KU LEUVEN

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN

Statistische add-ins in Excel

Stijn Van Damme

R0294162

Maarten Dhondt

S0210398

Andries Daem

R0627563

Masterproef aangeboden tot het behalen van de graad

MASTER IN HET MANAGEMENT

Promotor: Prof. Dr. Martina Vandebroek

Academiejaar 2016-2017

KU LEUVEN

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN

Statistische add-ins in Excel

Deze thesis is een vervolg op de thesis “Statistical add-ins in Excel (NoruST)” geschreven door Frederik Van de Velde en Thomas Van Rompaey in 2015-2016. In de thesis van 2015-2016 werd de basis gelegd voor het ontwerp van een statistische add-in “NoruST”. Deze thesis beschrijft het verbeteren en optimaliseren van de bestaande add-in voor Excel. Daar waar de auteurs van de vorige thesis zich verdiepten in het uitleggen van de functies en de theorie ervan, wordt hier vooral gefocust op de installatie van het benodigde programma (Visual Studio) om aanpassingen aan de broncode aan te brengen. Deze installatiehandleiding vergemakkelijkt het onderhoud en of volgend research. Verder zullen de aanpassingen aan het bestaande programma en de nieuwe functies toegelicht worden. Vervolgens zal uitgebreid toegelicht worden hoe een nieuwe functie toegevoegd wordt en opgebouwd is aan de hand van een functie. Daarna wordt het gebruik van een functie uitgelegd. Tot slot wordt een besluit gevormd en aangegeven waar toekomstig onderzoek zich in kan verdiepen.

Stijn Van Damme

R0294162

Maarten Dhondt

S0210398

Andries Daem

R0627563

Masterproef aangeboden tot het behalen van de graad

MASTER IN HET MANAGEMENT

Promotor: Prof. Dr. Martina Vandebroek

Academiejaar 2016-2017

Dankwoord

Doorheen de looptijd van onze thesis hebben we erg veel steun gekregen van onze promotor Martina Vandebroek. Zij was steeds beschikbaar per e-mail en bereid om vaak met ons samen te zitten. Tevens heeft ze ons om ons tot aan de deadline effectief ondersteund zodat we een succesvolle thesis zouden kunnen afleveren.

Verder zouden we in het algemeen de mensen rondom ons willen bedanken die gedurende dit eindwerk hun steun hebben gegeven. Onze ouders en respectievelijke vriendinnen hebben elk op hun eigen manier ons geholpen om onze thesis tot een goed einde te brengen. Tijdens het ganse traject hebben ze ons de ruimte gegeven om aan dit project te kunnen werken.

Tot slot bedanken wij ook KULeuven die ons de mogelijkheid biedt om deze thesis af te leggen. De logistieke ondersteuning en de input die werd gegeven, faciliteerde in belangrijke mate het eindproduct van deze thesis.

Tot slot bedanken wij de staff voor de algemene organisatie van de richting ‘Master in het management’.

Inhoudstafel

Algemene Inleiding

Deze thesis is een vervolg op de thesis “Statistical add-ins in Excel (NoruST)” geschreven door Frederik Van de Velde en Thomas Van Rompaey in 2015-2016. Zij legden in de vorige thesis de basis aan de zelfontworpen add-in “NoruST”. Deze add-in heeft als bedoeling om statistische bewerkingen op data uit te voeren via voorgeprogrammeerde functies.

Daar waar vorig jaar de basis werd gelegd aan de add-in, focust deze thesis op het verbeteren en optimaliseren van deze add-in. Tevens gaat deze thesis minder diep in op de uitleg van definities van functies en van gebruikte programma’s aangezien dit reeds duidelijk gedocumenteerd is in de vorige thesis.

De thesis legt in hoofdstuk 1 eerst en vooral de focus op het installeren van ‘Visual Studio’ en de nodige bestanden. Door gebrek aan info in de vorige thesis was het bij het uittesten voor ons niet duidelijk welke stappen gevolgd moesten worden voor de installatie hiervan. De installatieprocedure zal het maken van aanpassingen aan de broncode voor bv. onderhoud of volgend research vergemakkelijken.

Hoofdstuk 2 sluit nauw aan bij het eerste hoofdstuk en legt het gebruik van de compiler ‘Visual Studio’ uit.

Vervolgens zal in hoofdstuk 3 een update gegeven worden omtrent de compatibiliteit met Mac gebruikers.

Aansluitend wordt in hoofdstuk 4 toegelicht wat de veranderingen zijn t.o.v. vorig jaar.

Nadien wordt in hoofdstuk 5 duidelijk hoe een nieuwe functie kan aangemaakt worden. Hier zal uitgebreid bij stilgestaan worden. Zo zal eerst het gehanteerde principe toegelicht worden. Vervolgens wordt er met behulp van screenshots en code uitgelegd hoe de opbouw van een functie in elkaar zit.

Nadien gaat hoofdstuk 6 nog dieper in op het gebruik van een functie.

Tot slot is er een besluit waarin er wordt gereflecteerd op het resultaat van de thesis. Tevens wordt hier toegelicht wat de focus kan zijn van een vervolg studie.

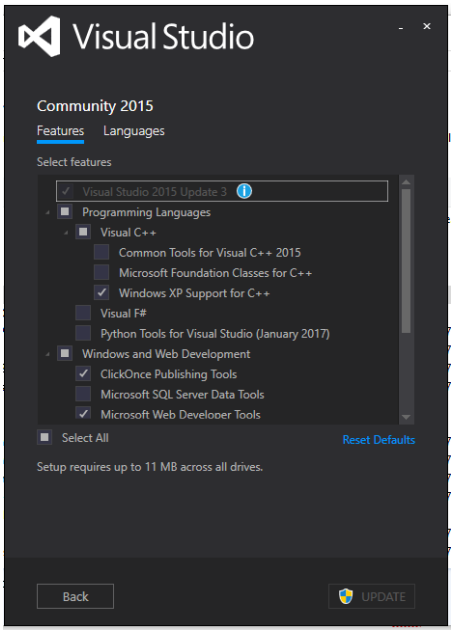
1 Installatie

Daar er nog geen uitgebreide handleiding bestond om de compiler en editor (Visual Studio) te installeren, zal hier in dit hoofdstuk uitgebreid aandacht aan worden besteed. Om aanpassingen, toevoegingen en onderhoud te verrichten aan de add-in moeten twee zaken geïnstalleerd worden. Eerst en vooral moet de compiler en editor namelijk Visual Studio geïnstalleerd worden. Daarnaast is het noodzakelijk om de VSTO (Visual Studio Tools for Office) plug-in te installeren. Deze plug-in maakt het mogelijk om plug-ins voor Office producten te ontwerpen.

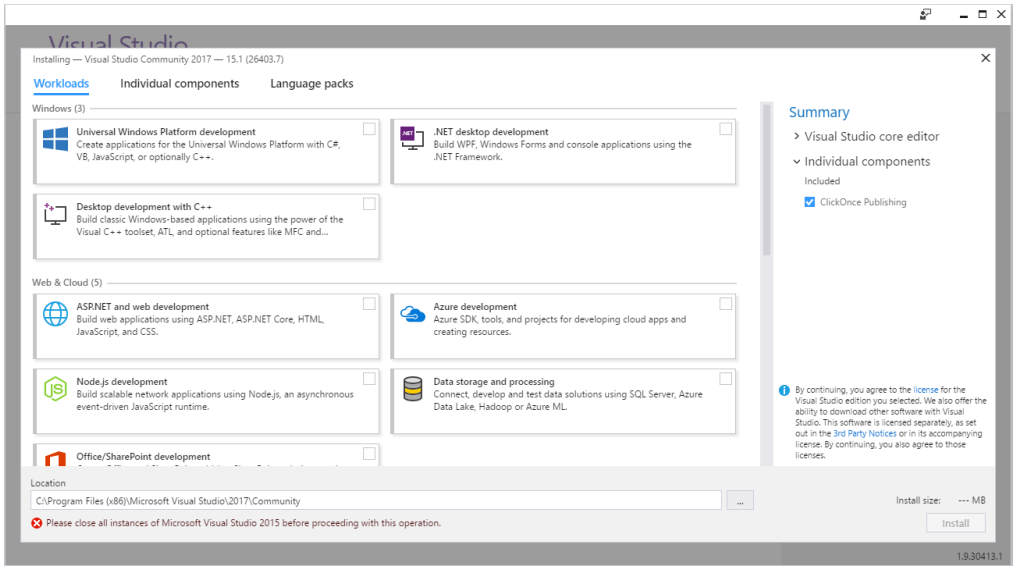
1.1 Visual Studio

Voor de mogelijkheden en de definitie van dit programma wordt er graag verwezen naar de thesis van 2015 – 2016. Deze worden daar zeer uitgebreid uitgelegd. Bijgevolg focust dit hoofdstuk zich vooral op de installatie. Op het moment van schrijven zijn Visual Studio 2015 en 2017 de twee meest actuele versies. Deze twee versies verschillen niet veel bij de installatie maar daar waar ze verschillen zal dit duidelijk toegelicht worden. Eens geïnstalleerd zijn de twee versies nagenoeg identiek in gebruik. Daar versie 2017 bij de start van deze thesis nog niet beschikbaar was, is hier vooral gebruik gemaakt van Visual Studio 2015. Tevens is dit de versie die in de thesis van 2015 – 2016 gebruikt werd. De gebruikte editie is de zogenaamde “Community edition” die gericht is op studenten, open source organisaties, kleine bedrijven, starters en individuele ontwikkelaars (zie bron Wikipedia). Bijgevolg is deze versie gratis te downloaden op de website van Microsoft. (Microsoft, 2017)

Na het downloaden van Visual Studio en het openen van de setup moet bij versie 2015 “custom” geselecteerd worden. Afhankelijk van de versie verschijnt vervolgens (Figuur 1.1) of (Figuur 1.2).



Figuur 1.1 Installatie scherm Visual Studio 2015



Figuur 1.2: Installatie scherm Visual Studio 2017

Dit scherm geeft de mogelijkheid om te selecteren welke onderdelen van Visual Studio geïnstalleerd moeten worden. Alle opties selecteren zou de computer overbodig belasten en niet gebruikte functies mee installeren. Vandaar dat enkel de noodzakelijke onderdelen mee geïnstalleerd zullen worden. Daar de installatie schermen voor beide versies verschillen zal er per versie uitleg verschaft worden.

1.1.1 Versie 2015

De optie “ClickOnce Publishing Tools” is noodzakelijk en moet steeds geselecteerd worden. Deze optie maakt het mogelijk om van de code een uitvoerbaar bestand .exe kan maken. De overige opties zijn optioneel. Indien er met GitHub gewerkt wordt (zie 1.3) moeten de opties “Git for Windows” en “GitHub extension for Visual Studio” mee geselecteerd worden. Hierover wordt in 1.3 meer uitleg over gegeven.

1.1.2 Versie 2017

Het installatie scherm van Visual Studio 2017 verschilt visueel sterk van versie 2015. In plaats van aparte opties worden er een aantal packs voorgesteld. Elk pack heeft reeds een aantal opties onder zich. Het “Universal Windows Platform development” pack is noodzakelijk en zal geselecteerd worden. Verder zijn er nog enkele individuele opties te selecteren. Daarvoor wordt het tabblad “Individual components” bovenaan geselecteerd. Zoals bij Versie 2015 is het belangrijk dat de optie “ClickOnce Publishing Tools geselecteerd wordt. Ten slotte kan in dit tabblad tevens de opties voor GitHub geselecteerd worden (zie Versie 2015).

1.2 VSTO plug-in

Om succesvol een add-in te programmeren voor een Office programma, in dit geval Excel, is het nodig om de VSTO plug-in of voluit geschreven “Visual Studio Tools for Office” tevens te installeren. Deze plug-in omvat alle benodigde bestanden om succesvol een add-in te programmeren. Voor uitgebreidere info wordt er opnieuw verwezen naar de thesis van 2015 - 2016. Deze plug-in is alsook gratis te downloaden via de website van Microsoft [1]. Na het downloaden van de setup verschijnt er een scherm waar “install” geselecteerd wordt.

1.3 GitHub

Aangezien er voor dit project met meerdere personen tegelijk is geprogrammeerd, werd er gebruik gemaakt van de tool “GitHub”. Deze tool maakt een efficiënte manier van samenwerken mogelijk. Eenvoudig gezegd werkt het door de broncode en bestanden van het programma in de “Cloud” te zetten. Na het aanmaken van een account op de website van GitHub, kan er een project aangemaakt worden. Nadien wordt in dit project een verzameling van de broncode en benodigde bestanden voor het schrijven van een programma geüpload. Dit project kan gedeeld worden met andere gebruikers van GitHub, zodat deze tevens wijzigingen aan het project kunnen aanbrengen. Het grote voordeel in deze tool zit hem in de manier waarop het aangebrachte wijzigingen in de code samenvoegt. Het bevat een slimme code die ervoor zorgt dat er tegelijk met meerdere personen wijzigingen kunnen aangebracht worden. Tevens wordt elke wijziging bijgehouden en zo is er tevens een back-up van de code en een overzicht van de gemaakte aanpassingen.

Het programma werkt tevens nauw samen met Visual Studio. Door de benodigde extensies te installeren (zie 1.1.1 & 1.1.2) is er een nagenoeg volledige integratie. De extensie van GitHub is na de installatie rechts boven gelokaliseerd in Visual Studio onder de “teamviewer”. De exacte installatie is vrij eenvoudig en bestaat uit het koppelen van een GitHub account en het selecteren van het juiste project. Er wordt steeds een lokale kopie van het project bewaard op je computer. De aanpassingen die er gemaakt worden aan het project worden bijgevolg ook niet meteen gesynchroniseerd evenals het ophalen van de recentste versie van het project. Dit gebeurt door “sync” te selecteren in de “Team Explorer” en nadien voor het ophalen van de laatste versie “pull”. Om zelf de gemaakte wijzigingen te synchroniseren,wordt de functie “push” gebruikt. Hier wordt tevens gevraagd om meer info te verschaffen wat de wijzingen juist inhouden. In 2.1.2 wordt hierover meer info verstrekt.

[1] https://aka.ms/GetLatestOfficeDevTools

2 Gebruik

Nu bovenstaande zaken uitgelegd en ondertussen geïnstalleerd zijn, is het tijd om de verdere werking van het programma uit te leggen.

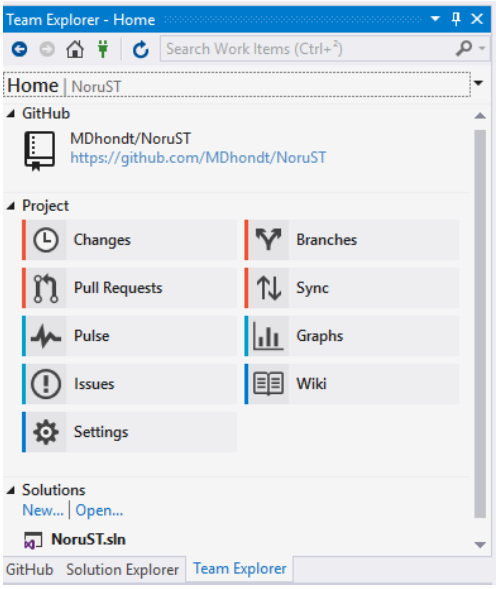
2.1 Project openen in Visual Studio

2.1.1 Zonder gebruik van GitHub

Om een project te openen, klikt men linksboven op het menu “File” vervolgens wordt “Open” geselecteerd en nadien op “Project/Solution…”. Ten slotte selecteer je in het dialoogvenster de locatie van het project.

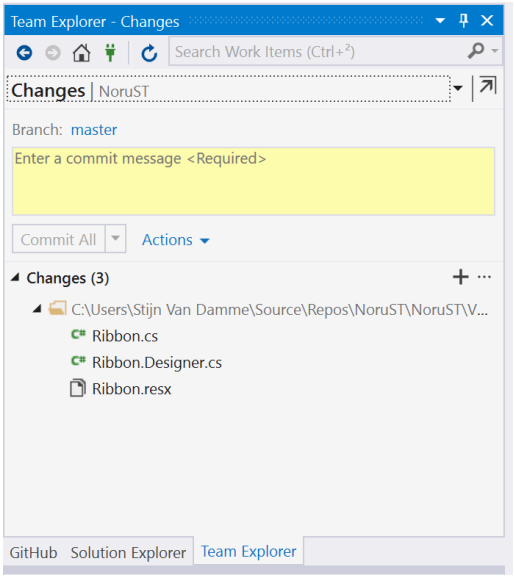
2.1.2 Met gebruik van GitHub

Na het installeren van de GitHub extensies, het koppelen van de GitHub account en project op de Computer, is er rechts op het scherm in Visual Studio de “Team Explorer” bijgekomen. De startpagina, zie hieronder Figuur 2.1, bestaat uit drie onderverdelingen “Github”, “Project” en “Solutions”.



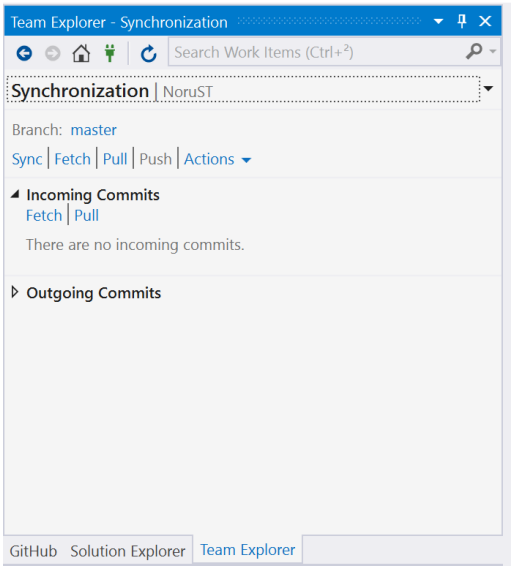
Figuur 2.1 Team Explorer

Bovenaan onder “GitHub” staat een korte verwijzing naar de eigenaar van het project en de directe URL naar de locatie van het project op de website van GitHub. Onder “Project” vindt je alle functies die betrekking hebben op het project zelf. Zo kan je bij “Changes” de gemaakte aanpassingen terug vinden die nog niet online staan (Figuur 2.2). Om deze aanpassingen te uploaden, kan er een “commit” uitgevoerd worden. Zonder uitleg of ‘commit message’ kan er echter geen “commit” uitgevoerd worden. In een ‘commit message’ wordt aangegeven wat er juist veranderd is.



Figuur 2.2 Github: Changes

Om de laatste versie op te halen, klikt men op “Sync” en vervolgens op “Pull” of “Sync”. Het laatste menu “Solutions” bevat het eigenlijke project. Door hier de “<projectnaam>.sln te selecteren, wordt het project geopend.



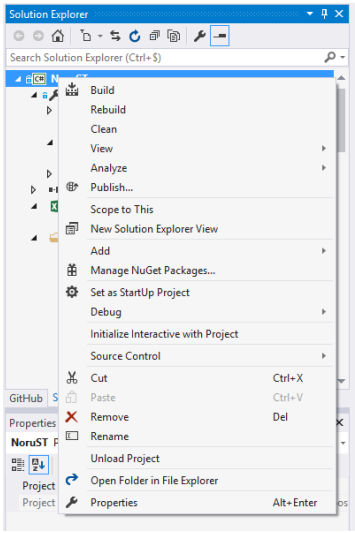
Figuur 2.3 Github: Synchronization

2.2 Testen/uitvoeren Project

Eenmaal het project open staat kunnen er aanpassingen gebeuren. Na de aanpassingen volgt er uiteraard het testen of uitvoeren van het programma.

2.2.1 Certificaat aanmaken/selecteren

Ongeacht of het programma getest of als afgewerkt project uitgevoerd wordt, is het noodzakelijk om een certificaat te selecteren. Op deze manier kan Microsoft zo controleren en registreren wie de maker is van het programma. Om zeker te zijn dat je een programma download of installeert zonder virussen, kan een certificaat aangevraagd worden van Microsoft. Zij geven enkel het certificaat aan een programma als het gecontroleerd is. Aangezien de add-in momenteel nog in de ontwikkelingsfase is, wordt er gebruik gemaakt van een tijdelijk certificaat. Deze zijn eenvoudig te genereren en maken het testen mogelijk. Een tijdelijk certificaat gebruikt voor het aanmaken ervan de gebruikersnaam die op dat moment actief is. Dit wil zeggen dat het enkel geldig is voor die gebruiker en bijgevolg heeft elke computer waarop het project wordt getest of uitgevoerd, een eigen certificaat nodig. Om dit certificaat aan te maken, klikt men op “Solution Explorer” rechts op het scherm. Vervolgens wordt er rechts op “NoruST” geklikt en “Properties” geselecteerd (zie Figuur 2.4).

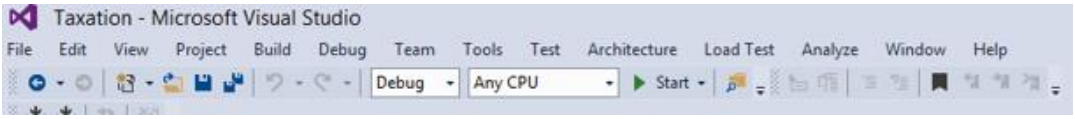


Figuur 2.4 Solution Explorer

In het bekomen scherm wordt links het tabblad “Signing” geselecteerd. Dit is het menu waar een certificaat aangemaakt kan worden of een bestaand certificaat geselecteerd kan worden. Een nieuw certificaat moet slechts één keer per computer aangemaakt worden. Nadien kan het eenvoudig geselecteerd worden. Voor het aanmaken wordt de knop “Create Test Certificate” gebruikt. Hier wordt dan automatisch een nieuw certificaat aangemaakt. In een pop-up vraagt het programma om een paswoord aan te maken en dit tweemaal in te voeren. Indien er reeds een certificaat aangemaakt is voor de computer, kan er met behulp van de knop “Select from File” het juiste certificaat geselecteerd worden.

2.2.2 Testen

Doorheen het programmeren van de add-in is het noodzakelijk om vaak het programma te testen. Hiervoor heeft ‘Visual Studio’ een handige knop ontwikkeld. Dit is het groene icoon “Start” bovenaan gelokaliseerd (Figuur 2.5). Door hier op te knikken zal het programma gecompileerd worden. Vervolgens zal een tijdelijk bestand gegenereerd worden. Dit bestand wordt automatisch geladen in Excel.



Figuur 2.5 Start knop

Als er tijdens het testen een fout in de add-in opduikt, zal ‘Visual Studio’ aanduiden op welke plaats hij vast is gelopen. Tevens geeft het programma gegevens weer over het processor gebruik enz. Om te stoppen met testen wordt de rode knop geselecteerd.

2.2.3 Uitvoeren

Deze stap is handig als het programma klaar is. Het programma is in staat om een setup bestand te genereren. Dit maakt het mogelijk om de add-in zonder problemen eenvoudig op eender welke computer te installeren. Dit gebeurt door bovenaan in de menu balk “Build” te selecteren. Vervolgens wordt ‘Publish NoruST’ geselecteerd. In de kader die dan openspringt wordt er gevraagd naar de locatie waar men de setup wil opslaan. Na het ingeven wordt op “Volgende” geklikt en ‘Select location from a CD-ROM or DVD ROM aangekruist. Nadien moet men terug “Next” selecteren en om af te sluiten “Finish”. Nu zullen er op de gespecifieerde locatie twee bestanden aangemaakt worden die noodzakelijk zijn voor de installatie van de add-in. Eerst staat er het bestand “Setup.exe” en het andere bestand is de folder ‘Application Files’ met de benodigde gegevens. Om de add-in te selecteren, kopieer je beide bestanden naar de doel computer en dubbelklik je op “Setup.exe”. De setup is vrij eenvoudig en zal niet verder toegelicht worden.

3 Mac

Dit onderwerp is vorig jaar uitvoerig besproken maar omdat er recent enkele nieuwigheden zijn uitgekomen wordt hier even wat uitleg bij gegeven.

3.1 Office 2016

Recent is Office 2016 for Mac uitgebracht. Deze update bracht verschillende nieuwigheden toe aan de Office programma's. Eén van deze nieuwigheden is de beperkte ondersteuning voor web-base add-ins. Vorig jaar is de beslissing genomen om de add-in te baseren op de COM technologie maar deze wordt helaas niet ondersteund door Mac. Ook is het niet eenvoudig om de add-in om te zetten van COM naar web-based. In de toekomst zou dit wel een oplossing kunnen bieden indien de add-inns geconformeerd worden naar web-based die wél compatibel zijn met Windows. Op deze manier probeert Microsoft de compatibiliteit tussen de twee besturingssystemen te verbeteren.

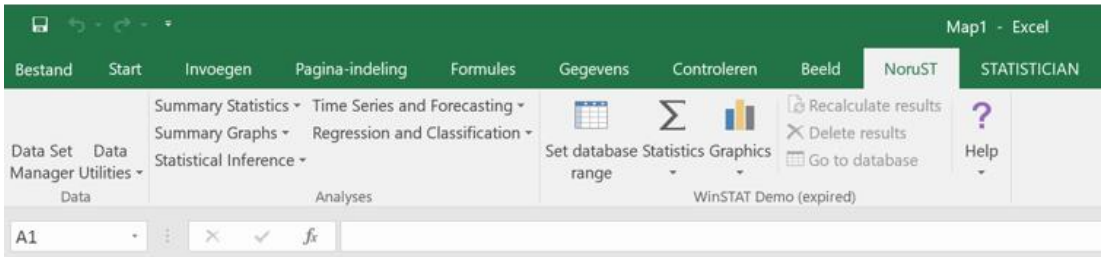
3.2 Visual Studio for mac

De tweede nieuwigheid op vlak van de compatibiliteit met Mac systemen, is het uitbrengen van Visual Studio for Mac. Recent heeft Microsoft een test versie gelanceerd om dit programma ook voor Mac gebruikers beschikbaar te maken. Het programma biedt wel wat minder functionaliteiten dan de versie voor Windows. De code waarin NoruST geschreven is met name C# wordt reeds ondersteund door dit programma. Desondanks is het programma eigenlijk geschikt voor het ontwerpen van apps voor de Cloud en voor mobiele besturingssystemen zoals iOS en Android. Kort samengevat wordt de kloof tussen de twee concurrenten kleiner maar is deze nog niet helemaal te overbruggen.

4 Aanpassingen

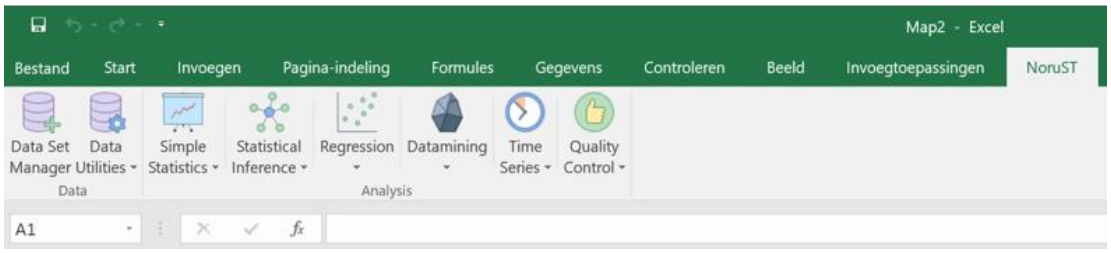
4.1 Ribbon

Eén van de eerste aanpassingen aan het bestaande programma had betrekking op de Ribbon. Dit is de balk bovenaan in Excel met menu's zoals "Bestand", "Start" en "Invoegen". In het begin verscheen de functie na installatie onder "Invoegtoepassingen". Het veranderde de naam van "Invoegtoepassingen" naar "NoruST". Dit zorgde voor verwarring bij het installeren van andere invoegtoepassingen. Deze kwamen dan bij onder het menu "NoruST". Figuur 4.1 illustreert dit probleem. Hier is het betalende programma "WinSTAT" tevens geïnstalleerd en zien we dat dit naast het programma van "NoruST" komt te staan. Om dit probleem op te lossen is er een aparte tab voor "NoruST" gemaakt. Op deze manier heeft het programma geen last van andere add-ins die mee geïnstalleerd worden.



Figuur 4.1 Ribbon met vorige versie NoruST

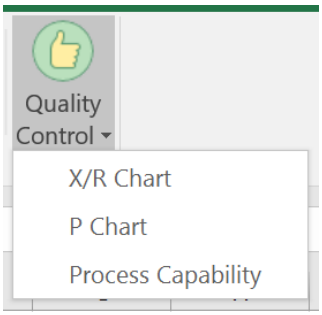
Het design van de Ribbon van de vorige versie van "NoruST" leunde sterk aan bij het bestaande "Stattools" programma. Doordat "NoruST" niet dezelfde functionaliteiten had/heeft als "Stattools", werd de plaats op de Ribbon niet efficiënt gebruikt. Bijgevolg was er na de installatie veel ruimte ongebruikt. Vandaar dat de menuknoppen groter zijn gemaakt. Om het programma wat aantrekkelijk te maken, zijn er tevens bijpassende icoontjes toegevoegd aan de menu's (Figuur 4.2). Nadien is er individueel gekeken of de groepering van de functies onder de menu's logisch was.



Figuur 4.2 Ribbon met nieuwe versie NoruST

4.2 Nieuwe functies

Het menu “Quality Control” (Figuur 4.3) is een geheel nieuw functie menu met drie volledig nieuwe functies. Het bevat onder meer het genereren van een “X/R chart” of “P Chart”. Tevens bevind de functie “Proces Capability” zich onder dit menu. Deze nieuwe functies worden uitgebreid toegelicht in Hoofdstuk 6.



Figuur 4.3 Menu Quality Control

4.3 Bestaande functies

Zoals in hoofdstuk 5 nog zal duidelijk worden, is er beslist om alle functies te her-programmeren volgens het MVP-principe (Model View Presenter) (zie 5.1.2). Hierdoor zijn alle functies dus herschreven zodat ze het MVP-principe volgen. Deze systematische opbouw zorgt voor duidelijk overzichtelijke functies. Kortom zijn alle bestaande functies volledig opnieuw ontworpen en is enkel de kern bewaard gebleven. Het opnieuw ontwerpen start bij het formulier en loopt door tot de broncode. Door hier de kern te bewaren (het eigenlijke berekenen) is de functionaliteit bewaard. Tevens werden gekende bugs reeds verwijderd. Sommige functies waren vorig jaar nog niet volledig compleet en deze zijn nu aangevuld. Kortom is er heel veel werk achter de schermen verricht.

5 Aanmaken nieuwe functies

In dit onderdeel wordt aan de hand van een bestaand codevoorbeeld uiteengezet hoe het aanmaken van nieuwe functies in de Ribbon verloopt. Dit is vanzelfsprekend niet de enige manier, maar het geeft wel een goed overzicht van de meest voorkomende functies, concepten en methodes die in dit project gebruikt werden. Bijgevolg kan dit onderdeel dienst doen als leidraad voor het aanmaken van nieuwe functies of verbeteren/aanpassen van bestaande functies. De functies die tijdens het programmeren grote struikelblokken vormden, zullen meer in detail beschreven worden. Vooraleer het aanmaken van een nieuwe functie wordt uitgelegd, zal eerst het gebruikte principe uitgelegd worden in 5.1.

In dit voorbeeld bespreekt men de opbouw van de functie ‘Runs Test For Randomness’ die onder de rubriek ‘Time Series’ valt. Dit is een relatief eenvoudige functie die gebruik maakt van enkele belangrijke concepten. Het aanmaken van grafieken komt in deze functie niet voor maar dit wordt later in 5.6 verder uitgelegd aan de hand van een andere functie.

5.1 Model View Presenter

Doorheen dit project maken we gebruik van de Model – View – Presenter werkwijze. Dit is een principe dat het toelaat om overzichtelijk verschillende stukken code op te splitsen naargelang hun functionaliteit.

5.1.1 Graphical User Interface (GUI)

De code mag nog zo robuust, performant, of elegant zijn; de eindgebruikers beoordelen de kwaliteit van een programma vooral op de gebruiksvriendelijkheid ervan. Dit houdt vooral in dat het programma een intuïtieve manier van werken aanbiedt. Daarom is doorheen de verschillende functionaliteiten die geïmplementeerd zijn, steeds gepoogd om hier zo veel mogelijk rekening met te houden.

Ondanks het feit dat zo’n grafische userinterface eigenlijk niet de kern van het programma uitmaakt, is het wel zo dat het de brug vormt tussen de gebruikers en de eigenlijke functionaliteit van het programma. Het is dus het deel dat het meest gebruikt zal worden en waarbij het een uitdaging is om alle interacties die een gebruiker wil uitvoeren mogelijk te maken. Tegelijkertijd biedt het een bescherming aan de code voor ongeldige invoer.

Het is dus – net zoals bij de meeste softwarecomponenten – erg belangrijk om ook dit deel op een gestructureerde manier te coderen. Hierbij is het hanteren van het encapsulatie principe belangrijk. Er is een reden dat object georiënteerd programmeren dit principe hoog in het vaandel draagt.

Encapsulatie streeft ernaar om de verschillende delen van een programma zo onafhankelijk mogelijk van elkaar te laten werken. Concreet wil dit zeggen dat als er iets wijzigt aan de interface (bv. een extra veld wordt toegevoegd en/of een ander veld wordt verwijderd) de bestaande code hier zo min mogelijk hinder van zou mogen ondervinden.

Doorheen de jaren zijn er heel wat verschillende methodologieën[2] gebruikt om de encapsulatie van software modules te bevorderen. Om deze segregatie van functionele code en grafische gebruikersinterface code te bekomen hebben wij gekozen voor het in softwareontwikkeling frequent gebruikte Model-View-Presenter patroon.

5.1.2 Model-View-Presenter (MVP)

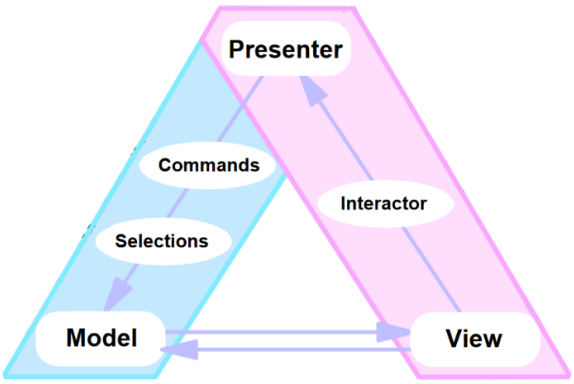
In het MVP-patroon wordt er voor quasi elk gebruikersscherm een Model-klasse, een View-klasse en een Presenter-klasse aangemaakt met elk hun eigen verantwoordelijkheid.

De terminologie Model, View, en Presenter komen uiteraard uit de wereld der softwareontwikkeling. De term Model mag hier dan ook niet verward worden met een econometrisch, statistisch, wiskundig, … model. In theorie mag er in de code een andere terminologie gehanteerd worden. Dit heeft geen enkele invloed op de ontwikkelde code. We zouden dus het Model in MVP bijvoorbeeld kunnen hernoemen door Data. Dit zou in de context van statistische software een nauwer bij de functionaliteit aansluitende naamgeving zijn.

MVP is echter een wijdverspreid patroon en het is gemakkelijker om dit qua terminologie zo goed mogelijk te volgen. Daarom is er gekozen om de termen Model en Presenter niet te wijzigen. View is wel gewijzigd naar Form omdat dit in de context van een .NET applicatie nu eenmaal de gebruikte terminologie is.

Als we de functie “Sample Size Estimation” als voorbeeld nemen, betekent dit dat er drie klassen bestaan namelijk: “SampleSizeEstimation**Model**”, “SampleSizeEstimation**Form**” en “SampleSize-Estimation**Presenter**”.

In een MVP-patroon heeft elke component zijn eigen functie.



Figuur 5.1 MVP-Patroon

[2] https://martinfowler.com/eaaDev/uiArchs.html

5.1.2.1 Model

De functie van het model is om alle data die nodig zijn om een bepaalde functie uit te voeren, bij te houden.

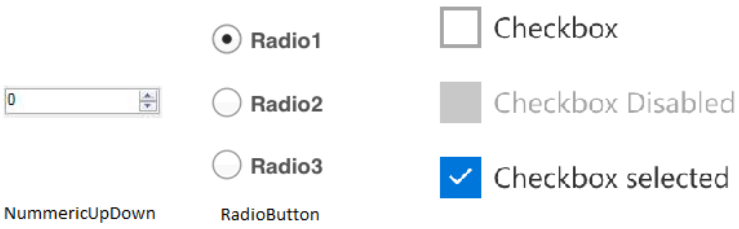
In het geval van het “SampleSizeEstimation**Model**” ziet dit eruit als

```
public class SampleSizeEstimationModel {
        public bool mean { get; set; }
        public bool proportion { get; set; }
        public bool diffMean { get; set; }
        public bool diffProportion { get; set; }
        public int confidenceLevel { get; set; }
        public string marginOfError { get; set; }
        public string estimation1 { get; set; }
        public string estimation2 { get; set; }
}
```

De variabelen mean, proportion, diffMean en diffProportion van het type bool geven dan bijvoorbeeld aan welke parameter er dient geschat te worden. In de Presenter worden deze data dan uitgelezen en verwerkt.

5.1.2.2 View

De functie van de view voorziet de gebruiker van een scherm met componenten waarmee deze kan interageren. Denk maar aan een “TextBox” om tekst in te voeren, een “NumericUpDown” om met pijltjes een geheel getal te selecteren (Figuur 5.2), een “CheckBox” die een vakje voorstelt dat kan aangevinkt worden of “RadioButtons” die bolletjes voorstellen waarvan er maar exact één tegelijkertijd geselecteerd kan zijn.



Figuur 5.2 Componenten

Deze componenten hebben allemaal hun basisfunctionaliteit. Een “CheckBox” bijvoorbeeld zal een hokje aanvinken als erop geklikt wordt en er nog geen vinkje stond. Wanneer het hokje al aangevinkt was, kan dit verdwijnen door nogmaals op het hokje te klikken. Zo’n gedrag is inherent aan een gebruikerscomponent en dient niet verder gedefinieerd te worden.

Als we echter iets complexer en meer gepersonaliseerd gedrag willen, zal dit zelf moeten gecodeerd worden. Een voorbeeld hiervan kan geïllustreerd worden door de functie “SampleSizeEstimationForm” (Code 5.1).

```
private void estimateCheckedChanged(object sender, System.EventArgs e) {
     nudConfidenceLevel.Value = 95;
     txtMarginOfError.Text = "0.1";

     if (rdbMean.Checked)
     {
         lblEstimate1.Text = "Estimated Standard Deviation";
         txtEstimate1.Text = "1";
         lblEstimate2.Visible = false;
         txtEstimate2.Visible = false;
     }
     if (rdbProportion.Checked)
     {
         lblEstimate1.Text = "Estimated Proportion";
         txtEstimate1.Text = "0.1";
         lblEstimate2.Visible = false;
         txtEstimate2.Visible = false;
     }
     if (rdbDifferenceOfMeans.Checked)
     {
         lblEstimate1.Text = "Estimated Common Standard Deviation";
         txtEstimate1.Text = "1";
         lblEstimate2.Visible = false;
         txtEstimate2.Visible = false;
     }
     if (rdbDifferenceOfProportions.Checked)
     {
         lblEstimate1.Text = "Estimated Proportion 1";
         txtEstimate1.Text = "0.1";
         lblEstimate2.Visible = true;
         txtEstimate2.Visible = true;
         txtEstimate2.Text = "0.1";
     }
}
```

Code 5.1 RadioButton

Deze code wordt uitgevoerd wanneer er op een “RadioButton” geklikt wordt. Het bepaalt welke tekst er te zien moet zijn afhankelijk van de geselecteerde “RadioButton”. Tevens maakt het sommige velden (on)zichtbaar al naargelang de geselecteerde radiobutton.

De pijlen in Figuur 5.1 tussen model en view duiden erop dat het model moet veranderen wanneer de view verandert en omgekeerd.

5.1.2.3 Presenter

De presenter is waar het eigenlijke werk dient te gebeuren. Wanneer een gebruiker op de knop drukt waarmee hij het resultaat wil zien, meestal ‘OK’-knop, is het aan de viewer om de controle af te staan aan de presenter. In “SampleSizeEstimationForm” ziet dit er als volgt uit (Code 5.2).

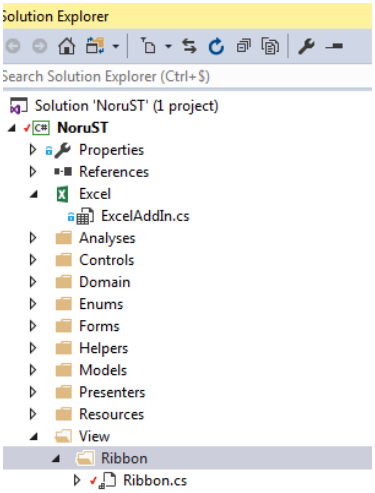
```
private void btnOk_Click(object sender, System.EventArgs e) {
     presenter.estimateSampleSize();
     Close();
}
```

Code 5.2 OK-Knop

De functie beschrijft de acties die ondernomen worden wanneer de ‘OK’ knop wordt gebruikt. In Code 5.2 vertelt deze functie dat de presenter zijn berekeningen moet doen. Dit wordt gevolgd door het Close() commando om het scherm van de view af te sluiten. Zo zal de gebruiker terug worden gestuurd naar Excel waar hij de uitvoering zal zien van wat de presenter zojuist berekend heeft.

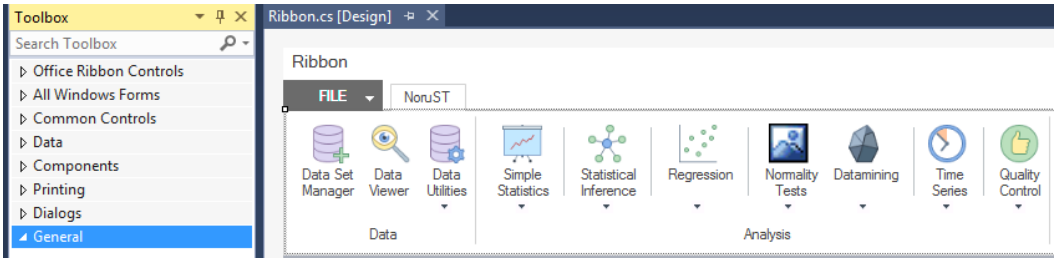
5.2 Ribbon

Nu er wat uitleg is gegeven omtrent het gehanteerde principe is het tijd om het eigenlijke aanmaken verder uit te leggen. Een nieuwe functie aanmaken begint bij het toevoegen van een knop aan de Ribbon (zie 4.1). Navigeer hiervoor doorheen de “Solution Explorer” naar het tabblad “View”, vouw de map open en dubbelklik op “Ribbon.cs”.



Figuur 5.3 Ribbon.cs in Solution Explorer

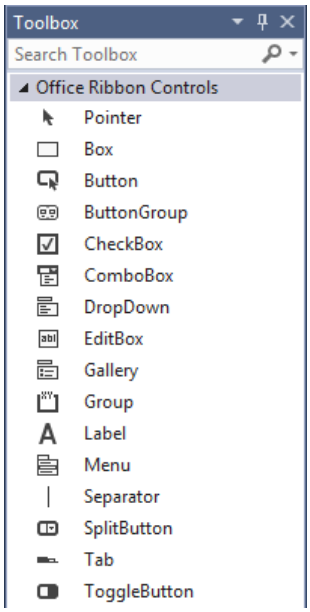
Dit opent de designer van de Ribbon. Deze geeft weer hoe de Ribbon er momenteel uitziet. Er zijn twee groepen aangemaakt (Data en Analysis). In de groep Data zijn er twee knoppen aangemaakt “Data Set Manager” en “Data Viewer” en één menu “Data Utilities”. Onder Analysis zijn er zes menu’s aangemaakt. Wanneer men op het pijltje onder een menu klikt, ziet men de knoppen die onder dit menu aangemaakt zijn bv. onder het menu Time Series zijn de knoppen Time Series Graph, Runs test for randomness en Forecast aangemaakt.



Figuur 5.4 Ribbon designer

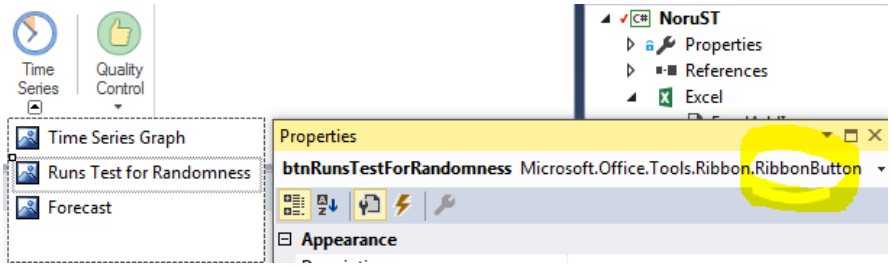
De items zoals ‘Menu, Group en Button’ waaruit het lint bestaat, zijn terug te vinden in het menu “Toolbox” onder het submenu ‘Office Ribbon Controls’ (Figuur 5.5). Hieronder staan alle mogelijke items die toegevoegd kunnen worden aan de Ribbon.

Zo kan men bv. een groep toevoegen of een button toevoegen aan de bestaande groepen of menu's. Indien het menu "Toolbox" nog niet zichtbaar is, kan dit in Visual Studio gewijzigd worden. Dit gebeurd bovenaan in Visual Studio onder het menu "View" en vervolgens "Toolbox" of door de shortcut Ctrl-Alt-X te gebruiken.



Figuur 5.5 Toolbox, Office Ribbon Controls

In volgend voorbeeld voegt men de knop 'Runs Test for Randomness' toe onder de "Group" 'Time Series'. Het toevoegen van een item uit de 'Office Ribbon Controls', gebeurt altijd door het principe van 'drag and drop" of vastnemen en verslepen naar de gewenste plaats. Indien er twijfel is over het type control dat toegevoegd moet worden, kan er gespiekt worden bij bestaande items in de Ribbon (Figuur 5.6). Door rechts te klikken op een gelijkaardig item in de Ribbon, vervolgens op "properties" kan het type vaak achterhaald worden. Vaak is het echter sneller om bij twijfel het item uit de 'Office Ribbon Controls' te verslepen naar de Ribbon om te kijken hoe het er uit ziet. Zo kan een groep (zoals de groep Analysis) toegevoegd worden door vanuit de 'Office Ribbon Controls' het element 'Group' te slepen naar de Ribbon en los te laten op de plaats waar de groep moet komen. Wanneer een groep aangemaakt is, kan je hierin knoppen toevoegen. In het voorbeeld zal dit de knop 'Time series' zijn. Dit gebeurd door het element "Menu" naar de Ribbon te slepen en los te laten in de "Group". Vervolgens kan er onder dit "Menu" nog een knop toegevoegd worden door op het kleine pijltje onder 'Time Series' te klikken. Vervolgens wordt het element "Button" versleept naar het verschenen menu onder 'Time Series'. Wanneer men een nieuw element naar de Ribbon gesleept heeft, is het mogelijk om de lay-out, afbeeldingen, grootte en andere eigenschappen van deze control aan te passen door rechts te klikken op de control en op "Properties" te klikken.



Figuur 5.6 Type detecteren

De Designer genereert automatisch bij bv. het toevoegen van elementen, verslepen etc. lay-out code. Deze code staat in de 'Solution Explorer' onder 'Ribbon' en vervolgens in het bestand "Ribbon.Designer.cs". Dit bestand dient enkel aangepast te worden indien men ervoor kiest om de lay-out te wijzigen via code. Normaal als men zoals hierboven de Designer gebruikt, hoeft men niets aan de "Ribbon.Designer.cs" te wijzigen.

De volgende stap is het toekennen van een functionaliteit aan de toegevoegde button. Dit doet men door op een willekeurige plaats op de Ribbon in de Designer rechts te klikken en vervolgens 'View Code' te selecteren. Nu opent zich de code in de "code-editor". De code hier is reeds automatisch gegenereerd op basis van de toegevoegde elementen. Om de nieuw toegevoegde button te activeren, moeten hier enkele lijnen code aan toegevoegd worden. Deze zijn in het geel gemarkeerd in Code 5.3.

```
using Microsoft.Office.Tools.Ribbon;
using NoruST.Forms;
using NoruST.Presenters;

namespace NoruST
{
   public partial class Ribbon
   {
      // Hier wordt een variabele van het type RunsTestForRandomnessPresenter aangemaakt met de naam runsTestForRandomnessPresenter
…
      private OneVariableSummaryPresenter oneVariableSummaryPresenter;
      private RunsTestForRandomnessPresenter runsTestForRandomnessPresenter;
      private ForecastPresenter forecastPresenter;
…
      private void Ribbon_Load(object sender, RibbonUIEventArgs e)
      {
         // Hier wordt de variabele geïnitialiseerd en gekoppeld met een object van de klasse dataSetManagerPresenter
         runsTestForRandomnessPresenter = new RunsTestForRandomnessPresenter(dataSetManagerPresenter);
…
         // Een klikactie op de button 'btnRunsTestForRandomness' die op de ribbon toegevoegd is zal de methode openView() van de bijhorende presenter aanroepen. Deze methode opent de form van de runs test
…
         btnAnova.Click += delegate { oneWayAnovaPresenter.openView(); };
         btnRunsTestForRandomness.Click += delegate { runsTestForRandomnessPresenter.openView(); };
…
      }
   }
}
```

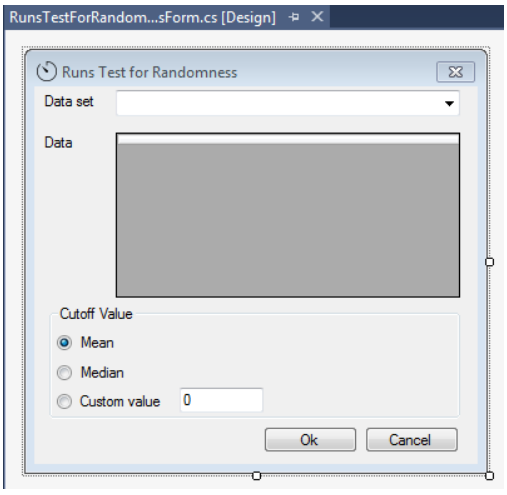
Code 5.3 View code

Aangezien er nu nog geen form, model of presenter aangemaakt is, zal deze code nog niet kunnen compileren (er zal een foutmelding komen en code zal rood worden gemaakt). Wanneer alle stappen van de opbouw van deze functie doorlopen zijn, zouden alle functies herkenbaar en correct moeten zijn voor de compiler.

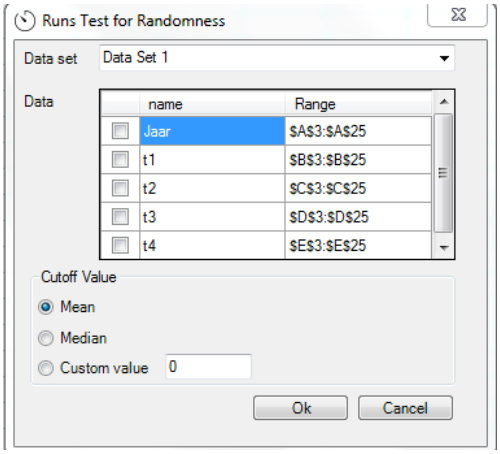
5.3 Form

Zoals in paragraaf 5.1.2 besproken, wordt de interactie met de gebruiker voornamelijk afgehandeld in de “View”. Om het principe te verduidelijken, gaan we in deze implementatie “View” met “Form” gelijkstellen. Het “Form” komt immers overeen met het scherm dat de gebruiker te zien krijgt wanneer hij een bepaalde functie wilt uitvoeren. In dit project zijn alle “Forms” of formulieren verzameld onder de “Solution Explorer” en vervolgens “Forms”. In Visual Studio is het mogelijk om formulieren op te bouwen via een designer. Deze werkt op dezelfde manier als de designer van de Ribbon. Er kan eenvoudig een nieuwe “Form” gemaakt worden door rechts te klikken op “Forms” in de “Solution Explorer”. Vervolgens op “Add” en dan op ‘Windows Form’. Daarna in de pop-up checken of ‘Windows Form’ als type geselecteerd is en daarna op “Add”. Het nieuwe formulier zal te vinden zijn opnieuw onder “Forms” in de ‘Solution explorer’. Er zal echter meestal voor een nieuwe functie gezocht worden naar een gelijkaardige functie die ongeveer hetzelfde formulier heeft als hetgeen nodig is. Nadien kan dit formulier dan nog m.b.v. de “Toolbox” aangepast worden en gepersonaliseerd. Na het zoeken naar een form die het meest gelijkaardig is aan de form die men wilt toevoegen, zal men deze kopiëren en plakken binnen de “Solution Explorer”. Dit doet men door rechts te klikken op het gelijkaardige form in de ‘Solution Explorer’ en te klikken op ‘copy’. Daarna klikt men rechts op de ‘Forms’-folder en kiest men voor ‘Paste’. Nu kan men het gekopieerde formulier hernoemen naar de nieuwe form.

Figuur 5.7 toont het formulier voor de runs test in de designer van Visual Studio. Figuur 5.8 toont hetzelfde formulier als het opgeroepen wordt in Excel en data sets geladen zijn.

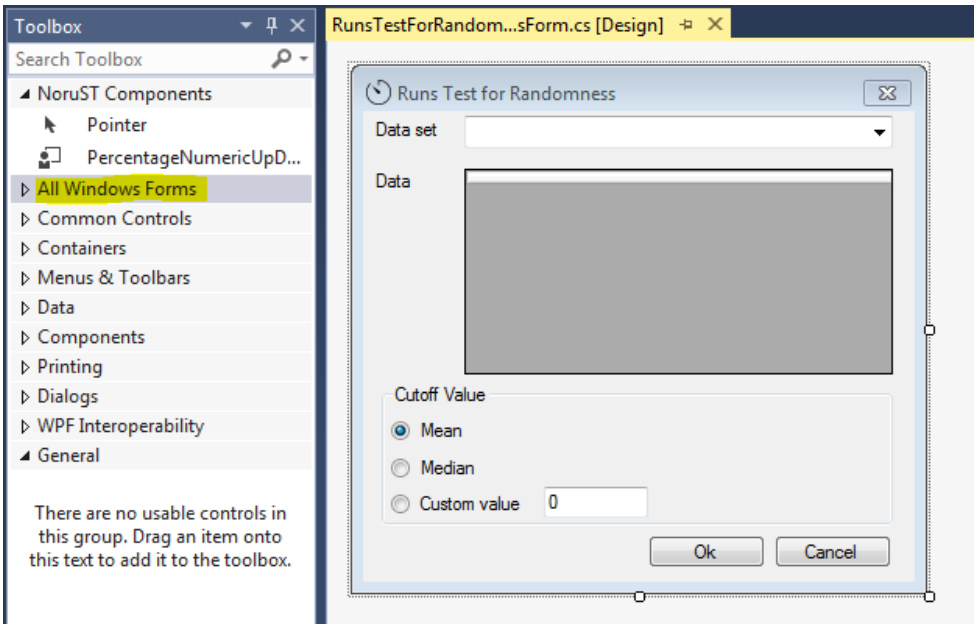


Figuur 5.7 Runs Test in Visual Studio



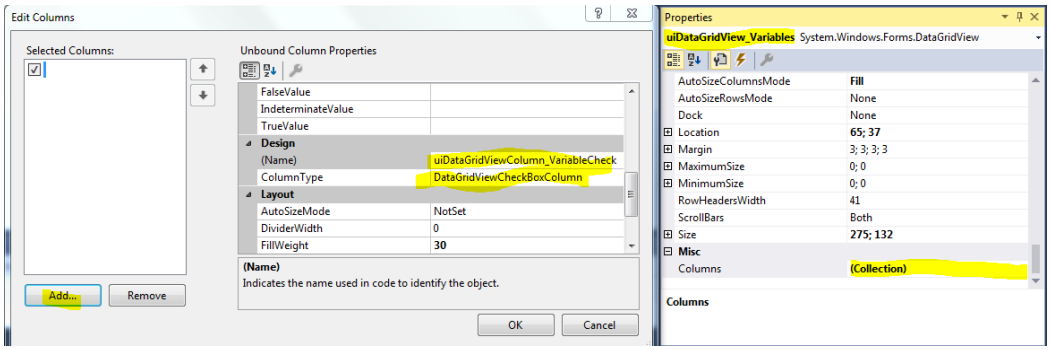
Figuur 5.8 Runs Test in Excel

In dit formulier moet de gebruiker aangeven op welke dataset men de functie wil toepassen. Hiervoor is naast het label 'Data set' een "DropDown" menu voorzien. De Cutoff Value wordt geselecteerd via "RadioButtons" (hiervan kan slechts één item tegelijk actief zijn) "Mean", "Median" en 'Custom value'. De "OK" – en "Cancel"-knoppen zijn standaard "Buttons" die men in de toolbox van de forms-designer kan terugvinden onder de map 'All Windows Forms'. De belangrijkste elementen op dit formulier zijn het "DropDown" menu met alle datasets en het "datagridview" waarin alle variabelen vermeld staan die horen bij de geselecteerde dataset. Het "DropDown" menu is van het type System.Windows.Forms.Combobox en het "datagridview" is van het type System.Windows.Forms.Datagridview. De naamgeving van al deze elementen op het formulier is belangrijk voor de verdere functionaliteit ervan en dient dus systematisch op dezelfde manier samengesteld te worden.



Figuur 5.9 All Windows Forms

De “datagridview” geeft aan de gebruiker de inhoud weer van een bepaalde dataset. In dit voorbeeld worden de verschillende variabelen geladen. Om de gebruiker toe te laten om een selectie te maken van meerdere variabelen, moet men aan het “datagridview” een extra kolom toevoegen met checkboxes (Figuur 5.8). Deze extra kolom hangt niet samen met de klasse ‘DataSet’ en dient tijdens het ontwerp van de lay-out toegevoegd te worden. Dit doet men door recht te klikken op de “datagridview”, vervolgens “Properties” te selecteren en onder ‘Misc’ een dummy kolom toe te voegen (Figuur 5.10).



Figuur 5.10 Misc

De naam van deze kolom moet opnieuw logisch gekozen worden, aangezien deze naam later bij het uitvoeren van de ‘Ok’-knop opgevraagd zal worden. Standaard kozen wij om telkens “uiDataGridViewColumn_zelftekiezennaam”. In dit geval is de naam van deze kolom dus ‘uiDataGridViewColumn_VariableCheck’. Bij sommige functies zijn meerdere dummy kolommen toegevoegd (bijvoorbeeld bij TimeSeriesGraph om verschillende variabelen ten opzichte van elkaar te plotten). Dit gebeurt op dezelfde manier als hierboven beschreven.

Nadat men alle elementen in het formulier heeft toegevoegd, kan men functies toevoegen in de vorm van code. Deze opent men door in de form designer op een willekeurige plaats rechts te klikken en vervolgens “View code” selecteert. Code 5.1 toont de code die men dient toe te voegen aan deze klasse, voorzien van commentaar.

```
using System;
using System.Windows.Forms;
using NoruST.Forms;
using NoruST.Presenters;
using NoruST.Domain;
using System.Collections.Generic;

namespace NoruST.Forms
{
   public partial class RunsTestForRandomnessForm : Form
   {
      // In de volgende code wordt het form gekoppeld aan de dataset
      private RunsTestForRandomnessPresenter presenter;
      private const string formTitle = "NoruST - Runs Test";
      public RunsTestForRandomnessForm()
      {
         InitializeComponent();
      }
      public void setPresenter(RunsTestForRandomnessPresenter RunsTestForRandomnessPresenter)
      {
         this.presenter = RunsTestForRandomnessPresenter;
         bindModelToView();
         selectDataSet(selectedDataSet());
      }
      private void bindModelToView()
      {
         uiComboBox_DataSets.DataSource = presenter.dataSets();
         uiComboBox_DataSets.DisplayMember = "name";
         uiComboBox_DataSets.SelectedIndexChanged += (obj, eventArgs) =>
         {
            if (selectedDataSet() == null) return;
            uiDataGridView_Variables.DataSource = selectedDataSet().getVariables();
         };
      }
      // Met deze code wordt het mogelijk om de geselecteerde dataset in het dropdownmenu op te vragen
      private DataSet selectedDataSet()
      {
         return (DataSet)uiComboBox_DataSets.SelectedItem;
      }
      public void selectDataSet(DataSet dataSet)
      {
         uiComboBox_DataSets.SelectedItem = null;
         uiComboBox_DataSets.SelectedItem = dataSet;
      }
      // De klikactie achter de 'Cancel'-button
      private void ui_Button_Cancel_Click(object sender, EventArgs e)
      {
         Close();
      }
```

```
        // Hier volgt de klikactie achter 'Ok'-button. Er wordt een lijst aangemaakt van objecten van het type 'Variable' op basis
        // van de geselecteerde checkboxes in het datagridview. Op het form moet het datagridview de naam
        //'uiDataGridView_Variables' hebben en moet de aangemaakte dummy kolom de naam
        //'uiDataGridViewColumn_VariableCheck' hebben. Daarna wordt de methode checkInput aangeroepen van de
        //presenter.
        private void btnOk_Click(object sender, EventArgs e)
        {
            List<Variable> variables = new List<Variable>();
            foreach (DataGridViewRow row in uiDataGridView_Variables.Rows)
            {
                if (Convert.ToBoolean(row.Cells[uiDataGridViewColumn_VariableCheck.Name].Value))
                {
                    variables.Add((Variable)row.DataBoundItem);
                }
            }
            bool check = presenter.checkInput(variables, selectedDataSet(), rdbMean.Checked, rdbMedian.Checked,
rdbCustomValue.Checked, uiTextBox_CustomCutoffValue.Text);
            if (check)
            {
                Close();
            }
        }
    }
}
```

Code 5.4 Runs test: Form

Op het einde van bovenstaande codevoorbeeld staat de actie die uitgevoerd wordt wanneer men op de 'Ok'-knop klikt. De header (in dit geval; '*private void btnOk_Click(Object sender, EventArgs e){}*) van deze functie wordt automatisch gegenereerd wanneer men in de formdesigner dubbelklikt op de 'Ok'-knop. Het is aan te raden om de klikacties achter elke knop op het formulier op deze manier toe te voegen omdat zo de functie automatisch gelinkt wordt aan de knop.

Het uitvoeren van de theoretische berekening achter de runs test gebeurt in de presenter van deze functie. Deze functie wordt aangeroepen vanuit het form via het commando presenter.checkInput(*argumenten*). De argumenten die meegegeven worden vanuit het form naar de presenter, zijn de lijst met geselecteerde variabelen, de geselecteerde dataset en alle inputs (in dit geval; de keuze van radiobutton en de waarde die ingegeven is als cutoff value) van de gebruiker op het formulier.

De functie checkInput van de presenter zal als return een booleaanse waarde hebben die aangeeft of alle input verwerkt is (true), dan wel of er een foutieve input gedetecteerd is van de gebruiker (false).

5.4 Model

De code die hoort bij Model (Code 5.5) is de minst complexe en kan hier heel kort beschreven worden. Het is mogelijk om bepaalde inputs van de gebruiker (bijvoorbeeld welke radiobutton geselecteerd is) op te slaan in variabelen in het Model, maar aangezien de input hier minimaal is, kan men alle inputdata rechtstreeks meegeven als argument naar de presenter. Het Model van de runs test in dit voorbeeld is bijgevolg gedefinieerd onder 'Models' in de solutions explorer, maar bevat hier geen code. In functies met meer gebruikersinput (bijvoorbeeld OneVariableSummary) wordt wel gebruik gemaakt van het Model.

```
namespace NoruST.Models
{
  class RunsTestForRandomnessModel
  {

  }
}
```

Code 5.5 Runs test: Model

5.5 Presenter

De presenter is de belangrijkste functie in de interpretatie van het Model-View-Presenter principe. Hier gebeurt de verwerking van de input en de uitwerking van de functies om de runs test uit te voeren. In alle bestaande functies zijn dit bijgevolg de uitgebreidste klassen. Code 5.6 illustreert de presenter functie voor de runs test.

```
// de using-statements dienen voor het toevoegen van functionaliteiten uit andere klassen of namespaces, afhankelijk van
wat de functie die geprogrammeerd moet worden kunnen deze afwijken, maar om het model-view-presenter principe te
gebruiken moeten NoruST.Models, NoruST.Forms en NoruST.Presenters zeker toegevoegd worden.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NoruST.Forms;
using NoruST.Models;
using NoruST.Presenters;
using NoruST.Domain;
using DataSet = NoruST.Domain.DataSet;
using System.ComponentModel;
using System.Windows.Forms;
using Microsoft.Office.Interop.Excel;

namespace NoruST.Presenters
{
   public class RunsTestForRandomnessPresenter
   {                    // Koppel de presenter aan bijhorende view en model
      private RunsTestForRandomnessForm view;
      private RunsTestForRandomnessModel model;
      private DataSetManagerPresenter dataSetPresenter;

      public RunsTestForRandomnessPresenter(DataSetManagerPresenter dataSetPresenter)
      {
         this.dataSetPresenter = dataSetPresenter;
         this.model = new RunsTestForRandomnessModel();
      }

      public void openView()
      {
         view = view.createAndOrShowForm();
         view.setPresenter(this);
      }
      public BindingList<DataSet> dataSets()
      {
         return dataSetPresenter.getModel().getDataSets();
      }
                        // Hier volgt checkInput, de functie die vanuit de form aangeroepen wordt. Hier wordt ook bepaald
                        //welke argumenten (lees: input) de functie aanvaardt.
      public bool checkInput(List<Variable> variables, DataSet selectedDataSet, bool rdbMean, bool rdbMedian, bool
rdbCustomValue, string CustomCutoffValue)
      {
         if (variables.Count == 0) return false; // wanneer de gebruiker geen variabele geselecteerd heeft, stop functie

         _Worksheet worksheet = WorksheetHelper.NewWorksheet("Runs test");
         int column = 1;
         int row = 2;
         worksheet.Cells[row, column] = "Runs test for randomness"; // schrijf strings naar worksheet
         worksheet.Cells[row++, column] = "Observations";
         if (rdbMean) worksheet.Cells[row++, column] = "Mean";
         if (rdbMedian) worksheet.Cells[row++, column] = "Median";
         if (rdbCustomValue) worksheet.Cells[row++, column] = "Custom cutoff Value";
         worksheet.Cells[row++, column] = "Below cutoff";
         worksheet.Cells[row++, column] = "Above cutoff";
         worksheet.Cells[row++, column] = "Number of runs";
         worksheet.Cells[row++, column] = "E(R)";
         worksheet.Cells[row++, column] = "Stddev(R)";
         worksheet.Cells[row++, column] = "Z-Value";
         worksheet.Cells[row++, column] = "P-Value (two-tailed)";
         ((Range)worksheet.Cells[row, column]).EntireColumn.AutoFit();
```

```
            row = 1;
            column = 2;
            foreach (Variable variable in variables) // deze loop wordt herhaald voor elke geselecteerde variabele van datagridview
            {
                worksheet.Cells[row++, column] = variable.name; // schrijf naam variabele naar worksheet
                var range = variable.getRange().Address(true, true, true); // sla range variabele op in "range"
                worksheet.Cells[row++, column] = selectedDataSet.rangeSize(); // schrijf de hoeveelheid gegevens in de variabele naar worksheet
                var ntotal = selectedDataSet.rangeSize();
                if (rdbMean) worksheet.Cells[row++, column] = "=AVERAGE(" + range + ")"; // schrijf afhankelijk van de gebruikersinput de cutoffvalue naar worksheet
                if (rdbMedian) worksheet.Cells[row++, column] = "=MEDIAN(" + range + ")";
                if (rdbCustomValue) worksheet.Cells[row++, column] = CustomCutoffValue;
                var cutoffValue = (double)worksheet.Cells[row-1, column].Value; // lees de cutoffvalue vanuit excel en sla ze op in variabele 'cutoffvalue'
                int amountOfRuns = calculateRuns(worksheet, selectedDataSet, variable.getRange(), cutoffValue); // roep functie calculateRuns aan en sla het resultaat op in amountOfRuns
                worksheet.WriteFunction(row++, column, "COUNTIF(" + range + ",\"<\"&" + AddressConverter.CellAddress(row - 2, column) + ")"); // schrijf functie voor het berekenen van #above cutoff naar worksheet
                worksheet.WriteFunction(row++, column, "COUNTIF(" + range + ",\">\"&" + AddressConverter.CellAddress(row - 3, column) + ")"); // schrijf functie voor het berekenen van #below cutoff naar worksheet
                worksheet.Cells[row++, column] = amountOfRuns; // schrijf het resultaat van functie calculate Runs naar worksheet
                worksheet.WriteFunction(row++, column, "(2*" + AddressConverter.CellAddress(row - 4, column) + "*" + AddressConverter.CellAddress(row - 3, column) + ")/(" + AddressConverter.CellAddress(row - 6, column) + ")"); // schrijf overige functies naar worksheet
                worksheet.WriteFunction(row++, column, "SQRT(((" + AddressConverter.CellAddress(row - 2, column) + "-1)*(" + AddressConverter.CellAddress(row - 2, column) + "-2))/" + AddressConverter.CellAddress(row - 7, column) + ")");
                worksheet.WriteFunction(row++, column, "(" + AddressConverter.CellAddress(row - 4, column) + "-" + AddressConverter.CellAddress(row - 3, column) + ")/" + AddressConverter.CellAddress(row - 2, column));
                worksheet.WriteFunction(row++, column, "2*(1-NORMSDIST(ABS(" + AddressConverter.CellAddress(row - 2, column) + ")))");
                ((Range)worksheet.Cells[row, column]).EntireColumn.AutoFit();
                row = 1;
                column++;
            }
            return true;
        }

        private int calculateRuns(_Worksheet sheet, DataSet dataSet, Range range, double cutoff) // functie die het aantal runs bepaalt
        {
            int runs = 1;
            double[,] array = RangeHelper.To2DDoubleArray(range); // Kopieer data die in range zit naar 2D-array
            double[] array2 = new double[array.Length];
            for (int i = 0; i < array.Length; i++)
            {
                array2[i] = array[i, 0];
            }
            for(int i = 0; i < array.Length-1; i++)
            { // als het het verschil tussen waarde en cutoff van teken verandert bij het doorlopen van de array, moet het aantal runs verhoogd worden met 1
                if(Math.Sign(array2[i] - cutoff) != Math.Sign(array2[i+1] - cutoff))
                {
                    runs++;
                }
            }
            return runs;
        }
    }
}
```

Code 5.6 Runs test: Presenter

Functies die uitgebreidere mogelijkheden of complexere berekeningen vereisen, worden in de presenter best opgedeeld in verschillende functies die elk een kleiner onderdeel van het geheel uitvoeren. Op die manier wordt het overzicht bewaard en kan het debuggen sneller gaan.

Nadat men de presenter vervolledigd heeft, zouden alle onderdelen van deze functie gelinkt moeten zijn.

5.6 Het aanmaken van grafieken

In de runs test for randomness worden geen grafieken gegenereerd. Aangezien dit echter een nuttige functie is waarbij vorig jaar vrij veel tijd is verloren, zal deze aan de hand van een codevoorbeeld (XR-Charts) gedemonstreerd worden. Deze code wordt altijd toegevoegd in de presenter. In dit voorbeeld (Code 5.7) wordt een grafiek gemaakt met 4 reeksen data.

```
        var Xcharts = (ChartObjects)sheet.ChartObjects(); //variabele die alle chartobjects bevat van het worksheet
        var XchartObject = Xcharts.Add(340, 20, 550, 300); // voeg een chart toe aan de variabele en geef dimensies mee
(positie en grootte)
        var Xchart = XchartObject.Chart; // .Chart is de eigenlijke grafiek van van het XchartObject
        Xchart.ChartType = XlChartType.xlXYScatterLines;
        Xchart.ChartWizard(Title: "X-Chart " + dataSet.Name, HasLegend: true);
        var XseriesCollection = (SeriesCollection)Xchart.SeriesCollection();
        var avgseries = XseriesCollection.NewSeries(); // voeg reeks toe
        var avgAvgseries = XseriesCollection.NewSeries();
        var UCLseries = XseriesCollection.NewSeries();
        var LCLseries = XseriesCollection.NewSeries();
        avgseries.Name = ("Observation Averages"); // geef reeks een naam
        avgAvgseries.Name = ("Center Line");
        UCLseries.Name = ("UCL");
        LCLseries.Name = ("LCL");
        avgseries.Values = plotaverages; // geef data mee aan de reeks, plotaverages is van het type double[] en vormt dus
een 1D-array van doubles
        avgAvgseries.Values = plotaverageOfAverages;
        UCLseries.Values = plotxChartUpperControlLimit;
        LCLseries.Values = plotxChartLowerControlLimit;
        avgseries.XValues = ArrayIndex; // geef x-waarden per reeks aan
        UCLseries.XValues = ArrayIndex;
        LCLseries.XValues = ArrayIndex;
        avgAvgseries.XValues = ArrayIndex;
```

Code 5.7 X/R Charts

5.7 Besluit

Indien men de voorgaande stappen correct volgde, zou het programma moeten kunnen compileren. Het toevoegen van functies kan bijgevolg gelijkaardig verlopen zoals hier voor de runs test for randomness is beschreven. Het opbouwen van het form, het model en de presenter kunnen grotendeels hierop gebaseerd zijn, met als gevolg dat andere functies enkel andere input zullen vereisen op het formulier en andere functies zullen hebben in de presenter. Op deze manier heeft de programmeur alle data beschikbaar tot in de presenter en kunnen daar complexere functies gebouwd worden.

Het is belangrijk om hier nogmaals te benadrukken dat als men als programmeur deze methode volgt, men zeer methodisch te werk kan gaan en de vaste structuur kan behouden zoals die hier uitgelegd is. Op die manier zal een nieuwe functie slechts afwijken op twee plaatsen van een oude functie, namelijk in de opbouw van het form (welke input moet de gebruiker leveren?) en in het uitwerken van de functies in de presenter (hoe wordt deze input verwerkt op de geselecteerde dataset?). Hierdoor biedt het softwarepakket zoals het opgeleverd werd een goed framework voor verdere ontwikkeling.

6 Gebruik van functies

In dit hoofdstuk zal aan de hand van drie functies, kort de functionaliteit ervan uitgelegd worden. Zo is het duidelijk wat er in de functies als input wordt gegeven en wat de functies teruggeven als output.

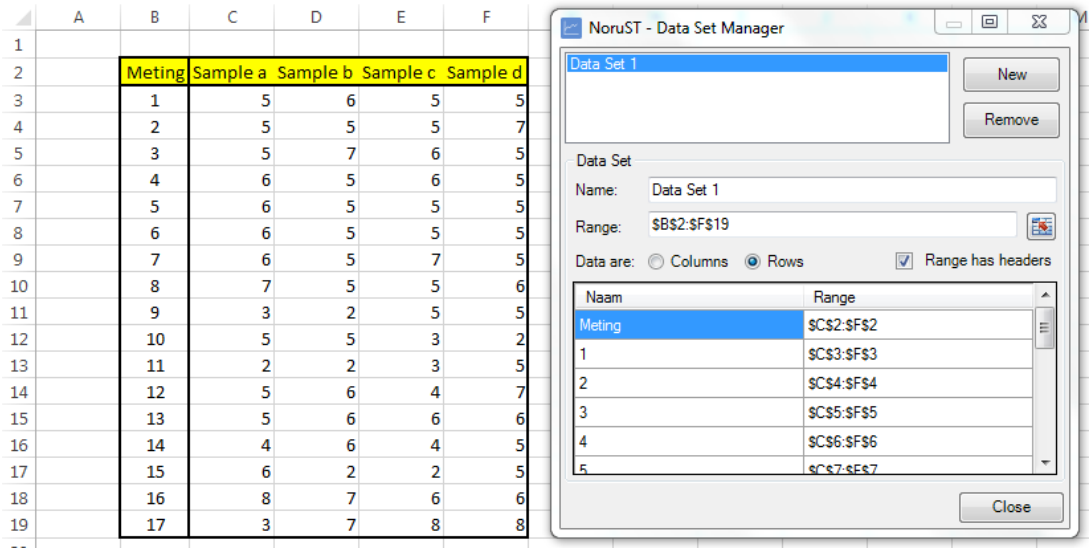
6.1 X/R-Chart

Deze functie leest een dataset in, waarin alle metingen (verzameld in subsamplegroepen) ondergebracht zijn. Om van een dataset een X/R-Chart te maken, selecteert de gebruiker uit het “DropDown” menu de gewenste dataset. Vervolgens klikt hij op OK waarmee de standaard instellingen geaccepteerd worden. Het is belangrijk om op te merken dat de geselecteerde dataset de variabelen als ‘simultane’ meetpunten bekijkt. Stel dat men bijvoorbeeld onderstaande data (Figuur 6.1) verzameld heeft:

Meting	Sample a	Sample b	Sample c	Sample d
1	5	6	5	5
2	5	5	5	7
3	5	7	6	5
4	6	5	6	5
5	6	5	5	5
6	6	5	5	5
7	6	5	7	5
8	7	5	5	6
9	3	2	5	5
10	5	5	3	2
11	2	2	3	5
12	5	6	4	7
13	5	6	6	6
14	4	6	4	5
15	6	2	2	5
16	8	7	6	6
17	3	7	8	8

Figuur 6.1 Voorbeeld data

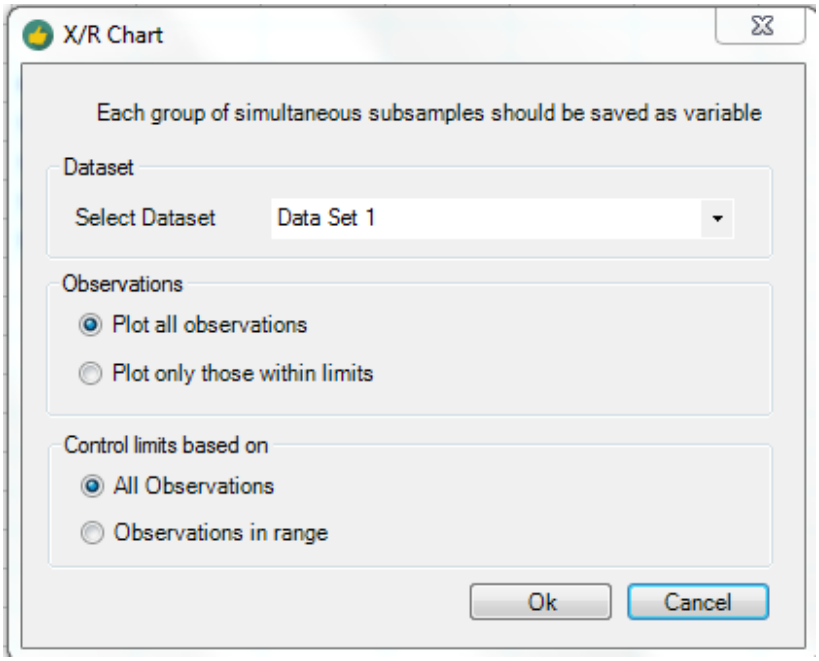
Op verschillende tijdstippen heeft men telkens vier metingen gedaan. Indien men hiervan een X/R-Chart wil genereren, moet men eerst de data inladen in de ‘Data Set Manager’. Dit gebeurd door alle data in deze dataset te selecteren en te kiezen om de data op te slaan als rijen. Op die manier komt meting 1 overeen met 1 variabele in de datasetmanager. Figuur 6.2 illustreert de vereiste actie in de ‘Data Set Manager’.



Meting	Sample a	Sample b	Sample c	Sample d
1	5	6	5	5
2	5	5	5	7
3	5	7	6	5
4	6	5	6	5
5	6	5	5	5
6	6	5	5	5
7	6	5	7	5
8	7	5	5	6
9	3	2	5	5
10	5	5	3	2
11	2	2	3	5
12	5	6	4	7
13	5	6	6	6
14	4	6	4	5
15	6	2	2	5
16	8	7	6	6
17	3	7	8	8

Figuur 6.2 Data inladen in Data Set Manager

Nu is de volledige dataset correct opgeslagen in de ‘Data Set Manager’. Als men nu een X/R-Chart genereert met de standaardwaarden (Figuur 6.3), bekomt men volgend resultaat (Figuur 6.4).

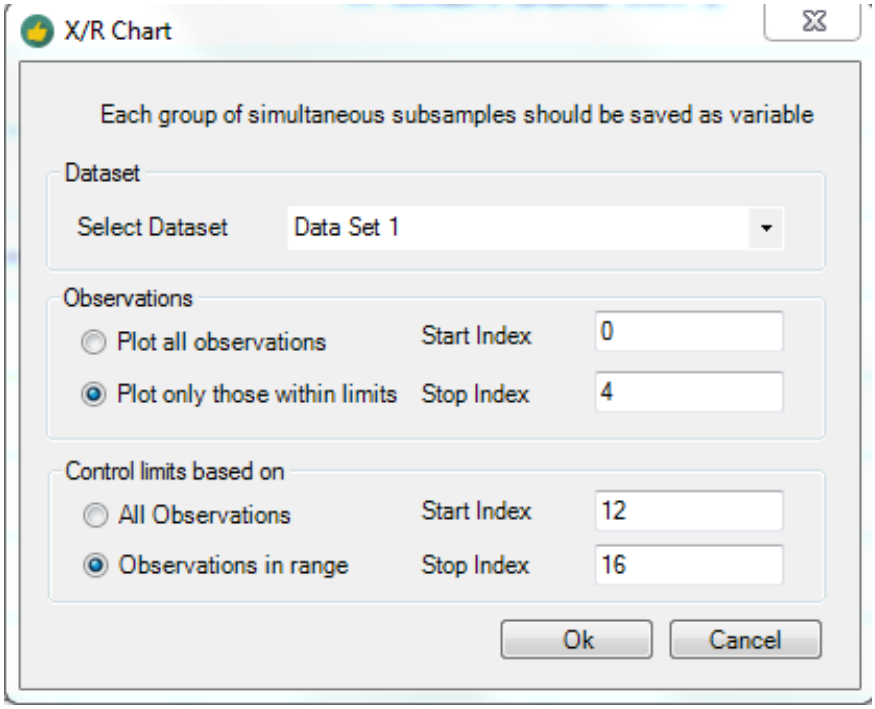


Figuur 6.3 X/R-Chart standaardwaarden

Index	Observat	Average	Max	Min	R
0	1	5,25	6	5	1
1	2	5,5	7	5	2
2	3	5,75	7	5	2
3	4	5,5	6	5	1
4	5	5,25	6	5	1
5	6	5,25	6	5	1
6	7	5,75	7	5	2
7	8	5,75	7	5	2
8	9	3,75	5	2	3
9	10	3,75	5	2	3
10	11	3	5	2	3
11	12	5,5	7	4	3
12	13	5,75	6	5	1
13	14	4,75	6	4	2
14	15	3,75	6	2	4
15	16	6,75	8	6	2
16	17	6,5	8	3	5

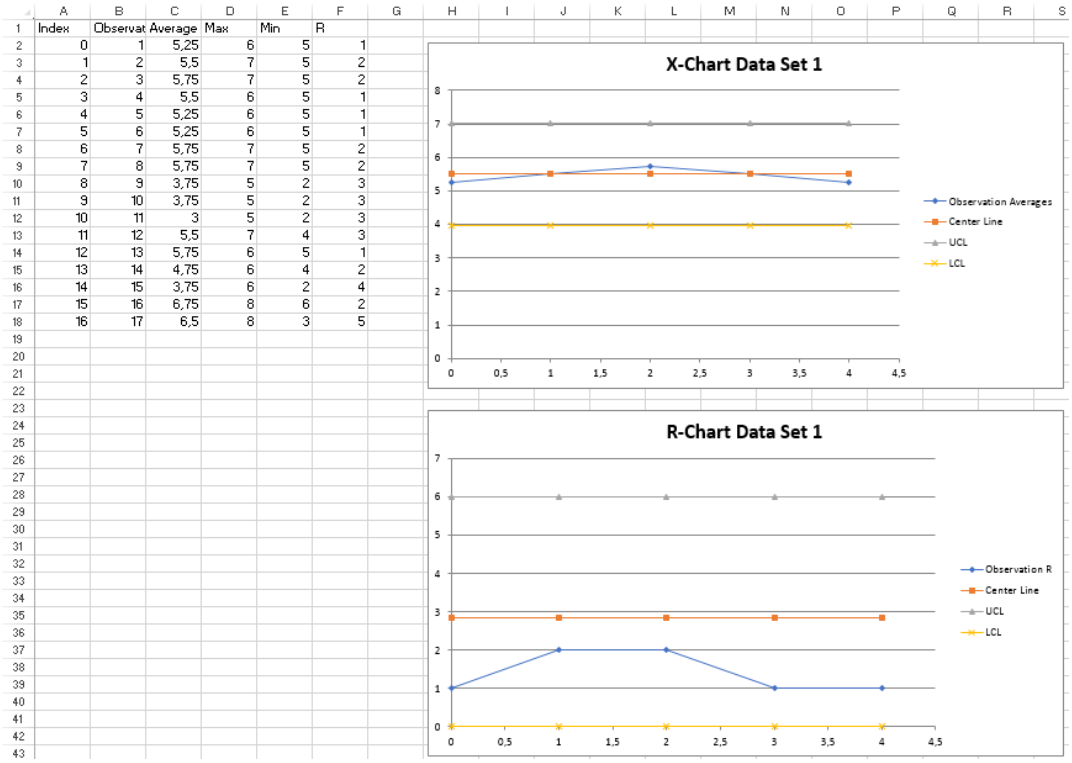
Figuur 6.4 Resultaat van X/R-Chart

Nu kan men als gebruiker ervoor kiezen om de X/R-Chart meer in detail te gaan bekijken. Men kan bijvoorbeeld vragen enkel de eerste vijf metingen te plotten en de controlelimieten (UCL, center line en LCL) te gaan berekenen enkel op basis van de laatste 5 metingen. Dit doet men door gebruik te maken van de automatisch gegenereerde index die naast de naam van elke meting vermeld staat (zie kolom A in Figuur 6.4). Om dit te doen selecteert men volgende opties in het formulier (Figuur 6.5).



Figuur 6.5 Aanpassingen X/R-Chart

Het bijhorende outputscherm zal er dan zo uitzien (Figuur 6.6):



Figuur 6.6 Aangepaste output

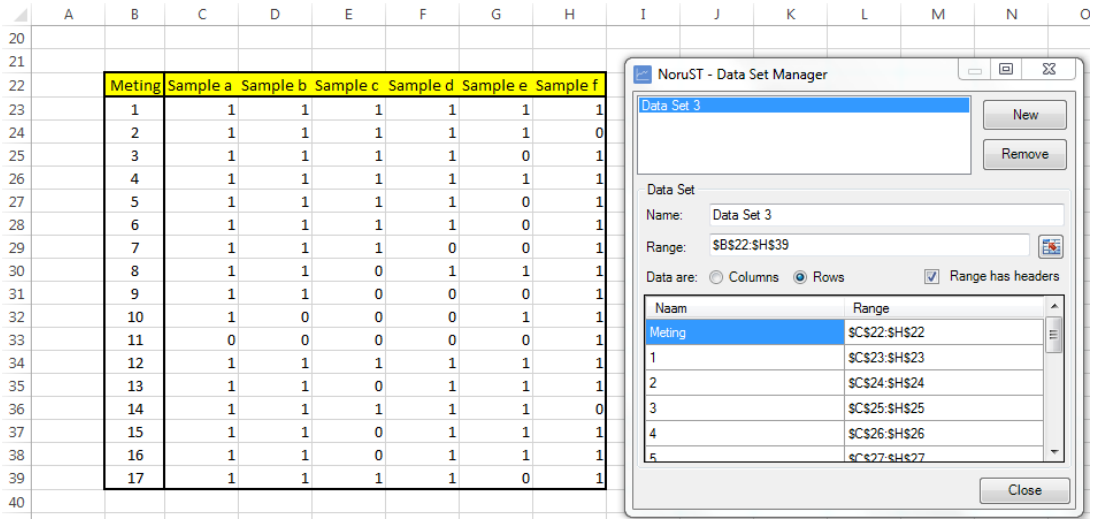
6.2 P-Chart

De P-Chart is op een gelijkaardige manier gemaakt. De functie laat de gebruiker toe om met ordinale data te werken die slechts twee waarden kan aannemen. Figuur 6.7 geeft een voorbeeld van dergelijke data:

Meting	Sample a	Sample b	Sample c	Sample d	Sample e	Sample f
1	1	1	1	1	1	1
2	1	1	1	1	1	0
3	1	1	1	1	0	1
4	1	1	1	1	1	1
5	1	1	1	1	0	1
6	1	1	1	1	0	1
7	1	1	1	0	0	1
8	1	1	0	1	1	1
9	1	1	0	0	0	1
10	1	0	0	0	1	1
11	0	0	0	0	0	1
12	1	1	1	1	1	1
13	1	1	0	1	1	1
14	1	1	1	1	1	0
15	1	1	0	1	1	1
16	1	1	0	1	1	1
17	1	1	1	1	0	1

Figuur 6.7 Voorbeeld data P-Chart

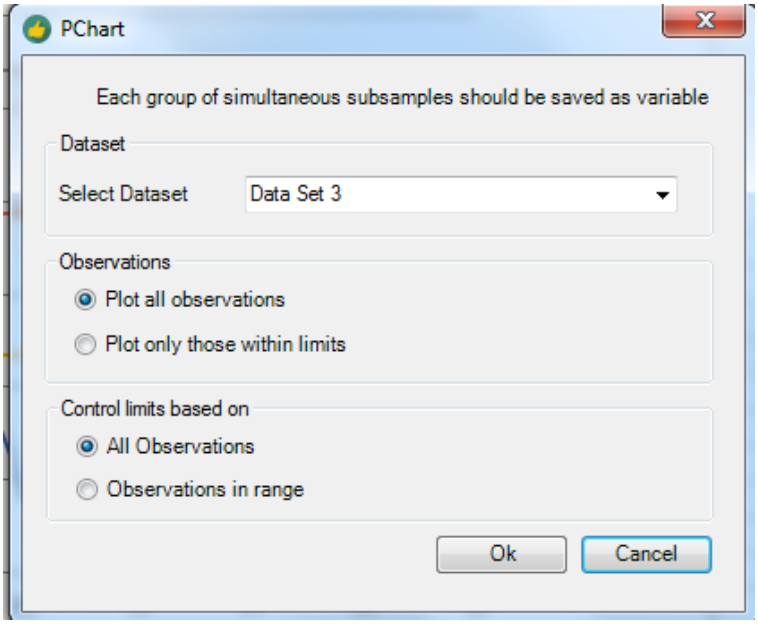
Om deze data te gebruiken voor het genereren van een P-Chart, moet men de metingen opslaan in rijen zoals hieronder weergegeven (Figuur 6.8).



Meting	Sample a	Sample b	Sample c	Sample d	Sample e	Sample f
1	1	1	1	1	1	1
2	1	1	1	1	1	0
3	1	1	1	1	0	1
4	1	1	1	1	1	1
5	1	1	1	1	0	1
6	1	1	1	1	0	1
7	1	1	1	0	0	1
8	1	1	0	1	1	1
9	1	1	0	0	0	1
10	1	0	0	0	1	1
11	0	0	0	0	0	1
12	1	1	1	1	1	1
13	1	1	0	1	1	1
14	1	1	1	1	1	0
15	1	1	0	1	1	1
16	1	1	0	1	1	1
17	1	1	1	1	0	1

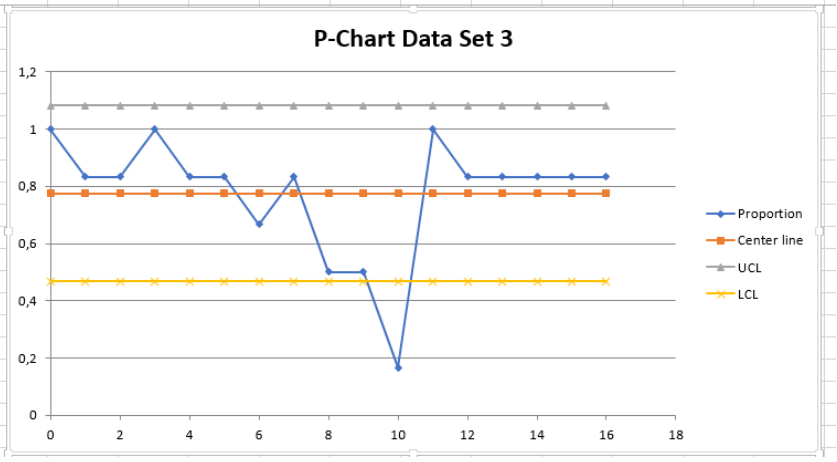
Figuur 6.8 Opslaan van data in rijen

De P-Chart genereren, kan nu opnieuw gebeuren via de default-instellingen of via de selectie volgens de start- en stopindexen. De default instellingen (Figuur 6.9) en bijhorend resultaat (Figuur 6.10) zijn hieronder weergegeven:



Figuur 6.9 Default settins P-Chart

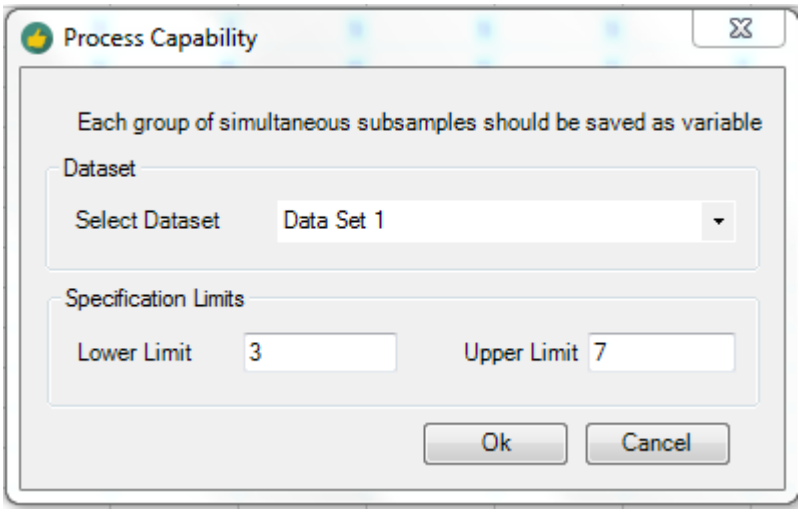
	A	B	C
1	Index	Observati	Average
2	0	1	1
3	1	2	0,833333
4	2	3	0,833333
5	3	4	1
6	4	5	0,833333
7	5	6	0,833333
8	6	7	0,666667
9	7	8	0,833333
10	8	9	0,5
11	9	10	0,5
12	10	11	0,166667
13	11	12	1
14	12	13	0,833333
15	13	14	0,833333
16	14	15	0,833333
17	15	16	0,833333
18	16	17	0,833333
19			
20			



Figuur 6.10 Output P-Chart

6.3 Process Capability

Deze functie maakt het mogelijk om bijvoorbeeld te controleren in hoeverre een proces in staat is om een output te leveren die binnen vooropgestelde specificatielimieten valt. Stel nu dat men op dataset 1 (de dataset in Figuur 6.1) deze functie wil toepassen. Hiervoor selecteert men onder 'Quality Control' de optie 'Process Capability'. De data in dataset1 ligt tussen de uiterste waarden 2 en 8. Indien men wil nagaan in hoeverre het proces een output kan leveren tussen de waarden 3 en 7 (de specification limits), doet men het volgende (Figuur 6.11).



Figuur 6.11 Process Capability

De bijhorende output is bijgevolg (Figuur 6.12):

	A	B	C	D	E	F	G	H	I	J	K
1	Index	Observati	Average	Max	Min	R		LSL	USL	Cp	Cpk
2	0	1	5,25	6	5	1		3	7	0,734526	0,683518
3	1	2	5,5	7	5	2					
4	2	3	5,75	7	5	2					
5	3	4	5,5	6	5	1					
6	4	5	5,25	6	5	1					
7	5	6	5,25	6	5	1					
8	6	7	5,75	7	5	2					
9	7	8	5,75	7	5	2					
10	8	9	3,75	5	2	3					
11	9	10	3,75	5	2	3					
12	10	11	3	5	2	3					
13	11	12	5,5	7	4	3					
14	12	13	5,75	6	5	1					
15	13	14	4,75	6	4	2					
16	14	15	3,75	6	2	4					
17	15	16	6,75	8	6	2					
18	16	17	6,5	8	3	5					

Figuur 6.12 Output Process Capability

7 Individueel deel

Bij de start van deze thesis bleek dat de studenten van vorig jaar er niet in geslaagd waren om het gebruikte programma (Visual Studio) geïnstalleerd en operationeel te krijgen op een andere computer dan diegene die zij gebruikten. Dit was de eerste uitdaging: het succesvol installeren van Visual Studio en het compileren van de code. Hieraan is heel wat tijd gespendeerd/verloren gegaan doordat hierover zeer weinig documentatie bestond. Door vaak samen te komen, hebben we het alsnog relatief snel voor elkaar gekregen dat iedereen Visual Studio had en code kon compileren.

Eens het programma van vorig jaar draaide, kon er in samenspraak met de promotor bepaald worden wat er juist verbeterd en aan toegevoegd moest worden. Hieruit bleek dat er een deel nieuwe functies toegevoegd moesten worden en dat er uit het bestaande programma een groot aantal functies niet compleet waren en vol “bugs” zaten. Om de taken succesvol te verdelen en een efficiënte manier van werken te bekomen, werd er eerst gekeken naar wie welke competenties heeft. Hieruit bleek dat één lid van de groep informatica gestudeerd had. De andere student had net als mij een basis kennis van JAVA programmatie. Logischerwijze gaf de programmeur aan dat hij graag de programmatie op zich zou willen nemen. Dat leek ons tevens het meest opportuun om zo weinig mogelijk tijd te verliezen. Wel gaf hij aan dat hij om snel te kunnen programmeren, nood had aan letterlijke instructies aangezien hij anders te veel tijd zou verliezen over het nadenken van het ontwerp van het programma en wat een programma juist moet doen. Aangezien één programmeur niet voldoende ging zijn om op tijd de thesis af te werken, heeft de andere student aangegeven dat hij tevens ging oefenen om zijn kennis omtrent het programmeren bij te schaven zodat ook hij zou kunnen helpen met programmeren. Mijn taak bestond er vanaf dan in om aan beide programmeurs de juiste info te bezorgen.

Die juiste info bestond uit verschillende delen.

7.1 Ontwerp van forms

Vooreerst was ik verantwoordelijk voor het ontwerp/uitzicht van de programma’s. Dit bestond erin de zogenaamde “forms” te ontwerpen op papier en nadien over te zetten in Visual Studio. Voor nieuwe functies gebeurde dit van nul. Bij oudere functies werd er gekeken of het bestaande form geoptimaliseerd kon worden. Om hierin zelf een leidraad te hebben, zocht ik naar gelijkaardige bestaande statistische add-ins voor Excel. Deze installeerde ik dan en ik vergeleek wat het handigste was en bv. welke functies bij ons niet van toepassing waren. Vervolgens creëerde ik een draft van hoe het “form” er zou moeten uitzien op papier.

De volgende stap bestond erin om de ontworpen programma’s op papier om te zetten in “forms” in Visual Studio. In het begin was het wat zoeken hoe dit in zijn werk ging. Door het analyseren van de bestaande “forms” kreeg ik het principe onder de knie. Omdat de naamgeving van de componenten van een “form” zo belangrijk is (deze namen worden later opgeroepen in de onderliggende code) werd afgesproken deze zo uniform mogelijk te houden.

7.2 Functieoverzichten

Naast het ontwerpen van de “forms” was ik verantwoordelijk voor de info die mijn thesisgenoten nodig hadden om de functies te programmeren. Hiervoor ontwierp ik Word documenten waar alle benodigde info over de desbetreffende functies in stond. Dit gebeurde zowel voor de nieuwe als voor de oude functies. De info haalde ik uit de handboeken ‘Kwantitatieve beleidsmethoden’, aangevuld met info van het internet. Het Word document bestond uit screenshots van internet of uit het boek, screenshots van het form, de theorie en de benodigde formules.

7.3 Ribbon

Samen met Andries werd de Ribbon opnieuw ontworpen. Hij maakte van de kleine knoppen grotere en ik voegde hier dan achteraf toepasselijke iconen aan toe. Tevens heb ik gekeken naar de logica van de locatie van de functies in de functiemenu’s. Voor de onderverdeling van de functies in de functiemenu’s, heb ik mij gebaseerd op de onderverdeling van dezelfde functie in “Stattools” en gelijkaardige statistische add-ins.

7.4 Schrijven van de paper

Bij het schrijven van de paper, heb ik me gefocust op het eerste deel. De nadruk hier lag op de installatie van de benodigde programma’s om te kunnen programmeren. Deze omvatten de installatie van ‘Visual Studio’, de “VSTO” plug-in en de installatie van GitHub. Verder beschreef ik de stappen die nodig zijn om ‘Visual Studio’ te gebruiken (openen van een project, testen en uitvoeren van een project,…). Aangezien ik de enige Mac/Apple gebruiker in de groep ben, was het duidelijk dat ik me hier tevens op zou focussen. Tot slot werd het stuk over aanpassingen ook grotendeels door mij behandeld. Doordat mijn andere groepsleden hier meer kennis over hadden, hebben zij het gedeelte betreffende het aanmaken van een nieuwe functie geschreven. Daar ik wel “forms” aangemaakt heb, werd dit deel ook door mij mee geschreven.

Bij het naderen van de deadline was er nog redelijk wat programmatiewerk. Op dat moment hebben we besloten dat ik me zoveel mogelijk zou bezig houden met het schrijven van de paper. Zo konden mijn co-auteurs zich volop focussen op de programmatie. Dit hield in dat de delen, door hun geschreven, werden nagelezen en samengevoegd door mij. Verder heb ik de algemene zaken van de paper geschreven (abstract, inleiding, besluit,…). Tot slot moest elke afbeelding de juiste verwijzing krijgen evenals de code.

7.5 Communicatie

Tijdens de thesisperiode heb ik voornamelijk de communicatie met de promotor gevoerd. Door duidelijk en transparant te communiceren met de promotor werden de verwachtingen scherp gesteld en kon er ook efficiënt worden gewerkt. Het behalen van de deadline was uiteraard ook alleen mogelijk doordat er op regelmatige tijdstippen werd geëvalueerd en feedback gegeven. Zelf ben ik tevreden over de manier waarop ik deze communicatie heb gevoerd.

7.6 Visie

Als ik terug kijk op het resultaat van de thesis, houd ik er over het algemeen een voldaan gevoel over. Het resultaat is een overzichtelijke add-in die visueel heel wat aantrekkelijker is dan de vorige. Tevens zijn bestaande functies gebruiksvriendelijker geworden en zijn er nieuwe functies toegevoegd.

Na het overlopen van de bestaande code, hebben we gezamenlijk beslist om de functies overzichtelijker te maken. Door gebruik te maken van het MVP principe is elke functie steeds op dezelfde manier opgebouwd. Dit maakt het zeer handig en gemakkelijk om een nieuwe functie toe te voegen alsook onderhoud uit te voeren bij bestaande functies. Over dit principe bestaat veel informatie over het internet dus kan men daar steeds op terug vallen indien er een probleem is.

Het invoeren van dit nieuwe principe had een groot nadeel. Doordat alle functies gebaseerd zijn op de datamanager, moest deze als eerste opnieuw geprogrammeerd worden. Hierdoor was er geen weg terug want doordat alle functies daar vanaf hingen, werkten de bestaande functies ook niet meer. Er is dan ook zeer veel tijd gestoken in het opnieuw operationeel krijgen van de bestaande functies. Hierdoor bleef er minder tijd over om ons te focussen op het programmeren van nieuwe functies.

Ik blijf er echter van overtuigd dat dit de juiste beslissing was en dat dit in de toekomst het vervolg van deze add-in eenvoudiger gaat maken. De goede documentatie van het aanmaken van nieuwe functies en de uniforme/systematische wijze om nieuwe functies te implementeren, betekenen een serieuze vooruitgang. Deze zorgen bijna letterlijk voor een stappenplan om nieuwe functies aan te maken. Ook zorgt de duidelijke installatie en gebruikshandleiding ervoor dat men hier niet te veel tijd mee zal verliezen bij een vervolg.

Doordat we alle drie werkten, was het vaak niet eenvoudig. Eén van de medestudenten is tevens van Gent en vandaar moest er een manier gezocht worden om eenvoudig van op afstand duidelijk te communiceren en samen te werken. Hierdoor werd het gebruik van Facebook en GitHub ingezet. Dit maakt het werken van thuis uit eenvoudig. Ongeveer éénmaal per week werd er toch samengekomen om de voortgang te bespreken.

Ik kan besluiten dat het programma in een fase is waarin het bruikbaar is voor studenten, gemakkelijk aanpasbaar maar nog niet volledig gefinaliseerd. Er is dus nog ruimte voor verbetering!

8 Reflectie seminarie

Het onderwerp van mijn thesis betrof het programmeren van statistische add-ins voor Excel. Deze thesis was het vervolg op een thesis van vorig jaar. Daar was reeds de basis gelegd aan het ontwerp van een statistische add-in. De add-in had reeds een deel van functies die werkten. Het doel van de thesis van dit jaar was om het bestaande programma te verbeteren meer bepaald, het verwijderen van fouten, optimaliseren en het toevoegen van nieuwe functies opgegeven door onze promotor.

Doordat onze thesis eerder praktisch van aard was, zijn veel zaken besproken in de verplichte sessies niet echt relevant naar onze thesis toe. Desalniettemin waren de sessies zeer informatief voor studenten met een ander type thesis maar ook voor mezelf voor het analyseren van studies en het schrijven of verrichten van een onderzoek in de toekomst. Ik werk momenteel in een productie omgeving en het is niet ongewoon dat hiervoor de besproken zaken aan bod komen.

Hier zijn we gestart bij het begin: de beschrijving van het probleem. De volgende stap is soms niet zo eenvoudig, nl. via het analyseren van een probleem om tot een probleemstelling te komen. Zo moet er steeds rekening gehouden worden met de focus. Op deze manier kan er een specifieke probleemstelling ontwikkeld worden. Dit is niet alleen nuttig in de context van het voeren van een onderzoek voor een masterproef, maar kan tevens nuttig zijn in het dagdagelijkse (bedrijfs-)leven. De probleemstelling staat in nauw verband met de “waarom” en de “wat”. Verder werd er tijdens de les aan de hand van duidelijke voorbeelden uitgelegd welke probleemstelling wel goed is en welke niet. Hiervoor werd de medewerking van de groep gevraagd wat het interessant maakte. Tevens is het gebruik van voorbeelden altijd leerrijker dan de pure theorie.

Vervolgens werd het literatuuroverzicht besproken. Hier werd opnieuw eerst duidelijk gemaakt wat het juist is en hoe het toegepast moet worden. Er werd aangegeven waar je kan zoeken en op die manier heb je toch een leidraad als je eraan moet beginnen. Bij ons beperkte het literatuuroverzicht zich tot het lezen van de vorige thesis en het gebruiken van de informatie uit de handboeken van ‘Kwantitatieve beleidsmethoden’.

Daarna werd het theoretisch kader toegelicht. Het ontwerpen van een theoretisch kader is niet eenvoudig indien je niet weet hoe er aan te beginnen. Een stappenplan moet hier verduidelijking in brengen. Weer opnieuw werd er met duidelijke voorbeelden gewerkt die een goed overzicht gaven over hoe een theoretisch kader er kan uitzien.

Tot slot werd er even ingegaan op het ontwikkelen van hypothesen. Hier opnieuw is het zeer interessant om te weten hoe je dit grondig kan doen. Alleen op die manier kan er een goede test ontwikkeld worden die de stelling dan weerlegt of bevestigt.

Het tweede seminarie was iets praktischer en betrof het toepassen van statistische methodes op data. Zo werd er een regressieanalyse uitgevoerd en geïllustreerd met duidelijke voorbeelden. Met deze methode kan worden geanalyseerd of een variabele afhankelijk is van de andere. Dit is een functie die we in onze thesis geprogrammeerd hebben. Data om te analyseren voor het doel van onze thesis hadden we echter niet. Desalniettemin zijn deze analyses ook nog interessant buiten een masterthesis. Zo kan er bv. in een productieomgeving gekeken worden of de hoeveelheid licht invloed heeft op hoe snel mensen werken.

Door hier een regressie op toe te passen kan gekeken worden of hier een verband tussen is. Het seminarie is hier vrij diep op ingegaan en legde tevens de interpretatie van een voorbeeld analyse uit.

Vervolgens werd er nog ingegaan op het voorspellen van de interactie tussen verschillende variabelen met behulp van regressiemodellen. Dit is interessant omdat men op die manier een voorspelling kan maken en deze achteraf kan nakijken of deze klopte. Dat laatste werd besproken tijdens de laatste sessie. De experimentele methode kan wederom in vele gevallen zijn nut bewijzen, zoals bv. een voorspelling checken. Een belangrijk aandachtspunt voor deze methode is wel dat er veel fouten bij de implementatie kunnen voorkomen. Door duidelijk te maken welke experimenten er allemaal zijn, kan de juiste keuze gemaakt worden.

Algemene Conclusie

Het algemeen doel in het begin van deze thesis bestond erin om de add-in zo te verbeteren dat het zeer eenvoudig is om nieuwe functies aan te maken en oude functies te onderhouden. Tevens zal deze paper een leidraad geven voor de installatie van de benodigde software. Dit zorgt ervoor dat hier weinig tijd wordt verloren. Door het implementeren van het MVP-principe, is het eenvoudig om een functie te begrijpen.

Er zijn reeds enkele nieuwe functies toegevoegd zodat de mogelijkheden van de add-in verbeterd zijn. Daarnaast werden er fouten uit bestaande functies gehaald.

Onze doelstelling was om een add-in te creëren waarin alle functies die zichtbaar zijn, werken. Op deze manier kan de add-in gebruikt worden met de functionaliteiten die het reeds bevat.

Tot slot kan er geconcludeerd worden dat “NoruST” veel vooruitgang geboekt heeft maar dat er nog steeds ruimte is voor verbetering. Zo zijn er een aantal functies nog niet geïmplementeerd. Het toevoegen hiervan zou door de degelijke documentatie eerder snel kunnen verlopen.

Wij hopen met deze thesis een bijdrage te hebben geleverd aan de optimalisatie van de gebruiksvriendelijkheid en de efficiëntie van het programma voor statistische add-ins in Excel.

Lijst met figuren

Lijst met codes

Bijlagen

Als bijlage bij deze masterproef werd een zip-bestand mee geleverd. Dit zip-bestand bevat 4 componenten:

1. Een map 'NoruST Broncode bestanden' met daarin alle code bestanden en project bestanden die nodig zijn om het project te openen met Visual Studio.

2. Een map 'NoruST Installatie bestanden' die de uiteindelijke installatie bevat die Visual Studio exporteert bij het compileren en publishen. Deze map bevat het bestand 'setup.exe' dat dient uitgevoerd te worden indien met de add-in wil installeren.

3. Het Word-document dat als bron diende voor de masterproef pdf. Dit brondocument bevat enkel het collectieve deel van de masterproef en dus niet de individuele reflectiecomponent en het individueel verslag van de masterproef seminaries zoals die wel in de eigenlijke masterproef te vinden zijn.

4. Een pdf-document dat enkel het collectieve deel van de masterproef en dus niet de individuele reflectiecomponent en het individueel verslag van de masterproef seminaries bevat zoals die wel in de eigenlijke masterproef te vinden zijn.

Bibliografie

Martina Vandebroek, A. W. (2011). *Kwantitatieve Beleidsmethoden Deel 1 & 2.* CENGAGE Learning.

Microsoft. (2017, Maart 2). *Visual Studio IDE*. Opgehaald van Microsoft: https://www.visualstudio.com/vs/

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN
Naamsetraat 69 bus 3500
3000 LEUVEN, België
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be

LID VAN ASSOCIATIE KU LEUVEN