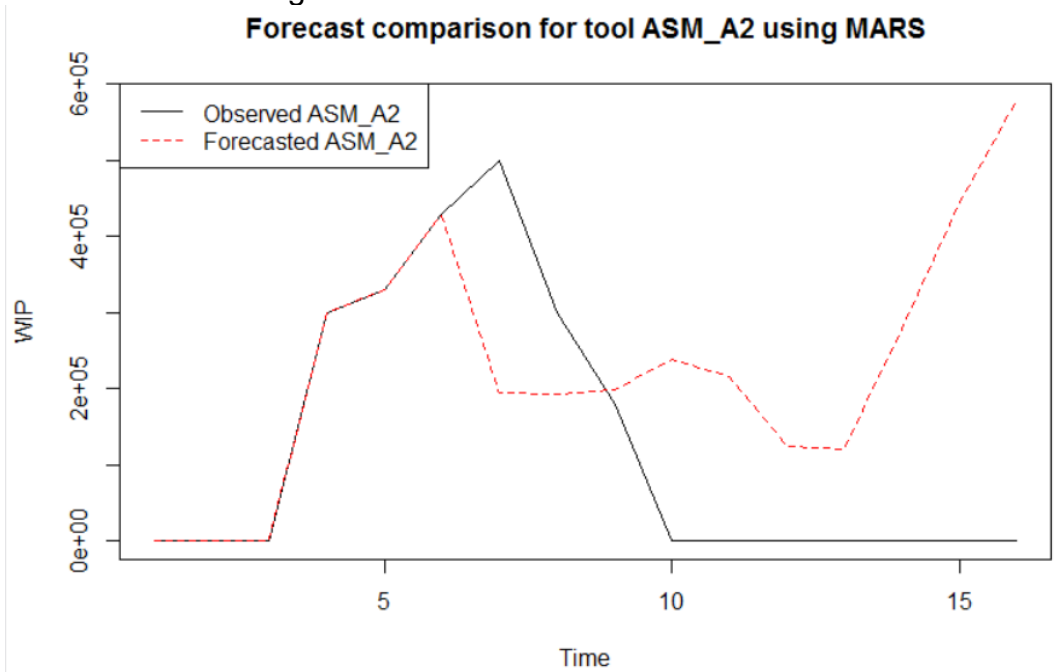


Figure 3.21. Forecast using MARS.

This forecast looks very different to the one obtained using VAR, let's see what RMSE thinks about it. Figure 3.22 for details.

```
> RMSE(pred_mars_ASM[1001:1010], aux_asm[1001:1010,1])
[1] 290979.3
>
```

Figure 3.22. RMSE of predicted values using MARS compared with observed data.

Looking back into our RMSE for the VAR forecast, the VAR forecast has a smaller RMSE value, which means it is a better forecast, please note that MARS was not too far off either, if we use 2 standard deviations the MARS forecast is also neatly placed between the standard deviation boundaries.

Let us now look at a more in depth comparison of these results. Figure 3.23 for details.

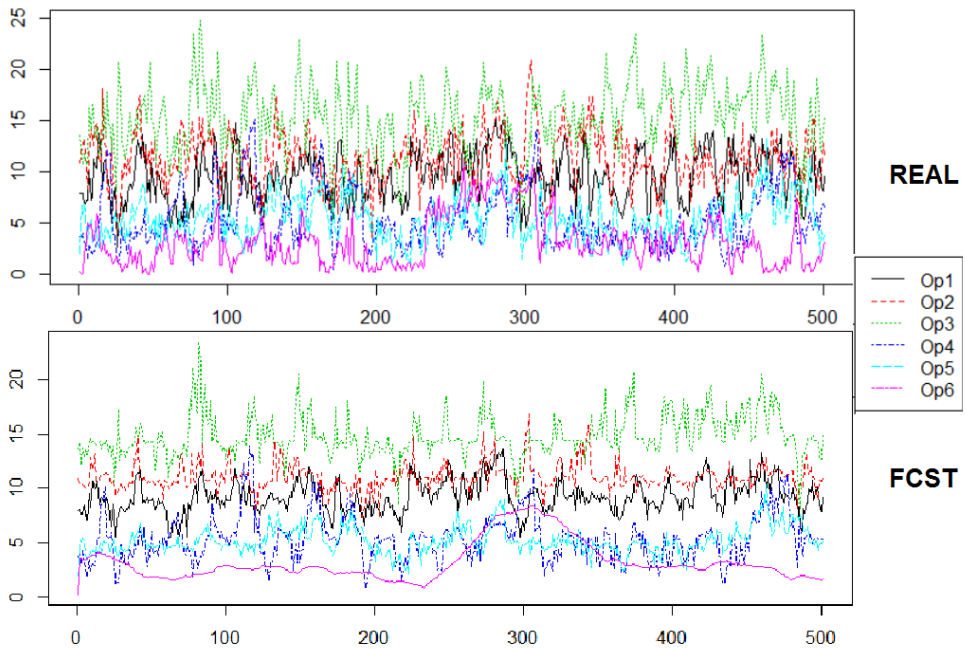


Figure 3.23. (Above) Real WIP for 500 hours. (Below) Forecasted WIP for 500 hours.

In figure 3.23 we find 6 operations, the top plot shows the real data while the bottom shows the forecast. 5 Of these operations have correlated operations or Granger causal relations with other operations, meanwhile, operation 6 is a stationary time series, thus the forecast for this operation is obtained using the ARIMA techniques explained above. The forecast here results in a smoother version of the real WIP, or rather a more average forecast, meaning that it would fit nicely between the standard deviation boundaries.The idea is that two forecasts are made on each operation, and the one which provides a smaller RMSE value is the one used for the plot as well as the forecast. This is not the case with operation 6, given that VAR and MARS both require at least one predictor in order to work, so ARIMA becomes the best solution for it.

The forecast is pretty accurate, aside from using RMSE, we can also make a correlation comparison of 2000 hours between the forecasted values with the observed values, we find that they are pretty similar. Recall that for a correlation test, two time series which are exactly the same would return a value of 1.

```
> cor(fcst_data[1500:3509,1], Op_1[1500:3509,])
[1] 0.7116812
> cor(fcst_data[1500:3509,2], Op_2[1500:3509,])
[1] 0.5842993
> cor(fcst_data[1500:3509,3], Op_3[1500:3509,])
[1] 0.7031266
> cor(fcst_data[1500:3509,4], Op_4[1500:3509,])
[1] 0.8603865
> cor(fcst_data[1500:3509,5], Op_5[1500:3509,])
[1] 0.7157824
> cor(fcst_data[1500:3509,6], Op_6[1500:3509,])
[1] 0.6288824
>
```

Figure 3.24. Comparison between real and forecasted using a correlation function.

In a perfect world, the values obtained by the correlation function would all be 1, implying a perfect match between the forecasted values and the observed values.

And thus, we have made a forecast on a tool and on a set of operations. It is important to note that not all tools and operations will have good results, some might have extremely good results while others will be average at best.

The following chapter contains the conclusions and future work.

Conclusions and future work

Let us look back into what we learned and summarize it all up.

We learned about the semiconductor industry, we learned how an integrated circuit is made, the steps required and the tools and operations (in general) required. We learned about some of the KPIs in semiconductor manufacturing while focusing mainly on the WIP, from then we moved into the industry 4.0 and its capabilities, then we learned a holistic model which gathers information from a specific subject and structures it into a neat format with atomics, fundamentals and fractals.

We then moved into the basic realm of statistics, where we learned about forecasting, linear models and their structure, some important analysis definitions like correlation and Granger causals together with some forecasting techniques, mainly ARIMA, VAR and MARS.

We then took an example from the Holistic Information Model (HIM) and applied our analysis knowledge on the data obtained from a semiconductor factory using an holistic approach. We took a tool and applied the techniques learned and saw that indeed, a forecasting model is possible, it is accurate and it requires (ideally) an holistic gathering of the data to work.

We can thus conclude this thesis with a new approach for **real-time** forecasting in semiconductor manufacturing, since the data is entered into the database as soon as events occur in the factory, the same data can be accessed by the developer or the program (if automated correctly) and the forecast for tools and operations is updated and keeps **evolving** together with the factory.

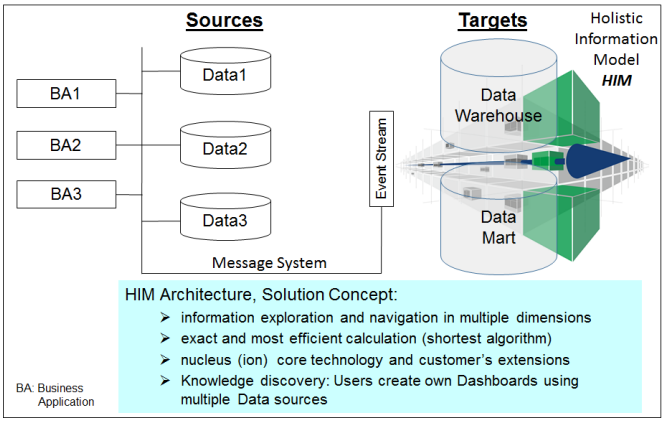
This system can go further away from the realm of semiconductors, since there exist many different factories in the world which use WIP as one of their KPIs, a similar system could be implemented into their ranks, allowing for an increased optimization of work and unlocking new paths for manufacturers to increase their profits.

Future work for this topic is still undecided. But the possibilities are practically endless.

Sources

[1] Luhn, G.: Concept paper: Real-time production information system as based on a holistic information model. Delivery in EU ECSEL Innovation Actions 692466-2, Power Semiconductor and Electronics Manufacturing 4.0, January 2017

Lehner, Wolfgang: Predictive Analytics – Opportunities for Factory Automation. Speech held at: 14thinnovationsforum for automation, January 19/20 2017, Dresden, Germany



Overview of a new approach (a holistic information concept), including continuously created information for forecasting (green boxes). The blue arrow indicates, that any aggregation of information is done by simple summations, which are executable in highest parallelism, and for this reason optimal distributable on standard multi-cpu hardware systems. The green boxes are representing continuously created and updated informations, covering forecasted future system states.

https://en.wikipedia.org/wiki/Big_data
http://www.mckinsey.com/business-functions/operations/our-insights/how-big-data-can-improve-manufacturing
http://www.mckinsey.com/industries/semiconductors/our-insights/optimizing-back-end-semiconductor-manufacturing-through-industry-40
https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf
http://www.britannica.com/EBchecked/topic/1116194/machine-learning
https://dzone.com/articles/reinforcement-learning
https://www.flexsim.com/factory-simulation

http://dataconomy.com/2015/04/predictive-maintenance-big-data-on-rails/
http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
Industry Forum Business Excellence Through Inspired People

Available:

https://www.industryforum.co.uk/wp-content/uploads/sites/6/2013/04/Planned-Maintenance.pdf

Jeff Banks The Applied Research Laboratory at The Pennsylvania State University

Available:
http://www.logisticsymposium.org/paperclip/speaker_management/14LA/presentation_file_distribution/358/8186d0e75243585e8e3bab96281ea9e1d0b2c646.pdf

https://www.lce.com/pdfs/The-PMPdM-Program-124.pdf
John D. Szwedo Baxter \& Woodman, Inc. Consulting Engineers

Available:
https://www.wwoa.org/files/publishedpapers/2012/Conference/Session\%20B1\%20Preventive,\%20Predictive,\%20and\%20Corrective\%20Maintenance.pdf

Jones Lang Lasalle

Available:http://www.pmmi.org/files/ms/certified/newsletters/preventivemaintenance.pdf

Jakob Krause for the Technical University of Dresden, Germany

Available:

http://www.qucosa.de/recherche/frontdoor/?tx_slubopus4frontend\%5bid\%5d=urn:nbn:de:bsz:14-qucosa-130908

http://research.microsoft.com/en-us/um/people/dmax/publications/dmart-final.pdf

https://www.academia.edu/12114216/Fast_Food_Forecasting
http://www.eyeon.nl/documenten/whitepapers/eyeon_wp_forecasting_planning_in_the_food_industry.pdf
https://en.wikipedia.org/wiki/Industry_4.0

https://goleansixsigma.com/lean-six-sigma-simulation/

[A-1] https://www.hindawi.com/journals/jam/2011/325690/
[A-1.1] http://slideplayer.com/slide/3715264/
[A-2] http://www.tel.com/ir/library/ar/document/ar2000_05.pdf
[A-2.1] https://www.accretech.jp/english/product/semicon/index.html
[A-2.2] https://en.wikipedia.org/wiki/Semiconductor_device_fabrication
[A-2.3] https://www.slideshare.net/ajal4u/silicon-manufacturing
[A-2.4] https://www.slideshare.net/avocado1111/lecture-8-43139462
[A-3] http://www.globalspec.com/learnmore/manufacturing_process_equipment/vacuum_equipment/thin_film_equipment/semiconductor_process_systems_cluster_tools
[A-3.1] https://www.cvdequipment.com/cvd-engineering/cluster-tools/
[Batch Ashin] http://www.pvateplaamerica.com/semi_stripping.php
[A-4] http://www.cmmmagazine.com/mems/single-wafer-vs-batch-wafer-processing-in-mems-manufacturing/
[A-4.1] https://www.researchgate.net/publication/266638896_Gate_Oxide_Cleans_on_Single_Wafer_Tool
[A-5] https://anomaly.io/seasonal-trend-decomposition-in-r/
[A-adult_ds] https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
[A-corr_ex] https://www.youtube.com/watch?v=igPQ-pI8Bjo
[A-Stephen] https://gist.github.com/stephenturner/3492773
[A-6] https://robjhyndman.com/hyndsight/longseasonality/
[A-7] https://www.youtube.com/watch?v=x4nO5yoarM0
[A-8] https://www.gcflearnfree.org/excel-tips/use-sumproduct-to-calculate-weighted-averages/1/
[A-9] https://explorable.com/null-hypothesis
[A-10] https://www.rdocumentation.org/packages/TSdist/versions/3.4/topics/CCorDistance
[A-11] https://www.r-bloggers.com/r-tutorial-series-simple-linear-regression/
[A-12] https://www.mathworks.com/discovery/linear-model.html
[A-13] *David A. Freedman (2009). Statistical Models: Theory and Practice. Cambridge University Press. p. 26.*
[A-14] http://onlinestatbook.com/2/regression/accuracy.html
[A-15] https://mathbitsnotebook.com/Algebra1/StatisticsReg/ST2Residuals.html
[A-16] http://dss.princeton.edu/online_help/analysis/interpreting_regression.htm
[A-17] http://blog.minitab.com/blog/statistics-and-quality-data-analysis/what-are-degrees-of-freedom-in-statistics
[A-18] http://blog.minitab.com/blog/adventures-in-statistics-2/multiple-regession-analysis-use-adjusted-r-squared-and-predicted-r-squared-to-include-the-correct-number-of-variables
[A-19] http://web.maths.unsw.edu.au/~adelle/Garvan/Assays/GoodnessOfFit.html
[A-20]http://www.statisticshowto.com/f-statistic/
[A-21] http://www.nlreg.com/results.htm
[A-22] https://stats.stackexchange.com/questions/9242/what-is-the-meaning-of-an-f-value-less-than-1-in-one-way-anova

[A-23] https://www.statsdirect.com/help/basics/p_values.htm
[A-24] https://www.openml.org/a/estimation-procedures/1
[A-25] https://www.cs.cmu.edu/~schneide/tut5/node42.html
[A-26] Generalized Cross Validation in variable selection with and without shrinkage Maarten Jansen Universit´eLibre de Bruxelles, departments of Mathematics and Computer Science December 9, 2015http://homepages.ulb.ac.be/~majansen/publications/jansen15gcvreprint.pdf
[A-27] http://www.murraylax.org/m230/spring2009/regvariation_print.pdf
[A-28] (2006). "Another look at measures of forecast accuracy". *International Journal of Forecasting*
[A-29] http://gisgeography.com/root-mean-square-error-rmse-gis/
[A-30] https://www.otexts.org/fpp/2/3
[A-31] https://tradingsim.com/blog/simple-moving-average/
[A-32] https://hlab.stanford.edu/brian/error_sum_of_squares.html
[A-33] http://www.analytictech.com/mb313/rootmean.htm
[A-34] http://informs-sim.org/wsc14papers/includes/files/212.pdf
[MRS_D] https://cran.r-project.org/web/packages/earth/earth.pdf
[IC-1] http://www.cpushack.com/WaferFull.html
[Chris-Roser] *Christoph Roser at AllAboutLean.com*
[RSTUDIO] https://www.rstudio.com/products/rstudio/download/
[St-Go] https://onlinecourses.science.psu.edu/stat501/node/358
[MBe-S] *An approach to predictive maintenance using SAP HANA*- Meryam Ben Ammar,Private Higher School of Engineering and Technology (ESPRIT), Tunis, Tunisia– 2017
[IBM-5nm] https://www.ibm.com/blogs/think/2017/06/5-nanometer-transistors/
[A-Sta] https://people.duke.edu/~rnau/411diff.htm
[A-Sta-2] https://www.otexts.org/fpp/8/1
[Ts-1]http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm
[Ts-2] https://en.wikipedia.org/wiki/Time_series
[Ts-3] http://www.investopedia.com/terms/t/timeseries.asp
[C-1] http://www.math.ku.dk/~sjo/papers/LisbonPaper.pdf
[SI-W] http://www.worldatlas.com/articles/all-about-the-semiconductor-industry.html
[MARS-1] https://statistics.stanford.edu/sites/default/files/LCS%20110.pdf
[MARS-2] https://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines