master PSW

# Toegepaste multivariate analyse (TMA)

Prof. Dimitri Mortelmans

Quickprinter
Koningstraat 13
2000 Antwerpen
www.quickprinter.be

B51

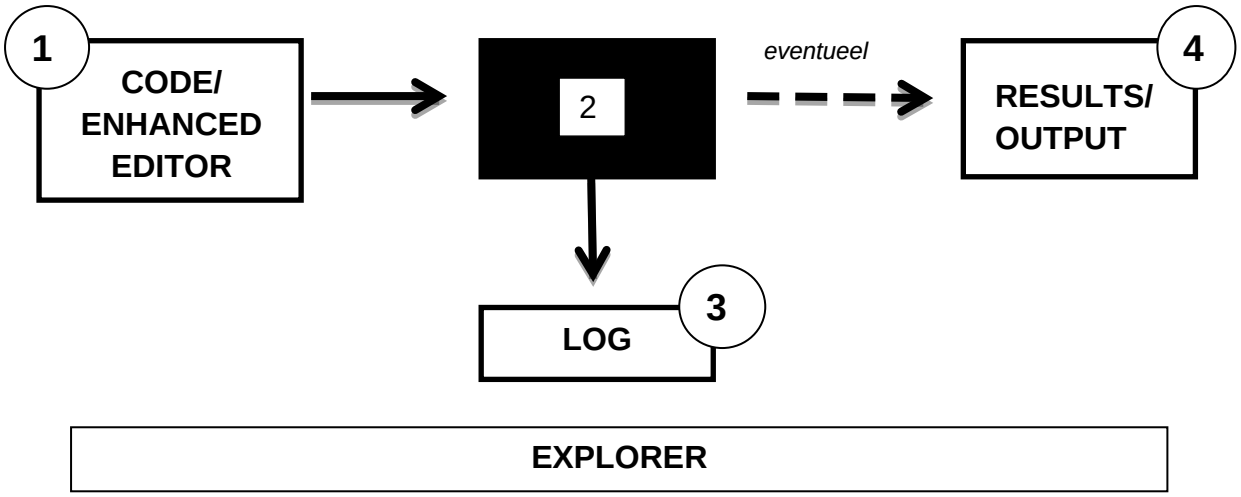
8,20 €

# SAS: GEDEELTE 1

## 1. De SAS studio

**SAS** = statistical analysis system

### 1.1. De interface van de SAS studio




**1) Code = enhanced editor**

= SAS-syntax wordt hier geschreven met instructies die aan SAS aangeven wat er moet gebeuren

- **enhanced** = houdt het midden tussen een tekstverwerker en een klassieke editor
  > teksverwerker : heeft heel wat functies om lay out van een tekst in te stellen (bvb. Microsoft Word)
  > klassieke editor : ontdaan van alle opmaakfunctionaliteit en dient enkel om tekst in te voeren en bewaren (bv. Windows Kladblok)

- SAS syntax opgeslagen in **ASCII –formaat**: enkel tekens

→ toch enhanced omdat aangevuld met enkele bijkomende functies:

A. Tekstbewerkingsfuncties:
   = standaard windows-functies om tekst kopiëren, plakken, verplaatsen of verwijderen (zie p. 24)

B. Syntax inkleuring:
   → zie pagina 24-25

### 2) Black box = SAS

→code geef je door aan black box (zie je niet)

### 3) Log

= laat zien wat je hebt gedaan, feedback over procedures
→ programma voert instructies van editor uit en vertel aan gebruiker wat er tijdens uitvoering allemaal gebeurd is

***- Drie groepen feedback:***

**A. NOTE**

= opmerking over de programmaregels
→ geven meestal aan dat alles normaal verlopen is
*→ bvb; hoelang procedure gelopen heeft, bij nieuw databestand hoeveel variabelen en hoeveel observaties..*

**B. WARNING**

= opmerkingen die aangeven dat er iets abnormaal is gebeurd zonder dat er een strikte fout in programma stond
*→ bvb. het maken/gebruiken van een lege variabele*

**C. ERROR**

= er is een fout in programmaregels waardoor programma instructies niet correct kan uitvoeren
→ zorgen dat SAS de uitvoering van programma stopt

### 4) Results = output

= resultaten van je statistische analyse (krijg je niet in geval van databeheer)
= resultaten van ingevoerde instructies

- elke statistische procedure stuurt output naar output venster
→ MAAR: optie **Noprint**: onderdruk je alle output

- standaard: output weergeven in ASCII
→ nu ook mogelijk om opgemaakte output te krijgen in HTML en kan je nadien nog bewerken (sinds het invoegen van windows-versie)

### 5) Explorer:

= geeft toegang tot de bestanden en directories van SAS en tot de windows omgeving
→ te vergelijken met windows verkenner

### 6) Commandolijn

→ vroeger gebruikt voor commando's die los stonden van schrijven van syntax: het besturen van het programma zelf (syntaxprogramma's oproepen, opslaan, wisselen tussen vensters, etc.)
→ Nu: door gebruik van windows omgeving; functionaliteit verloren

## 1.2. Cruciale concepten

- SAS beperkt zich tot databestanden op basis van extensie .sas

**- SAS heeft interne directorylogica:**

1) **Libraries = bibliotheken**

= gedeelde mappen tussen computer en SAS
= gedeelde SAS-map met databestanden waarnaar je kan verwijzen in een SAS-programma
→ interne SAS-naam die over windows-directory wordt geplakt
*→bvb: waar opslagen van procedure dat je hebt gedaan*

**→twee cruciale libraries: WORK & ESS:**
1) WORK = tijdelijke databestanden, wordt na afsluiten leeggemaakt
→ tip:in werken en nadien pas naar permanent overzetten
2) ESS = permanente databestanden, staat in myfolders
3) SASUSER? (p 27)

**→ voordelen van werkwijze:**

1. enkel essentiële bestanden te zien
2. schrijffouten vermijden omdat korte namen wanneer je verwijst (<> windows pad: lang en gecompliceerd)

**→ zelf library maken: libname librefnaam '/pad/'**

> libname creëert een nieuwe SAS-library over bepaalde windows-subdirectory
> binnen SAS library zal je dan enkel SAS bestanden kunnen zien
> buiten SAS heeft library andere naam (p.28) en zul je tevens ook bijvoorbeeld word-bestanden kunnen zien

### 2) My folders

= gedeelde map van de virtuele machine waar je databestanden en SAS-syntax kan uitwisselen tussen je eigen computer en de virtuele SAS-machine

### 1.2.1. Libraries en databestanden in de explorer

→ 4 iconen via explorer *(nog steeds zo?)*

#### *A) Favorite Folders en File shortcuts*

= dienen beide om snelle toegang te krijgen tot respectievelijk een map of een bestand op de harde schijf (windows)

**- File shortcut:**

= je maakt een snelle link naar een bestand op harde schijf en dit wordt opgeslagen in file shortcuts

→ door er dan op te klikken kan je deze snel openen

**- Favorite folders:**

= geven mogelijkheid om snel te gaan naar een folder waar je materiaal van je onderzoek opgeslagen is (zelfde dus als file shortcut)

→ standaard staan er al twee in; 'my desktop' (bureaublad) en 'my document' (mijn documenten)

#### *B) 'computer' icoon*

→ link naar harde schijf: je ziet alle bestanden (alle? Ook word?)

→ Sas-versie van windows verkenner

#### *C) SAS libraries*

**- standaard drie libraries:**

1) SAShelp= waar SAS databestanden opslaat die het gebruikt bij HELP-voorbeelden
2) SASURER= waar databestanden permanent worden opgeslagen (ESS)
3) WORK = waar tijdelijke databestanden worden opgeslagen

**-Data van databestanden zien:**

→ kan door te klikken op bestand in explorer (in library)

**- Inhoud van databestanden zien:**

→ 'view columns': overzicht van alle variabelen in bestand (bvb zien welke numeriek en welke tekstvariabelen zijn)

**- Eigenschappen van variabele bekijken**

→ column attributes: kenmerken van die bepaalde variabele

**- Cataloogbestanden:** naast databestanden ook dit in explorer
= bevatten info om gegevens uit databestand te structureren
**(<>SPSS:** gegevens die nodig zijn om gegevensbestand te structureren worden in 1 bestand bewaard)

**> SAS: maakt onderscheidt tussen:**
A. Descriptor portion: beschrijving van gegevens
B. Data portion: de eigenlijke data
C. Cataloogbestand: info over voorstelling van gegevens

### 1.2.2. SAS naar je hand zetten

#### *A) Output*

→ voor versie 7 van SAS had je weinig mogelijkheden om output te lay-outen
→ met versie 7 introduceerde men het ODS systeem, wel mogelijkheden hiermee

**1. de standaardwijze zonder extra syntax**
→ vermijden van ODS commando's: enkel in output viewer en HTLM formaat
→ enkele opties:
> naast standaardoutput ook HTML output
> output op harde schijf bewaren ipv in WORK
> stijl van output instellen
> resultaten in eigen webbrowser ipv result viewer

**2. Met behulp van ODS-syntax**
= verzameling van SAS- commando's die de output van SAS stuurt naar nieuwe databestanden of naar een output formaat:
> HTML bestand
> RTF bestand (via word openen )
> CSV bestand (via excel openen)
> PDF bestand

| | |
|---|---|
| **ODS outputformaat opties**;<br>SAS procedure(s)<br>**ODS Outputformaat** end; | **ODS HTML FILE** = 'C:\ESS\output.**html"** style = minimal;<br>Proq freq data = ….<br>**ODS HTML end;** |

## *B) Opties*

| **OPTIONS** kenwoorden; |
|---|

| **OPTIONS** | no center<br>No date; |
|---|---|

**- Opties om output in output-venster aan te passen:**

**1) linesize = regellengte**

→ standaard: 78 karakters per regel
→ vaak te weinig en maakt het moeilijk leesbaar
→ maximumlengte dat je kan instellen = 256 karakters per regel

**2) paginasize= paginalengte**

→ standaard: 20 regels per blad
→ maximumlengte = 32737 regels per blad
→ bij elke nieuwe statistische procedure, automatisch nieuwe pagina

**3) nocenter**

→ centreren van output uitzetten

**4) nodate**

→ standaard: welke datum en welk uur output gegenereerd
→ weglaten

**- Opties om commentaar in log-venster aanpassen**

**1) Nosource**

→doel: programma's die geschreven zijn in syntax worden niet nogmaals in LOG venster afgedrukt
→ gevolg: enkel weergeven van regels waar fouten in staan

**2) nonotes**

→ onderdrukt afprinten van notes in het log venster

**3) Errors =aantal fouten**

→ soms steeds opnieuw dezelfde fout, zal deze blijven herhalen tot limiet van fouten overschreden wordt.
→ standaard staat waarde op 20
→ doel: aantal identieke fouten die SAS tegenkomt in één programmaregel gereduceerd tot kleiner aantal

**- Andere opties:**

**1) _last_ = databestand**

→ standaard: SAS moet steeds weten op welk databestand hij statistische technieken moet toepassen, gebeurt door bij elk programma (data = databestand)

→ doel: in 1 sessie telkens zelfde databestand gebruiken zonder dit specifiek op te geven

**2) nofmterr**

→ standaard: wanneer format die je opvraagt niet wordt teruggevonden, zal data niet gebruikt kunnen worden of getoond kunnen worden

→ doel: optie onderdrukt de fouten die voorkomen met errors en laat data in originele vorm zien

**- Speciale optie:**

**1) title; "eigen titel"**

→ Standaard : "The SAS system"

→ doel: titel onderdrukken

→ "eigen titel": titel aan output van SAS toevoegen

## 1.3. SAS Syntax

**- onderscheidt SAS-syntax en programmastap:**

1° **SAS syntax/SAS programma** = geheel van programmastappen die samen onder één naam worden opgeslagen in het ASCII-bestand

2° **Programmastap**= een inhoudelijk samenhorende reeks instructies die begint met DATA of PROC en wordt afgesloten met RUN

**- Waarom Syntax gebruiken?**

*1) Tijdbesparend:* bvb commando's tegelijk laten uitvoeren ipv alles apart aanduiden

*2) Noodzaak:* bepaalde procedures enkel via syntax

*3) Herhaling van analyse:* bvb. analyse herhalen door kopiëren syntax en variabelennaam veranderen

*4) Overzicht:* na vele commando's wordt analyse complex, in syntax staan ze allemaal mooi onder elkaar zodat je kan overlopen

*5) lange termijn perspectief:* door enkel aanklikken, krijg je enkel output. Maar wanneer je iets moet veranderen moet je helemaal opnieuw beginnen. Indien syntax, gewoon opnieuw uitvoeren

**- Soorten programmastappen:**

→ algemene regel: - elke regel eindigt met ;
- elke programmastap wordt afgesloten met RUN bevel

**1) DATA-stap:**

= manipulatie van de gegevens
→ Data-invoer: nieuw databestand aanmaken
→ Databeheer: bestaand gegevensbestand wijzigen
→ programma begint altijd met woord 'Data'

```
DATA nieuw databestand; SAVE AS
SET oud databestand;    OPEN
Definieer data bewerking;
Run;
```

**2) PROC-stap:**

= Statistische bewerkingen
= univariaat, bivariaat of multivariaat techniek (?)
→ programma begint altijd met woord 'Proc'

```
Wat ga je doen? (proc benaming) + waar ga je het doen (bestand)
Hoe ga je het doen ?
Run
```

➔ **UITZONDERINGEN:**

1) *PROC FORMAT* : toewijzen van labels
2) *PROC SORT*: sorteren van data
= zijn eigenlijk data-stappen

*3) EENREGELPROGRAMMA's:*

a. **options** + kenwoorden;
= opties instellen
b. **Libname** + naam;
= van een windows map een SAS library maken

### 1.3.1. DATA-INVOER:

```
DATA naam nieuw databestand;
  INFILE 'bestand met de externe gegevens';
  INPUT variabelenstructuur;
RUN;
```

→ **DATA:** hier aangeven of tijdelijk bewaard wordt of permanent databestand
→ **INFILE** : welk extern gegevensbestand moet worden gebruikt om nieuw SAS-gegevensbestand op te bouwen
→ **INPUT**: welke variabelen SAS in het externe bestand moet gaan zoeken en op welke manier deze moet worden ingelezen

### 1.3.2. DATABEHEER

- ***bedoeling:*** komen tot databestand dat helemaal klaar is om analyse op te doen

```
DATA nieuw databestand;
SET oud databestand;

Nieuwe variabele = berekening;
Label;
…
RUN;
```

**- data-programma's worden in twee stappen uitgevoerd:**

1) **Compilatiefase:**

→SAS checkt of onderzoeker geen syntax fouten heeft geschreven
> indien dit geval stopt uitvoering van deze programmastap

→ Creëert hier ook *inputbuffer* (wordt enkel gebruikt wanneer je data inleest)

→ creëert *Program Data vector* (PDV)
= gereserveerde geheugenplaats waar databestand stap voor stap wordt opgebouwd alvorens het wordt weggeschreven
> DATA + naam: locatie waar resultaten uit PDV worden opgeslagen
> SET + naam: plaats waar gegevens voor bewerking moet ophalen
- ✓ Start: alle variabelen op missing
- ✓ Eerste record ingelezen en gevraagde bewerkingen op uitgevoerd
- ✓ Eerste lijn weggeschreven in PDC
- ✓ Volgende lijn...
- ✓ Alle info over vorige lijn wordt gewist

> !! Je kan niet et PDV automatisch informatie over records heen overdragen; daarvoor bevel RETAIN nodig

2) **Uitvoeringsfase:**

### 1.3.3. Proc-programmastappen

→ elke statistische techniek heeft eigen procedure

## 1.4. Moeilijkheden en fouten bij programmeren

**- Veelvoorkomende fouten:**

1) Puntkomma vergeten
2) Aanhalingstekens vergeten
3) Verkeerd databestand gebruikt

**- dedecteerbare fouten en problemen**

1) ***Een procedure blijft lopen:*** 2 mogelijke omstandigheden
   a. Interactieve procedure
      → normaal dat het blijft lopen
      → schatten van modellen meermaals achter elkaar
      → zie later
   b. Niet interactieve procedure
      → vergeten van RUN bevel of puntkomma na RUN bevel

### *2) Er verschijnt geen output*

→ is normaal bij DATA stappen
→ kan je nagaan of alles correct verlopen is door frequentietabellen

### *3) Variabele die werd aangemaakt is leeg*

→ komt bijna uitsluitend voor in databeheerprogramma's
→ zijn dus vaak fouten in programmeerwerk

### *4) Ernstige fouten*

→fouten waardoor SAS sessie onderbroken wordt
> bvb. met teveel gebruikers aan het werken aan één databestand
> zie pagina 50

## 1.5. Syntax structureren en becommentariëren

**- 1 zin commentaar:** *commentaar;

**- commentaar-alinea**: /* commentaar */

# 2. Databeheer met SAS

## 2.1. Een vragenlijst omzetten in een SAS-bestand (extra boek)

**- Omzetten van een vragenlijst in een databestand omvat volgende stappen:**
1) een codeboek maken
2) een codeboek in SAS-bestand overbrengen
3) een databestand bewaren en gegevens invoeren

### 2.1.1. Een codeboek maken

#### *A) Typen vragen herkennen*

**- één variabelen kan slechts één gegeven bevatten**

> DUS: telkens afvragen: wat is de kleinste eenheid informatie die ik in één veld kan invoeren?
> GEVOLG: soms meerdere variabelen voor één vraag te coderen

**- Soorten vragen en bijbehorende variabelen:**

***1) Eenvoudige categoriale variabele***

→ één variabelen die twee waarden kan aannemen (categorieën)
→ *bvb. geslacht: man/vrouw*

***2) Eenvoudige continue variabele***

→ één variabelen met x aantal waarden (meer digits vrijmaken dan 1)
→ *bvb. leeftijd; 3 digits = 100 jaar*

***3) Meerdere antwoordmogelijkheden***

→ voor elk antwoord (categorie) een variabelen te voorzien met vaak 0/1 als waarden
→ voordelen van 0/1-codering: je kan variabelen optellen
→ *bvb. waarvoor gebruik je iets: 3 categorieën aangeduid, door 0/1 codering gemakkelijk optellen en zien dat er 3 zijn aangeduid.*

***4) Een tekstvariabele***

→ moet telkens een aparte variabele voor worden voorzien NAAST de categoriale variabelen voor indien het bolletje is aangekruist in geval van een vraag met meerdere antwoordmogelijkheden
→ worden bewaard in apart type van variabelen: **stringvariabelen**

***5) Een groep (categoriale) variabelen: schaal***

→ variant van eerste type
→ reeks items of uitspraken telkens antwoorden op eenzelfde categoriale schaal
→ voor elk item of uitspraak variabelen voorzien

***6) Een uniek codenummer***

= unieke koppeling tussen papieren vragenlijst en de case in SAS bestand
→ !! niet echt soort variabelen
→ moet in elk onderzoek aanwezig zijn

## *B) Variabelen namen geven*

**- Algemene regels:**

1) beperkt tot 8 karakters (tegenwoordig meer, maar beter 8)
2) beginnen met letter (daarna letters of cijfers)
3) geen spaties
4) geen name die verwijzen naar commando's in je programma
5) geen speciale tekens zoals /
6) verschillende variabelen kunnen nooit dezelfde naam hebben

**- 3 manieren van naamgeving:**

***1) Doornummeren;***

→ variabelennamen doornummeren doorheen hele vragenlijst: V001-V009

***2) Vraagafhankelijke namen***

→ zelfde principe als doornummeren alleen splits je nummer van variabele op in twee delen:
> eerste twee digits voor vraagnummer
> laatste twee digits voor variabelennummer binnen vraag

→ bijkomende regels:
> vraag slechts één variabelen: achteraan '00' bij variabelennummering
> meerdere variabelen: start de variabelennummering op '01'

→ vb.p79

***3) Afgekorte namen***

→ afkortingen die verwijzen naar inhoud van variabele

### *C) Categoriewaarden en missende waarden*

**- Missing values = ontbrekende waarden**

= geen informatie over variabele voor bepaalde respondenten

**- 2 soorten: System missings of Users missings**

> lege cel
> of bepaalde code
➔ later meer

### *D) De vorm van het codeboek*

**- 2 mogelijke vormen waarin een codeboek kan worden aangemaakt:**

1) vragenlijstvorm
2) lijstvorm

→ *gaat dus over hoe onderzoeken op papieren versie of op pc versie zijn eigen variabelen gaat benoemen en naast vragen te schrijven, gaat nog niet over SAS*

**1) Vragenlijstvorm**

→ naast alle vragen op vragenlijst variabele namen plaatsen + codes toekennen aan categorieën van variabelen (bvb. ja = 1, Nee = 2)
→ *vb. p 82*

**2) Lijstvorm**

→ ook variabelennaam en categorieën zijn gecodeerd MAAR niet vraag wordt weergegeven, wel het label.
→ ziet eruit als variabele view in SPSS +SAS
→ *vb. P. 83*

### *E) Automatisch een codeboek aanmaken in SAS*

**- Codeboek maak je aan op papier**

→ idee: database wordt makkelijker hanteerbaard door af en toe te kijken naar codeboek
→ Maar niet altijd noodzakelijk omdat je na het aanmaken van variabelen ook overzicht van database in SAS krijgt (in lijstvorm)

> moet je wel rechtstreeks variabelen aanmaken in SAS

**- Aanvragen van codeboek in SAS:**

```
PROC CONTENTS DATA = naam databestand;
RUN;
```

## 2.1.2. De variabelen in SAS-bestand plaatsen

### *A) Een nieuw databestand aanmaken*

*→ zie pagina 84-86 voor hoe te doen via SAS aanklikken, hier enkel syntax*

**- Data + naam nieuw databestand**

→ hier bepaal je of je in work (tijdelijk) of sasuser werkt (permanent)

**- Manieren om variabelen aan te maken (+ zie later)**

***1) variabele als constante waarde die missing is***

A) numerieke variabele: **.;** (wordt als missing omschreven)

B) tekstvariabele : **“”;** (wordt als missing omschreven)

```
DATA  (SASUSER/WORDK). naam bestand;
  Variabelennaam 1 = .;
  Variabelennaam 2 = “”;
RUN;
```

→ voordeel: snelle aanmaak van nieuwe variabele

→ nadeel: standaardkenmerken worden door SAS aan variabelen toegewezen en moet je nadien eventueel wijzigen (bvb. lengte)

***2) variabele aan de hand van LENGHT bevel***

```
DATA (SASUSER/WORK).naambestand;
LENGTH   variabelennaam       4
              Variabelennaam      $256
;
RUN;
```

→ **LENGTH:** maak je ook nieuwe variabele aan en stel je meteen lengte van aantal tekens of digits in

> numerieke variabele : gewoon cijfer dat lengte aangeeft (bvb. 4)

> tekstvariabele : voor cijfer $ (bvb 256)

## 2.1.3. De variabelen analyseklaar maken

**- Belangrijk om eigenschappen (naast naam en type) van variabelen te definiëren**

> wat het net zegt

> hoe te lezen

>…

**- Eigenschappen:**
1) nieuwe variabelen aanmaken en lengte geven
2) label opgeven aan variabele
3) format opgeven aan categorieën van categoriale variabele
4) missing values definiëren

→ zie punt 2.3. Variabelen leesbaar maken!

## 2.2. Nieuwe databestanden

**- Basisvereiste voor een gebruiksklaar databestand: Datacleaning**
= proces waarbij onmogelijke of foute waarden op variabelen worden gecorrigeerd
→ komt later uitbundiger aan bod

**- Essentiële principes over databestanden in SAS:**

*1) SAS- Databestanden staan op harde schijf maar zijn niet zo eenvoudig te openen*
→ vaak verwijs je er enkel naar

*2) cataloogbestanden en SAS-databestanden kunnen tijdelijk of permanent zijn*
→ permanent: opgeslagen op harde schijf, ook nog steeds na afsluiten van SAS
> standaardlibrary: SASUSER
→ tijdelijk: bestanden worden gewist na het afsluiten van SAS
> standaardlibrary: WORK
→ libraries kan je uitbreiden

*3) gebruiker moet steeds libraries vermelden voor gebruik van databestanden*
→ permanent: librarynaam moet telkens voor naam van bestand staan
→ tijdelijk: librarynaam (work) moet je niet perse vermelden

*4) je maakt altijd een kopie van een bestand als je erin werkt*
→ je maakt altijd kopie door SET (bestand waar wijzigingen worden op uitgevoerd) en DATA (programma waar wijzigingen terecht komen)

*5) je kan geen bestanden bewerken die intern open staan*
→ indien databestand geopend is in viewtable view kan geen enkel databeheer programma werken

### 2.2.1. Nieuwe databestanden op basis van bestaande

```
DATA nieuw databestand;
SET oud databestand;
RUN;
```

**- Notatie tijdelijke en permanente databestanden + omzetten:**

**A) Tijdelijk:** (work.) bestandsnaam
*Bvb. DATA (work.)ESS1 = DATA ESS1*

**B) Permanent:** liberary.databestand
*Bvb. ESS.ESS1 = DATA ESS.ESS1*

➔ **omzetten via DATA en SET:**
- → je kan vertrekken van permanent of tijdelijke bestanden (SET)
- → je kan opslagen als permanent of tijdelijke bestanden (DATA)
  - >bvb Van permanent naar tijdelijk
  - *> DATA nieuw databestand (tijdelijk)* *(DATA ESSA)*
  - *> SET oud databestand (permanent)* *(SET ESS.ESS1)*
- → kan ook van tijdelijk naar permanent of zelfde houden

### 2.2.2. Selecties maken van databestanden

- Mogelijkheden:

**1) Variabelen selecteren**

```
Data essvariabelen;
Set ess1 (KEEP = gndr marital);
Run;
```

*Nieuw databestand essvariabelen (tijdelijk) op basis van oud databestand (tijdelijk)*
*Je neemt alle cases(respondenten) mee, maar enkel variabelen gender en marital*

➔ Je opent oud databestand met enkel die twee variabelen + opslagen

```
Data essvariabelen (KEEP = gndr marital);
Set ess1;
Run;
```

*Nieuw databestand essvariabelen (tijdelijk) op basis van oud bestand (tijdelijk)*
*Je neemt alle cases mee, maar enkel voor variabelen gender en marital*

➔ Je opent oud databestand volledig met alle variabelen + kan er bewerkingen mee doen, maar in nieuw bestand enkel opslagen met twee variabelen!!

- Keep = kolommen (variabelen) behouden, evenveel cases
- Drop = kolommen (variabelen) uitlaten, evenveel cases
→ je haalt **kolommen** weg/bij: even lang, maar minder breed

### 2) Observaties selecteren

#### A. WHERE BEVEL

→ beperkingen opleggen aan observaties die in nieuwe databestand terecht zullen komen

```
Data essvariabelen ;
Set ess1 (KEEP = gndr marital);
 WHERE gndr = 2;
Run;
```

→ kan ook complexer door met AND/OR te werken

#### B. IF-optie

= even veel aantal variabelen, maar minder cases
→ je haalt **rijen** weg: even breed, maar minder lang

```
Data essvrouwen;
Set ess1;
If gndr = 2;
Run;
```

*Nieuw databestand essvrouwen (tijdelijk) maken op basis van oude databestand (tijdelijk).*
*Je neemt alle variabele mee, maar kijkt enkel naar vrouwen*

➔kopiëren van gegevens van oude bestand naar nieuw maar enkel voor vrouwen

## 2.2.3. Bestanden samenvoegen

→ nieuwe bestand moet niet altijd gebaseerd zijn op kopie van andere, kan ook verschillende bestanden bij elkaar zijn

**- Twee manieren:**

### 1) Concateneren:

→ records/cases worden toegevoegd
→ twee gegevensbestanden bevatten dezelfde variabelen maar andere cases
→ breedte van bestand blijft gelijk, LENGTE neemt toe
→ 'bestanden worden onder elkaar geplakt'
→ variabelennamen moeten in beide bestanden gelijk zijn aan elkaar

**2) MERGEN;**

→ variabelen worden toegevoegd
→ gegevensbestanden bevatten dezelfde cases maar andere variabelen
→ lengte constant, BREEDTE neemt toe
→ !! identificatievariabelen nodig die in beide bestanden toelaat de cases juist aan elkaar te koppelen
> bvb. ID –variabele

*→ vb'en zie pagina 119*

### *2.2.3.1. Het concateneren van bestanden*

```
DATA naam nieuw bestand;
SET databestand1 databestand2;
RUN;
```

→ Het gegevensbestand dat eerst in de lijst staat zal eerst in nieuwe bestand staan
> MAAR van weinig belang omdat SORT-procedure dit kan veranderen

### *2.2.3.2. Het mergen van databestanden*

→ complexer als concateneren
→ vereiste: sleutelvariabele

**- 2 mogelijke situaties:**

1. ***Mergen op 1-1:***

**- situatie:** evenveel records in beide bestanden
*> bvb. databestand in twee stukken ingevoerd en gegevens van individuen uit ene bestand moeten worden samengevoegd met gegevens van individuen uit andere bestand*

**- voorwaarde:** elke sleutelvariabele komt één keer voor in beide bestanden (dezelfde IDNO dus)
→ zie p 120

## 2. ***Mergen op 1-veel***

**- situatie:** bestanden zijn van ongelijke lengte

*> bvb. bestand met 1 record per land (bvb huwelijksgraad) samenvoegen met bestand met veel records per land (bvb gender van land, BE komt vaak voor (sleutelvariabele) omdat meer inwoners dan 1 met gender)*

-**Voorwaarde**: sleutelvariabele is in 1 bestand uniek: komt in 1 bestand telkens maar een keer voor

→p 121

```
DATA huwelijken;
  INFILE datalines delimiter = ',' missover;
  INPUT country $ marp1000;
  DATALINES;
AT, 4.5
BE, 3.9
CH, 5.5.
…
;
RUN;
```

= nieuw databestand dat we aanmaken om later gebruiken

**- Mergen:** vereist dat beide bestanden gesorteerd zijn op sleutelvariabele

```
PROC SORT DATA = ess1;
BY cntry;
RUN;
PROC SORT DATA = huwelijken;
BY cntry;
RUN;

DATA samen;
MERGE ess1 huwelijken;
BY cntry;
RUN;
```

→ Set wordt vervangen door Merge

**- Mergen van bestanden met ingebouwde controle:**

```
PROC SORT DATA = ess1;
BY cntry;
RUN;
PROC SORT DATA = huwelijken;
BY cntry;
RUN;

DATA samen;
MERGE ess1 (IN = essx) huwelijken (IN = huwx);
BY cntry;
  Esscontr = essx;
  Huwcontr = huwx;
  Label esscontr = "record komt uit origineel essbestand"
        Huwcontr = "record komt uit extern huwelijkbestand";
RUN;
```

**- controle gebeurt in 2 stappen:**

***1) IN bevel: geheugenvariabele:***

→ bij samenvoegen ontstaat extra variabele die bijhoudt of bepaald record uit originele bestand gebruikt wordt bij samenvoegen (bvb ESSx= 1, huwx = 1)

> 0, 0 kan in feite niet voorkomen, 1,1 = perfecte match

→ nadeel: verdwijnen na het mergen uit geheugen

***2) nieuwe variabelen op basis van geheugenvariabele***

→ info uit geheugenvariabele wordt gekopieerd naar databestand

→ label bevel om leesbaarheid van variabelen te verbeteren

**- vervolgens tabel opvragen via proc freq data met variabelen huwcontr en esscontr**

→ kan je zien hoeveel matchen er voor komen (zie p 123)

→ je kan ook enkel perfecte koppelingen overhouden door nieuw bestand op basis van oude ('samen') en IF commando invoeren (p124)

### 2.2.4. Een bestand aggregeren

= samenvoegen van waarnemingen per groep

→ we veranderen het analyseniveau (analyse-eenheid) van het bestand (van eenheid naar groep)

→ kan enkel op numerieke variabelen

- **Geen éénvoudige procedure: gebruik maken van diverse omwegen en hulpmiddelen**:

1) groepeervariabele: moet nieuwe waarnemingen vormen

2) variabelen waarvan we kenmerken willen opnemen in bestand

**- Manieren:**

***1) Tables bevel;***

```
Proc Freq DATA =ess1
TABLES cntry / noprint out = aggreg1;
RUN;
PROC PRINT DATA = aggreg1;
RUN;
```

→ **Cntry =** groepeervariabele

→ **OUT** = schrijft resultaat van procedure weg naar nieuw SAS bestand met naam aggreg1

→ **Proc Print procedure** : om resultaat te bekijken op te vragen

➔ kan ook met **meerdere groepeervariabelen:** door cntry * gndr;

### *2) Means procedure:*

```
PROC MEANS DATA =ess1 noprint;
  CLASS cntry gndr;
  TYPES cntry*gndr;
  VAR age;
  Output out = aggreg1;
RUN;
PROC PRINT DATA = aggreg1;
RUN;
```

→ **MEANS**= berekent kengetallen van intervalvariabelen (gemiddelde)

→ **CLASS** = kengetallen (gemiddelde) laten berekenen voor subgroepen :
> groepeervariabele (soort van)

→ **TYPES** = zorgt ervoor dat enkel die combinaties van variabelen moet worden getoond in ons nieuw databestand die we vragen
*> bvb cntry * gndr = enkel combinatie van beide zien, niet apart en tezamen*

→ **VAR** = variabele waarvan kenmerken berekend moet worden

→ **OUTPUT OUT** = resultaat in nieuw databestand
? wrm nu ineens output ervoor?

### *3) Complexere aggregaties*

→ zie boek p.128!!!

- **BY**= groepeervariabele

**- RETAIN**
= zorgt bij een BY-bevel voor het behouden van de waarde van een variabele
→ **standaard:** SAS maakt waarden van elke observatie leeg wanneer hij bij doorlopen van bestand naar volgende observatie gaat en schrijft regel per regel weg in databestand
→ retain: betekent dat enkel de waarden van opgegeven variabelen worden weggeschreven in nieuw bestand

- **IF First.variabele=** bevel dat zegt dat bij het tegenkomen van een nieuwe waarde voor de eerste keer van een variabele, SAS bepaalde dingen moet doen

→ p 130