

SPSS

SYNTAXE

uživatelská příručka verze 2.17

Pomůcka pro práci s SPSS pomocí syntaxí

OBSAH

Úvod	2
SPSS - Statistical Package for Social Science	3
DATA prostředí	4
SYNTAX prostředí	5
Spouštění syntaxí	5
Typy příkazů	6
Zkracování příkazů	6
Proměnné	7
Formát proměnných	7
Vytvoření nové proměnné	7
Dočasné proměnné	7
Systémové proměnné	8
Operátory	9
Logický výraz	9
Matematické operátory	9
Relační operátory	9
Logické operátory	9
Přehled příkazů – Obslužné	10
Přehled příkazů – Transformační	13
Přehled příkazů – Funkce	14
Missing value funkce	14
Porovnávací funkce	15
Matematické funkce	16
Statistické funkce	16
Stringové funkce	17
Datumové funkce	18
Ostatní funkce	18
Přehled příkazů – Řídící	19
Přehled příkazů – Strukturální	23
Manipulace s proměnnými	23
Formát proměnných	24
Popisky proměnných	25
Další atributy proměnných	26
Přehled příkazů – Souborové	27
Otevírání / ukládání datových souborů	27
Spojování datových souborů	30
Transformace datového souboru	31
Struktura datového souboru	32
Další manipulace se soubory	34
Přehled příkazů – Statistické	35

ÚVOD

Občas se člověk něco naučí, případně i něco vymyslí, pak to nějakou dobu nepoužívá a zase to zapomene. Tak tento problém mám taky, proto jsem si začal psát tuto příručku, kterou jsem pojal jako pomůcku, do níž jsem si zapisoval veškeré příkazy a informace, které jsem se postupně naučil a potřeboval je při zpracovávání datových souborů.

Časem se z toho stal větší balíček, který začali využívat také mí kolegové, tudíž jsem rozhodl zpřístupnit to i případným dalším zájemcům. Shrnul jsem příkazy a funkce, které jsem využíval během doby, kdy jsem se zabýval prací s daty jednak v agenturách zabývajících se marketingovými průzkumy trhu a také na straně klienta. Pokusil jsem se tomu dát nějakou formu a výsledkem je tato příručka.

Příručka neslouží jako manuál pro naučení se práce se systémem SPSS, ani není vyčerpávajícím přehledem všech možností syntaxí. Díky mnoha příkladům pro použití jednotlivých příkazů poslouží hlavně v případě zapomenutí, jakou strukturu mají příkazy a jakými různými způsoby je lze použít. Ale na druhou stranu, díky menšímu obsahu, může být dobrým pomocníkem pro rychlý rozvoj základních znalostí analytiků, kteří se začínají učit zpracovávat data pomocí syntaxí a nemusí pak základní informace hledat v tisícistránkových originálních dokumentech.

Jelikož jsem první verzi vytvářel na základě zkušeností při práci s verzí SPSS 11.5 a poslední úpravy vznikaly při práci s verzí SPSS 17, je možné, že některé informace a hlavně některá omezení a možnosti nebudou platit pro každého stejně. Nicméně snahou je, aby informace byly platné pro poslední mnou používanou verzi.

V případě kritiky nebo konstruktivních připomínek mě prosím kamenujte na adrese sedlacev@centrum.cz.

sedlo

SPSS - STATISTICAL PACKAGE FOR SOCIAL SCIENCE

SPSS je statisticko analytický software.

SPSS se dá používat k několika věcem. Licenční politika je nastavena tak, že při koupi základní verze SPSS lze s tímto softwarem pracovat pouze omezené - např. je možné provádět operace s daty, pouze jednoduché frekvenční tabulky a pouze základní statistické analýzy. V plné verzi je možností mnohem více.

Možnosti využití jsou tyto:

- můžeme zpracovávat data (tzv. data processing) – spojování dat, čištění dat, identifikace duplicitních a podezřelých interview atd.
- tabulkové výstupy – při použití modulu TABLES lze využít pro základní tabelace, vhodné spíše pro přípravu tabulek, které slouží jako zdroj pro grafy, pro komplexní a detailní tabelace je lepší použít jiné aplikace
- statistické analýzy – segmentace, korelace, regrese, testy významnosti a další analýzy

SPSS má 5 základních pracovních prostředí:

- **DATA** – hlavní pracovní prostředí, ve kterém se zobrazuje soubor s daty; datové soubory mají příponu **.SAV**
- **SYNTAX** – prostředí, v němž pomocí příkazů můžeme zpracovávat data; soubory se syntaxí mají příponu **.SPS**
- **OUTPUT** – prostředí, v němž se zobrazují statistické výstupy, grafy nebo informativní hlášení a výpisy o průběhu některých operací a chybových stavech; otevře se automaticky po spuštění nějakého výstupu nebo při výpisu hlášení; výstup lze uložit do souboru s příponou **.SPO** nebo ho je možno vyexportovat do jiných formátů (např. **.XLS**, **.HTM**)
- **DRAFT OUTPUT** – stejné jako OUTPUT, ale výstupy jsou zde v jednoduchém textovém formátu.
- **SCRIPT** – prostředí, v němž můžeme vytvářet programy, pomocí nichž můžeme automatizovat některé úlohy v SPSS (automatická úprava výstupů, spuštění transformací dat, export! dat jako grafických souborů, atd.); jazykem, v němž se skripty píšou, je BASIC; soubory se skripty mají příponu **.SBS**

DATA PROSTŘEDÍ

V prostředí DATA jsou dvě záložky – Variable View a Data View. Přepínat se mezi nimi lze klávesovou zkratkou CTRL+T.

1) Variable View

Každý řádek představuje 1 proměnnou. Každá proměnná představuje nějakou otázku. U single choice otázek odpovědi na tuto otázku zaznamenáváme do 1 proměnné. U multiple choice otázek zaznamenáváme každou možnost do samostatné proměnné. Každý sloupec představuje konkrétní parametr proměnných:

- **Name** – jméno proměnné - může mít maximálně 64 znaků nesmí začínat číslicí, a nesmí obsahovat znaky ; , - /
- **Type** – datový typ - Numeric, String, Date, Dollar, atd.
- **Width** – počet pozic pro hodnoty, kterých proměnná může nabývat (minimálně musí být 1); např. pokud číselná proměnná bude nabývat hodnot 1 - 99, stačí tento parametr nastavit na hodnotu 2
- **Decimals** – počet desetinných míst u číselné proměnné; tohle je nutné vzít v úvahu i při definici Width; pokud budeme mít hodnoty 1.5, 2.5 apod., nastavíme Width na 2 a Decimals na 1 (kdybychom nechali nastavené Width 1 a Decimals 0, tak SPSS s tím i tak bude pracovat správně, ale problém by nastal při exportu dat do jiného datového formátu, třeba ASCII)
- **Label** – popis proměnné, může mít až 256 znaků
- **Values** – popis hodnot proměnné, každá hodnota může mít popis dlouhý maximálně 120 znaků
- **Missing** – nastavení hodnot, které budou považovány za chybějící (tzv. user missing – uživatelem nastavená chybějící hodnota), tyto hodnoty nejsou zahrnovány do statistických výstupů
- **Columns** – nastavení šířky sloupce pro zobrazení proměnných v záložce Data View
- **Align** – nastavení zarovnání pro hodnoty proměnných v záložce Data View
- **Measure** – nastavení typu proměnné SCALE, NOMINAL, ORDINAL

2) Data View

V této záložce každý řádek představuje 1 interview a každý sloupec představuje proměnnou (otázku). Konkrétní buňka tedy obsahuje odpověď konkrétního respondenta na příslušnou otázku. V buňkách lze nechat zobrazit buď konkrétní hodnoty, nebo popisky hodnot (třetí tlačítko zprava na nástrojové liště).

Stavová lišta

Stavová lišta se nachází v dolní části okna na okraji vpravo. Zobrazují se na ni informativní hlášky o stavu. V různých prostředích (DATA, SYNTAX, ...) se mohou lišit.

- **SPSS Processor is ready** (ve všech prostředích současně) – informace, že systém je v klidovém stavu; pokud se provádějí nějaké operace, tak se místo této hlášky, zobrazují informace o tom, co se právě provádí
- **Transformation pending** (ve všech prostředích současně) – tato hláška se zobrazuje, pokud byla spuštěna transformační operace, která zatím nebyla ukončena
- **Weight On** (pouze v prostředí DATA) – informace, že je zapnutá váha
- **Filter On** (pouze v prostředí DATA) – informace, že je aktivní filtr
- **Split File On** (pouze v prostředí DATA) – informace, že data jsou rozděleny na části
- **In 10 Col 3** (pouze v prostředí SYNTAX) – informuje, že kurzor je na řádku 10, ve sloupci 3 Type – datový typ - Numeric, String, Date, Dollar, atd.

SYNTAX PROSTŘEDÍ

V syntaxích zadáváme pomocí textových příkazů operace, které se mají vykonat na proměnných v datovém souboru.

Všechny *příkazy musejí být ukončeny tečkou*. Pokud naopak ukončovací tečku zapomeneme napsat, SPSS při zpracování příkazu vypíše chybovou hlášku a příkaz nespustí.

Příkaz může být rozdělen na více řádků. Pokud je příkaz rozdělen na více řádků, tak se ukončovací tečka píše až na poslední řádek, na kterém příkaz končí. Pokud je příkaz na více řádcích, tak nikdy nesmí být uprostřed příkazu volný řádek.

Každý příkaz musí začínat na novém řádku.

Syntaxe se provede na aktivním datovém souboru. Aktivovat požadovaný soubor můžeme taky pomocí syntaxe.

Při psaní syntaxe je rozumné dbát na přehlednost a čitelnost (tzv. strukturovat syntaxi). Pro zvýšení přehlednosti syntaxe používejme poznámky nebo nadpisy, které vysvětlují, co příslušná část syntaxe provádí. Části syntaxe, které řeší ucelené úlohy je dobré oddělovat několika volnými řádky. Příkazy, které jsou podřízeny nějakému jinému příkazu se odsazují zleva.

HELP (F1)

Pokud si nejsme jistí, jak se příkaz používá, tak stačí v syntaxi napsat jeho jméno, mít na něm kurzor a stisknout F1. Vyskočí nápověda s kompletní ukázkou všech možností a parametrů příkazu.

SPOUŠTĚNÍ SYNTAXÍ

Syntaxe se mohou spouštět 2 způsoby:

- Jednak lze spouštět přes menu Run v syntax okně.
- Jednak lze spouštět pomocí klávesových zkratk. Tento způsob preferuji, neboť urychluje práci.

Spouštění přes menu Run

- **All** – spustí všechny příkazy v okně
- **Selection** – spustí všechny příkazy, které jsou označeny; pokud není označeno nic, tak spustí pouze jeden příkaz, na kterém je aktivní kurzor.
- **To End** – spustí všechny příkazy od aktuální pozice kurzoru až do konce syntaxe

Spouštění pomocí klávesových zkratk

- **CTRL+R** – spustí syntaxi; a co spustí, záleží na tom, co máme označeno
- **jeden příkaz** – na tomto příkaze musíme mít aktivní kurzor a spustíme stiskem CTRL+R
- **více příkazů** – musíme si tyto příkazy označit (myší nebo pomocí Shift+šipky) a spustit CTRL+R
- **všechny příkazy** – stisknout CTRL+A (tímto se označí vše) a pak spustit CTRL+R
- **příkazy od pozice kurzoru do konce** – stisknout CTRL+SHIFT+END a spustit CTRL+R
- **příkazy od začátku do pozice kurzoru** – stisknout CTRL+SHIFT+HOME a spustit CTRL+R

TYPY PŘÍKAZŮ

Je důležité uvědomit si rozdíly mezi příkazy. A to rozdíly ve funkcionalitě a ve způsobu, jakým operace probíhají.

Typy příkazů podle funkcionality

- **Obslužné** – popisné a obslužné operace pro chod syntaxí a přehlednost výstupů
- **Transformační** – slouží k úpravám hodnot proměnných, vytváření proměnných, dopočítávání proměnných, kódování proměnných apod.
- **Funkce** – jsou to příkazy, které nevykonávají žádnou operaci, ale vracejí nějakou hodnotu; používají se s transformačními příkazy a to především pro přiřazení určité hodnoty společně s příkazem COMPUTE nebo pro ověření nějaké hodnoty společně s příkazem IF
- **Řídící** – pomocí nich směřujeme úpravy konkrétními větvemi, a ovládáme s nimi tok transformačních příkazů
- **Strukturální** – pomocí nich nastavujeme strukturu dat a formát proměnných
- **Souborové** – pomocí nich děláme operace s datovým souborem (spojování souborů, ukládání souboru, výběry)
- **Statistické** – pomocí nich vytváříme statistické (frekvence, průměry, ...) a analytické výstupy (korelace, segmentace, regrese, ...)

Typy příkazů podle průběhu operace

- Vyžadující ukončení operace
 - jedná se o příkazy, které přímo **mění hodnoty** proměnných nebo **řídí procesy změny hodnot** proměnných (tj. transformační, řídicí a některé obslužné / souborové příkazy) nebo **mění strukturu datového souboru** (např. seřazení řádků, spojení souborů apod.)
 - tyto příkazy jsou nejprve vykonány pouze virtuálně a k tomu, aby se provedené změny vykonaly i fyzicky se musí ukončit; dokud nejsou ukončeny, je stále možné zrušit jejich vykonání
 - ukončení proběhne spuštěním příkazu EXECUTE nebo spuštěním jakéhokoliv jiného příkazu požadujícího načtení hodnot proměnných (např. statistické, sortovací apod.)
 - fyzicky tyto operace probíhají tak, že se vykonají nejprve na prvním interview, až poté na druhém a tak postupně až do konce souboru (tzv. pracují na úrovni interview)
- Nevyžadující ukončení operace
 - jsou to příkazy, které **nemění hodnoty** proměnných, ale buď mění strukturu proměnných nebo vykonávají obslužné činnosti nebo popisné a statistické příkazy
 - tyto příkazy jsou vykonány okamžitě po jejich spuštění a není třeba je ničím ukončovat
 - jedná se o všechny statistické příkazy a některé obslužné / souborové příkazy

ZKRACOVÁNÍ PŘÍKAZŮ

Většina příkazů (kromě některých výjimek) lze zkrátit na 3-písmenné (některé pak na 4 nebo 5-písmenné).

Například:

EXECUTE	→	EXE
FREQUENCIES	→	FRE
TEMPORARY	→	TEM
LABEL	→	LAB
COMPUTE	→	COMP (Lze zkrátit pouze na 4 znaky)
RECODE	→	RECOD (Lze zkrátit pouze na 5 znaků)

PROMĚNNÉ

FORMÁT PROMĚNNÝCH

Fnum1.num2	(standardní číslo, num1 = počet cifer, num2=počet desetinných míst)
Nnum1.num2	(ascii číselný formát, nevyužité cifry jsou doplněné nulami)
Time	(časový formát, čas ve tvaru h.mm.ss)
Date	(datum formát, datum ve tvaru dd-mmm-yyyy)
Date10	(datum formát, datum ve tvaru dd-mmm-yy)
ADate	(datum formát, datum ve tvaru dd/mm/yyyy)
eDate	(datum formát, datum ve tvaru dd.mm.yyyy)
Anumber	(textový formát, number udává počet znaků)

VYTVOŘENÍ NOVÉ PROMĚNNÉ

Proměnnou můžeme vytvořit dvěma způsoby.

1. způsob:

Přímé vytvořením prázdné proměnné, pomocí příkazů k tomu určených. Při tom lze nastavit i formát proměnných. Pro textové proměnné je to jediný způsob.

Př1: Vytvoření prázdných (sysmisových) číselných proměnných.

```
NUMERIC var_list1 (F2) ... var_listN (F5.2).
```

Př2: Vytvoření prázdných textových proměnných.

```
STRING var_list1 (A10) ... var_listN (A5).
```

2. způsob:

Přiřazením konkrétní hodnoty do ještě neexistující proměnné pomocí transformačních příkazů.

Př1: Vytvoření nové a v tomto případě prázdné proměnné varX pomocí příkazu COMP a systémové proměnné \$sysmis.

```
COMP varX = $sysmis.
```

Př2: Vytvoření nové proměnné varX pomocí příkazu COMP a přiřazení hodnoty 0.

```
COMP varX = 0.
```

Př3: Vytvoření nové proměnné varX pomocí příkazu RECODE a přiřazení hodnot do nové proměnné.

```
RECODE varA (ELSE=COPY) INTO varX.
```

DOČASNÉ PROMĚNNÉ

S dočasnými proměnnými se pracuje stejně jako s běžnými proměnnými. Rozdíl je v tom, že dočasné proměnné jsou pouze virtuální, takže se fyzicky nevyskytují v datovém souboru a nemohou vstupovat do statistických výstupů. Platnost proměnné trvá do chvíle, než je ukončena příkazem EXE, nebo jiným příkazem ukončujícím transformační operace. Využívají se v případě, že potřebujeme proměnné použít jen jako pomocné pro nějaké výpočty nebo pro cykly a po ukončení výpočtu nebudou dále potřeba.

Dočasné proměnné mají před vlastním jménem uvozovací znak **mřížka (#)**.

Př1: Vytvoření dočasné proměnné #var_temp, do které přiřadíme hodnotu 1.

```
COMP #var_temp = 1.
```

SYSTÉMOVÉ PROMĚNNÉ

Systémové proměnné jsou takové proměnné, které jsou skryté a nejsou vidět v datovém souboru. Můžeme s nimi pracovat ve funkcích nebo v logických výrazech stejně jako s běžnými proměnnými v datovém souboru. Nemůžeme jim však přiřazovat žádné hodnoty.

\$SYSMIS

Proměnná, která má v každém řádku hodnotou `SYSMIS`. Užitečná je hlavně pro přímé přiřazení hodnoty `SYSMIS` do proměnné. Využití je možné třeba k přiřazení hodnoty `SYSMIS` do proměnné a hlavně pokud chceme vytvořit novou prázdnou proměnnou.

```
COMP var = $SYSMIS.
```

POZOR: nenechte se zlákat nadějí, že tuto proměnnou použijete v podmínce typu "IF (varX = \$SYSMIS)". Výsledkem vyhodnocení této podmínky je vždycky `SYSMIS`. Nikdy nedokáže porovnat hodnotu `SYSMIS` s jakoukoliv hodnotou a tudíž nikdy nebude výsledkem `TRUE` ani `FALSE`.

\$CASENUM

Proměnná, která v každém řádku obsahuje hodnotu určující pořadí příslušného řádku v datovém souboru. Možné využití je třeba k vytvoření ID proměnné.

```
COMP varID = $CASENUM.
```

\$DATE

Proměnná obsahující aktuální datum v mezinárodním formátu dd-mmm-yy.

\$DATE11

Proměnná obsahující aktuální datum v mezinárodním formátu dd-mmm-yyyy.

\$JDATE

Proměnná obsahující počet dnů od data 14. října 1582 do aktuálního data.

\$TIME

Proměnná obsahující počet sekund od půlnoci data 14. října 1582 do aktuálního času.

OPERÁTORY

LOGICKÝ VÝRAZ

Logický výraz je výraz, jehož výsledkem mohou být 3 hodnoty:

1 (pravda – TRUE) nebo 0 (nepravda – FALSE) nebo SYSMIS (nelze vyhodnotit)

Logický výraz se může skládat z matematických, relačních nebo logických operátorů anebo funkcí.

MATEMATICKÉ OPERÁTORY

+	součet	(4+2 = 6)
-	rozdíl	(4-2 = 2)
*	součin	(4*2 = 8)
/	podíl	(4/2 = 2)
**	mocnina	(4**2 = 16)

RELAČNÍ OPERÁTORY

=	EQ	rovnost	(varA=varB, varA EQ varB)
≠	◇	NEROVNOST	(varA≠varB, varA◇varB, varA NE varB)
<	LT	menší	(varA<varB, varA LT varB)
≤	LE	menší nebo rovno	(varA≤varB, varA LE varB)
>	GT	větší	(varA>varB, varA GT varB)
≥	GE	větší nebo rovno	(varA≥varB, varA GE varB)

LOGICKÉ OPERÁTORY

AND

Logický operátor, který slouží k řetězení logických výrazů.

Vrací hodnotu TRUE (oba výrazy jsou pravdivé) nebo FALSE (jeden z výrazů je nepravdivý)

```
(logický_výraz AND logický_výraz)
((var1 = var2) AND (var3 = var4))
```

OR

Logický operátor, který slouží k řetězení logických výrazů.

Vrací hodnotu TRUE (alespoň jeden z výrazů je pravdivý) nebo FALSE (oba výrazy nepravdivé)

```
(logický_výraz OR logický_výraz)
((var1 = var2) OR (var3 = var4))
```

NOT

Logická negace, který vyhodnocuje logický výraz a vrací hodnotu TRUE (pokud je výraz nepravdivý) nebo FALSE (pokud je výraz pravdivý)

```
NOT logický_výraz
NOT varA=varB
```

Logický výraz s AND	Vyhodnocení	Logický výraz s OR	Vyhodnocení
true AND true	true	true OR true	true
true AND false	false	true OR false	true
false AND false	false	false OR false	false
true AND sysmis	sysmis	true OR missing	true
sysmis AND sysmis	sysmis	missing OR missing	sysmis

PŘEHLED PŘÍKAZŮ – OBSLUŽNÉ

COMMENT

Příkaz, který neprovádí žádnou operaci. Slouží pouze k psaní poznámek v syntaxi, abychom si mohli uvnitř syntaxe psát komentáře. Pokud by komentář nezačínal tímto příkazem, SPSS by se snažilo je spouštět jako příkazy, což by způsobilo chybové hlášky.

```
* Text of coment.
```

Př: * Zde je část syntaxe k transformaci dat potřebné pro korelace.

ECHO

Vypíše zadaný text do výstupního okna. Užitečný k popiskům, které nám zpřehlední výstupy.

```
ECHO "Text".
```

Př: ECHO "Blok s korelacemi".

EXECUTE

Tento příkaz ukončuje transformační operace.

```
EXE.
```

Př: Příkazem COMP načteme do proměnné varX hodnotu 1. Fyzicky dojde k vytvoření této proměnné, ale hodnoty v ní budou až po spuštění příkazu EXE.

```
COMP varX = 1.
```

```
EXE.
```

CLEAR TRANSFORMATIONS

Tímto příkazem zrušíme předchozí vykonané transformační operace, které zatím nebyly ukončeny příkazem EXE nebo jiným způsobem.

```
CLEAR TRANS.
```

Př: Příkazem COMP načteme do proměnné varX hodnotu 1. Ale spuštěním příkazu CLEAR TRANS tuto operaci zrušíme a v proměnné varX zůstanou původní hodnoty.

```
COMP varX = 1.
```

```
CLEAR TRANS.
```

TEMPORARY

Pomocí tohoto příkazu nastavíme, že všechny transformační příkazy, které následují, budou provedeny pouze dočasně. Transformace budou vykonány, ale po jejich ukončení příkazem EXE nebo jiným, se hodnoty proměnných vrátí do původního stavu, jako byly před spuštěním příkazu TEM.

```
TEM.
```

Př: Nejprve zapneme dočasnost operací příkazem TEM. Poté provedeme výběr řádků, ve kterých je varX=1. Pak vyjedeme frekvenci proměnné varY a to bude pouze z interview, ve kterých platí, že varX=1. Díky TEM však výběr byl pouze dočasný a spuštěním příkazu FRE se výběr zruší.

```
TEM.
```

```
SEL IF (varX = 1).
```

```
FRE varY.
```

LEAVE

Tento příkaz potlačuje inicializaci hodnot proměnných při čtení dalšího řádku.

```
LEAVE.
```

Obvykle se využívá k vytvoření proměnné, která je kumulovaným součtem jiné proměnné.

Př: Příkazem COMP fyzicky sčítáme do proměnné Tot_varX hodnotu Tot_varX z předchozího řádku a hodnotu varX z aktuálního řádku. Tímto vytváříme proměnnou, která je kumulovaným součtem proměnné varX.

```
COMP Tot_varX = Tot_varX + varX.  
LEAVE Tot_varX.  
EXE.
```

DISPLAY

Spuštěním tohoto příkazu se do OUTPUT okna vypíše informace o datovém souboru. Informace mohou být seřazeny, můžeme vybrat, jaké informace chceme a můžeme vytvořit i seznam proměnných, o kterých tyto informace chceme.

```
DISPLAY SORTED parameter /VARIABLES = var_list.
```

Možnými parametry příkazu jsou:

SORTED – nepovinný parametr, který zajistí, že výpis proměnných bude seřazen abecedně

NAMES – vypíše seznam proměnných

INDEX – vypíše seznam proměnných a jejich pořadí v datovém souboru

LABELS – vypíše proměnné, pořadí a popisky

VARIABLES – vypíše proměnné, pořadí, popisky, typ, vstupní a výstupní formát

DICTIONARY – oproti VARIABLES navíc vypíše i šířku zobrazení a zarovnání

/VARIABLES – zde definujeme proměnné, které chceme zahrnout do výpisu

Př1: Vypíše seznam všech proměnných v abecedním pořadí.

```
DISPLAY SORTED NAMES.
```

Př2: Vypíše pro proměnné varA varB varC jejich jména, pořadí v souboru a popisek.

```
DISPLAY LABELS /VARIABLES varA varB varC.
```

LIST

Spuštěním tohoto příkazu se do OUTPUT okna vypíše hodnoty zadaných proměnných a v zadaných řádcích.

```
LIST var_list /FORMAT = parameter /CASES = FROM val1 TO val2 BY val3.
```

Př1: Vypíše hodnoty všech proměnných ve všech řádcích.

```
LIST.
```

Př2: Vypíše hodnoty proměnných varA varB varC v každém třetím řádku počínaje řádkem 10. V prvním sloupci budou čísla řádků.

```
LIST varA varB varC /FORMAT = numbered /CASES FROM 10 BY 3.
```

PŘEHLED PŘÍKAZŮ – TRANSFORMAČNÍ

Příkazy, pomocí kterých se mění hodnoty v proměnných nebo přiřazují hodnoty do jiných proměnných.

COMPUTE

Přiřazuje do konkrétní proměnné nějakou hodnotu. K přiřazení hodnoty lze použít číslo, funkci i logický výraz a také jakákoliv jejich kombinace.

COMP variable = FUNKCE(argumenty funkce).

COMP variable = 1.

COMP variable = var1 + var2.

COMP variable = SUM(var1,var2).

RECODE

Mění hodnoty proměnné na jiné hodnoty.

RECODE variables (val1=val2).

RECODE var1 var2

(LO thru 0,11 thru HI = SYS)

(1,2,3=COPY)

(4=5)

(6 thru 9=4)

(SYS = 10)

(ELSE=-1).

V příkladu je ukázáno, že lze měnit hodnoty u více proměnných najednou a pak různé kombinace na změnu hodnot. Hodnoty se podle závorek mění postupně zleva. Není možné konkrétní hodnotu měnit 2x (tj. že by byla ve dvou závorkách na levé straně a současně bych ji chtěl změnit na dvě různé hodnoty).

V proměnných var1 a var2 dojde ke změně jejich hodnot následujícím způsobem:

(LO thru 0,11 thru HI = SYS)	=> hodnoty od $-\infty$ do 0 a hodnoty od 11 do ∞ se vymažou
(1,2,3=COPY)	=> hodnoty 1,2,3 se zkopírují
(4=5)	=> hodnota 4 se změní na 5
(6 thru 9=4)	=> hodnota od 6 do 9 se změní na 4
(SYS = 10)	=> místo chybějící hodnoty SYS se doplní hodnota 10
(ELSE=-1)	=> jakékoliv jiné hodnoty se změní na -1 (ELSE musí být až na konci)

RECODE ... INTO

Mění hodnoty proměnné na jiné hodnoty, ale zapíše je do jiné proměnné a ta původní zůstane stejná.

RECODE variables1 (val1=val2) INTO variables2.

RECODE varA1 varA2 (1 thru 10=1)(ELSE=2) INTO varB1 varB2.

Pokud je v proměnné varA1 (varA2) hodnota 1 až 10 zapíše do proměnné varB1 (varB2) hodnotu 1, pro všechny ostatní hodnoty zapíše do varB1 (varB2) hodnotu 2.

COUNT

Slouží ke zjištění v kolika proměnných jsou určité hodnoty. Přiřadí do proměnné počet, kolik proměnných ze seznamu proměnných obsahuje hodnotu ze seznamu hodnot.

COUNT variable = variables (values).

COUNT var = var1 var2 var3 (2).

COUNT var = var1 var2 var3 (2) var4 var5 (3).

COUNT var = var1 TO var5 (1 THRU 3).

COUNT varA = var1 var2 var3 (LO thru 1)/varB = var1 var2 var3 (2 thru HI).

V posledním příkladě např. dojde k následujícímu: Do proměnné varA dá počet, v kolika z proměnných var1 var2 var3 je hodnota menší nebo rovna 1. Do proměnné varB dá počet, v kolika z proměnných var1 var2 var3 je hodnota větší nebo rovna 2.

PŘEHLED PŘÍKAZŮ – FUNKCE

Funkce se používají hlavně společně s transformačními, kde přiřazují hodnoty do proměnných a také pak s řídicími příkazy v logických podmínkách.

MISSING VALUE FUNKCE

Jde o funkce, které umožňují pracovat s proměnnými i v případě, že obsahují chybějící (missingové) hodnoty.

VALUE(var)

Pokud je nějaká hodnota proměnné definována jako USER MISSING, tak s ní pracuje jako s normální hodnotou. Ignoruje USER MISSING.

```
VAL(var)  
COMP varA = varB + VAL(varC).
```

MISSING(var)

Pokud je hodnota argumentu USER MISSING nebo SYSTEM MISSING, vrací hodnotu 1 (TRUE). Pokud je v proměnné normální hodnota, funkce vrací hodnotu 0 (FALSE).

```
MIS(var)  
COMP varA = MIS(varB).
```

SYSMIS(var)

Pokud je hodnota argumentu SYSTEM MISSING, vrací hodnotu 1 (TRUE), jinak vrací hodnotu 0 (FALSE).

```
SYS(var)  
COMP varA = SYS(varB).
```

NMISSING(vars)

Vrací počet proměnných, které mají hodnotu MISSING.

```
NMIS(vars)  
NMIS(var1, var2, var3).
```

NVALID(vars)

Vrací počet proměnných, které mají validní hodnotu.

```
NVAL(vars)  
NVAL(var1, var2, var3).  
NMIS(vars) + NVAL(vars) = počet vars
```

POROVNÁVACÍ FUNKCE

Jde o funkce, které specifickým způsobem porovnávají hodnoty argumentů.

ANY(seznam_argumentů)

Vrací hodnotu TRUE, jestliže první argument je roven jednomu z následujících argumentů. Argumenty mohou být proměnné nebo hodnoty.

ANY(argA, arg1, arg2, arg3)

ANY(var1, 5, 10, 65)

(porovná, jestli var1 je rovna jedné z hodnot 5,10,65)

ANY(10, var1, var2, var3)

(porovná, jestli některá z proměnných var1 až var3 je rovna 10)

ANY(varX, var1, var2, var3)

(porovná, jestli některá z proměnných var1 až var3 je rovna varX)

RANGE(seznam_argumentů)

Vrací hodnotu TRUE, jestliže první argument nabývá hodnoty z intervalů definovaných následujícími dvojicemi argumentů. Celkový počet argumentů musí být lichý počet.

RANGE(argA, arg1, arg2, arg3, arg4)

RANGE(varA, 5, 10, 65, 80)

(porovná, jestli proměnná var1 obsahuje hodnotu z intervalu <5,10> nebo <65,80>)

RANGE(10, var1, var2, var3, var4)

RANGE(varA, var1, var2, var3, var4)

MATEMATICKÉ FUNKCE

MOD (var1, var2).	(zbytek po celočíselném dělení argumentu var1 argumentem var2)
SQRT (var1).	(odmocnina)
ABS (var1).	(absolutní hodnota)
EXP (var1).	(mocnina čísla e umocněná argumentem v závorce)
LG10 (var1).	(dekadický logaritmus)
LN (var1).	(přirozený logaritmus)
SIN (var1).	(sinus)
COS (var1).	(cosinus)
ARSIN (var1).	(arcus sinus)
ARCOS (var1).	(arcus cosinus)
RND (var1).	(zaokrouhlení na nejbližší celé číslo: -4.3→-4; -4.7→-5; 4.3→4)
TRUNC (var1).	(oříznutí desetinné části)

Pokud argument, který vstupuje do aritmetických funkcí, nabývá hodnoty MISSING, pak je výsledkem hodnota MISSING. Výjimkou jsou případy $0 * \text{MISSING}$, $0 / \text{MISSING}$ a $\text{MOD}(0, \text{MISSING})$, které jsou vyhodnoceny jako 0 (nula).

STATISTICKÉ FUNKCE

SUM (var1 to var5, var10).	(suma argumentů)
SUM.M (var1, ..., varN).	
(suma argumentů, za podmínky, že jich alespoň M není MISSING)	
MEAN (var1, ..., varN).	(průměr argumentů)
VARIANCE (var1, ..., varN).	(rozptyl)
SD (var1, ..., varN).	(směrodatná odchylka)
MIN (var1, ..., varN).	(minimální hodnota)
MAX (var1, ..., varN).	(maximální hodnota)

Do statistických operací mohou vstupovat argumenty s MISSING hodnotou. Tyto funkce MISSING hodnoty vynechají ze zpracování. Musí však obsahovat alespoň 1 hodnotu, která není MISSING.

STRINGOVÉ FUNKCE

UPCASE (var) .	(převod na velká písmena)
LOWER (var) .	(převod na malá písmena)
RTRIM (var) .	(ořeže mezery z pravé strany řetězce)
LTRIM (var) .	(ořeže mezery z levé strany řetězce)
LENGTH (var) .	(délka řetězce v proměnné var, včetně mezer na začátku i konci)
INDEX (var, text) .	(vrátí pozici prvního výskytu řetězce text v proměnné var)
RINDEX (var, text) .	(vrátí pozici posledního výskytu řetězce text v proměnné var)
CONCAT (var1, var2) .	(spojí dva řetězce, nová proměnná musí mít dostatečnou délku)
REPLACE (var, text_old, text_new, number) .	
(v proměnné var nahradí řetězec text_old, řetězcem text_new;)	
(parametr number, který lze vynechat, určí kolik výskytů řetězce text_old se má nahradit)	
NUMBER (var1, formát) .	(převede str. prom. na číselnou v zadaném číselném formátu)
STRING (var1, formát) .	(převede num. prom. na textovou v zadaném číselném formátu)
SUBSTR (var, N1) .	(z proměnné var od pozice N1 vezme všechny znaky)
SUBSTR (var, N1, N2) .	(z proměnné var od pozice N1 vyřízne N2 znaků)

Pokud chceme přiřadit textové proměnné nějakou hodnotu (řetězec), musí být tato proměnná vytvořena předem (např. pomocí příkazu **STRING**). Hodnoty přiřazované textovým proměnným musí být v uvozovkách. Textová proměnná musí mít předem nastavenou délku. Pokud je v této proměnné řetězec kratší, než je délka proměnné, je doplněn mezerami.

DATUMOVÉ FUNKCE

XDATE.YEAR (var) .	(vrací rok)
XDATE.JDAY (var) .	(vrací počet dnů od začátku roku)
TIME.HMS (h, m, s) .	(vytváří hodnotu ve formátu času)
DATE.DMY (d, m, y) .	(vytváří hodnotu ve formátu data)

Proměnné, které použijeme při práci s těmito funkcemi, musí být typu DATE.

OSTATNÍ FUNKCE

LAG (var, N) .	(hodnota v proměnné var o N řádků výše)
UNIFORM (N) .	(pseudonáhodné číslo mezi 0 a číslem N)

PŘEHLED PŘÍKAZŮ – ŘÍDÍCÍ

Jde o příkazy, které řídí tok jiných příkazů. Jedná se primárně o příkazy podmínkové a o příkazy pro cykly.

IF

Příkaz za nímž následuje logická podmínka a pak přiřazení hodnoty do proměnné. K přiřazení hodnoty do proměnné dojde v případě, že logická podmínka za příkazem IF je vyhodnocena jako pravda (TRUE).

```
IF (logický_výraz) var = funkce(variables).
```

Př1: pokud je splněno, že hodnota v proměnné varA je 1, pak do proměnné varB je přiřazena hodnota z proměnné varC.

```
IF (varA=1) varB=varC.
```

Př2: pokud je splněno, že hodnota v proměnné varA je rovna hodnotě v proměnné varB, pak do proměnné varC zapíše hodnotu 1.

```
IF (varA=varB) varC=1.
```

Př3: pokud platí, že v proměnné varA je hodnota větší než 3, pak do proměnné varX vepíše součet hodnot proměnných varY a varZ.

```
IF (varA>3) varX=SUM(varY,varZ).
```

Př4: pokud v proměnné varA chybí hodnota (tj. její hodnota je SYSMIS), pak do této proměnné zapíše vyhodnocení logického výrazu (varY=varZ). Tzn., že pokud jsou si proměnné varY a varZ rovny, tak do varA zapíše hodnotu 1 (ta představuje vyhodnocení TRUE – pravda), pokud se tyto proměnné nerovnají, pak do proměnné varA zapíše hodnotu 0 (ta představuje vyhodnocení FALSE – nepravda)

```
IF (sys(varA)) varA=(varY=varZ).
```

DO IF

Příkaz za nímž následuje logická podmínka. Po tomto příkazu může následovat blok libovolně mnoha jiných příkazů a pak se tento blok uzavře příkazem END IF. K vykonání bloku příkazů, které následují za příkazem DO IF, dojde v případě, že logická podmínka za příkazem DO IF je vyhodnocena jako pravda (TRUE).

```
DO IF (logický_výraz).  
  Blok příkazů.  
ELSE IF (logický_výraz).  
  Blok příkazů.  
ELSE.  
  Blok příkazů.  
END IF.
```

Příkazy ELSE IF a ELSE je možné vynechat. V této kompletní struktuře vykonávání příkazů probíhá následujícím způsobem: Nejprve se vyhodnotí logický výraz v příkaze DO IF. Pokud je vyhodnocen jako TRUE, dojde k vykonání bloku příkazů a pak vykonávání skočí až na END IF a celé DO IF je ukončeno. Pokud výraz v DO IF je vyhodnocen jako FALSE, pak se blok příkazů za ním nevykoná a vyhodnotí se logický výraz za příkazem ELSE IF. Podobně jako u DO IF, dojde nebo nedojde k vykonání následujícího bloku příkazu podle toho, jestli vyhodnocením je TRUE nebo FALSE. Příkaz ELSE IF se může libovolněkrát opakovat. Pokud logický výraz v DO IF a současně ve všech ELSE IF je vyhodnocen jako FALSE, pak dojde k vykonání bloku příkazů, které následují za příkazem ELSE.

```
Př1: DO IF (varA=1).  
  COMP varB = 1.  
  RECODE varC (ELSE=SYS).  
END IF.
```

```
Př2: DO IF (varA=1).  
  COMP varB=varC.  
ELSE IF (varA=2).  
  COMP varB=varD.  
ELSE.  
  COMP varB=0.  
END IF.
```

LOOP

Příkaz pro cyklus, který probíhá buď v zadaném počtu kroků, nebo probíhá tak dlouho dokud nejsou splněny zadané podmínky v logických výrazech.

```
LOOP #I = N1 TO N2 BY N3 IF (logický_výraz).  
    Blok příkazů.  
END LOOP IF(logický_výraz).
```

Počet kroků cyklu je řízen buď stanovením počtu kroků v pomocné proměnné (např. #I) nebo může mít podmínku na začátku cyklu nebo může mít podmínku na konci cyklu a nebo může mít kombinaci všech částí, které řídí počet kroků cyklu. U cyklů s podmínkami je potřeba dát si pozor, aby se hodnoty proměnných, které jsou v těchto logických podmínkách, měnily uvnitř cyklu tak, aby mohlo dojít k ukončení cyklu poci této podmínky.

Př1: Cyklus s podmínkou na začátku. Blok příkazů mezi LOOP IF a END LOOP je vykonáván tak dlouho, dokud je podmínka TRUE (tj. dokud platí, že hodnota v proměnné varA je menší než 10). Na začátku se vyhodnocuje, jestli je podmínka TRUE, pokud ano, tak se vykoná blok příkazů, pak se cyklus vrátí na začátek a znovu vyhodnocuje podmínka atd.

```
LOOP IF (varA<10).  
    Blok příkazů.  
END LOOP.
```

Př2: Cyklus s podmínkou na konci. Blok příkazů mezi LOOP IF a END LOOP je vykonáván tak dlouho, dokud je podmínka FALSE (tj. dokud platí, že hodnota v proměnné varA není větší než 10). Nejprve se vykoná blok příkazů a pak se vyhodnocuje, jestli je podmínka FALSE, pokud ano, tak se opět vykoná blok příkazů atd. V momentě, kdy je podmínka vyhodnocena jako TRUE, dojde k ukončení cyklu.

```
LOOP.  
    Blok příkazů.  
END LOOP IF (varA>10).
```

Př3: Cyklus s podmínkou na začátku i na konci. Blok příkazů mezi LOOP IF a END LOOP je vykonáván tak dlouho, dokud je podmínka na začátku TRUE a současně podmínka na konci FALSE. Na začátku se vyhodnocuje, jestli je podmínka TRUE, pokud ano, tak se vykoná blok příkazů, pak se vyhodnotí podmínka na konci, pokud je FALSE, jde na začátek a vyhodnocuje se podmínka na začátku atd.

```
LOOP IF (varA<10).  
    Blok příkazů.  
END LOOP IF (varB=2).
```

Př4: Cyklus s předem definovaným počtem 10 opakování.

```
SET MXLOOPS = 10.  
LOOP.  
    Blok příkazů.  
END LOOP.
```

Př5: Cyklus s počtem opakování definovaným pomocí dočasné proměnné #I. Cyklus proběhne celkem 5x a v jednotlivých opakováních bude proměnná #I nabývat postupně hodnot 1,2,3,4,5.

```
LOOP #I = 1 TO 5.  
    Blok příkazů.  
END LOOP.
```

Př6: Cyklus s počtem opakování definovaným pomocí dočasné proměnné #I a současně omezený podmínkou na začátku cyklu.

```
LOOP #I = 1 TO 5 IF (var1 < 5).  
    Blok příkazů.  
END LOOP.
```

Př7: Cyklus s počtem opakování definovaným pomocí dočasné proměnné #I a současně omezený podmínkou na konci cyklu. V tomto případě se hodnota dočasné proměnné #I zvyšuje o 2, takže nabývá postupně hodnot 2,4,6,8,10.

```
LOOP #I = 2 TO 10 BY 2.  
    Blok příkazů.  
END LOOP IF (var1 > 5).
```

DO REPEAT

Příkaz pro cyklus, který se používá v případě, že stejnou operaci chceme provést pro více proměnných nebo skupin proměnných.

```
DO REPEAT
  virtual_var1 = argument_list1 /
  ...
  virtual_varN = argument_listN .
  Blok příkazů.
END REPEAT.
```

Proměnné, případně hodnoty, se přiřazují do virtuálních proměnných. Operace se zapíše pomocí těchto virtuálních proměnných. Fyzicky se pak operace v jednotlivých opakováních vykonávají postupně se všemi argumenty. Všechny virtuální proměnné musí mít stejný počet položek v příslušném seznamu argumentů. Uvnitř cyklu DO REPEAT nelze mít další cyklus DO REPEAT a také některé další příkazy (např. VAR LAB).

Př1: Do všech proměnných var1 až var5 se postupně přiřadí hodnota 1.

```
DO REPEAT
  var = var1 TO var5.
  COMP var = 1.
END REPEAT.
```

Př2: Do proměnných var1 až var5 se postupně přiřadí hodnoty 1,2,3,4,5.

```
DO REPEAT
  var = var1 TO var5 /
  varN = 1 TO 5 .
  COMP var = varN.
END REPEAT.
```

Př3: Do proměnných var1 až var5 se postupně přiřadí hodnoty 1,2,3,4,5 a do proměnných var6 až var10 se postupně přiřadí hodnoty 11,12,13,14,15.

```
DO REPEAT
  var = var1 TO var5, var6 TO var10 /
  varN = 1,2,3,4,5,11,12,13,14,15 .
  COMP var = varN.
END REPEAT.
```

VECTOR

Příkaz pomocí něhož lze vytvořit virtuální vektor proměnných.

```
VECTOR vec = var1 TO varN.
```

Proměnné, které přiřazujeme do vektoru musí být v datovém souboru seřazeny za sebou. Proměnné lze do vektoru vložit pouze zadáním rozsahu (var1 TO var3), nelze použít výčet (var1,var2,var3). K jednotlivým proměnným ve vektoru přistupujeme pomocí indexu (tj. pořadí proměnné ve vektoru). Nejčastější využití vektoru je ve spojení s cyklem LOOP, kdy indexem pro přístup k jednotlivým proměnným je používána dočasná proměnná, která v cyklu LOOP definuje počet opakování.

Př: Proměnné var1 až var10 se načtou do vektoru. V příkazu LOOP, který se bude opakovat 10 krát (tj. pro každou proměnnou vektoru), se do každé proměnné načte mocnina indexu.

```
VECTOR vec = var1 to var10.
LOOP #I = 1 TO 10.
  COMP vec(#I)= #I*#I.
END LOOP.
```

Spuštění příkazu EXE, nebo libovolného statistického příkazu ukončí platnost vektoru.

BREAK

Příkaz pomocí něhož lze ukončit cyklus LOOP. Používá se v případech, které nejde ošetřit pomocí podmínek na začátku nebo konci cyklu LOOP.

BREAK.

Př: Proměnné var1 až var10 se načtou do vektoru. Příkazu LOOP se bude opakovat 10x. Pokaždé se provede součet proměnných varX a vec(#I). Pokud by však v některé proměnné ve vektoru byla hodnota SYSMIS, dojde k ukončení cyklu příkazem BREAK.

```
VECTOR vec = var1 to var10.  
LOOP #I = 1 TO 10.  
  DO IF (sys(vec(#i))).  
    BREAK.  
  END IF.  
  COMP varX = varX + vec(#I).  
END LOOP.
```

PŘEHLED PŘÍKAZŮ – STRUKTURÁLNÍ

Těmito příkazy se primárně nastavuje struktura proměnných. Jedná se o atributy, které jsou vidět v záložce Variable View.

MANIPULACE S PROMĚNNÝMI

RENAME VARIABLE

Příkaz pro přejmenování proměnných. Proměnné zůstanou na své pozici a budou mít i stejný formát. Pokud nové jméno je jménem existující proměnné, SPSS zahlásí chybu a příkaz neprovede.

```
REN VAR (VarList1_Old = VarList1_New) ... (VarListN_Old = VarListN_New).
```

Př1: Každou proměnnou přejmenováváme v samostatné závorce.

```
REN VAR (var1_old = var1_new)(var2_old = var2_new).
```

Př2: Nyní přejmenováváme seznam proměnných najednou v jedné závorce.

```
REN VAR (var1_old var2_old = var1_new var2_new).
```

Př3: Kombinace předchozích dvou příkladů.

```
REN VAR (varA_old = varA_new)(var1_old var2_old = var1_new var2_new).
```

DELETE VARIABLE

Příkaz pro odstranění proměnných. Před spuštěním tohoto příkazu musí být ukončeny transformační příkazy příkazem EXE nebo jiným možným příkazem. Pokud by nebyly ukončeny, tak se tento příkaz neprovede.

```
DEL VAR VarList.
```

Př1: Jedná se o jednoduchý příkaz, kde dojde k vymazání všech proměnných v definovaném seznamu.

```
DEL VAR var1 var2 ... varN.
```

SORT VARIABLES

Příkaz pro seřazení proměnných podle hodnot zvolených atributů proměnných.

```
SORT VAR BY attribute (D).
```

Implicitně (tj. bez parametru v závorce) jsou proměnné řazeny vzestupně. Pokud bychom to potřebovali sestupně, musíme do závorky zapsat písmeno D.

Př1: Seřadí proměnné podle jména.

```
SORT VAR BY name.
```

Př2: Seskupí proměnné podle typu (String/Numeric) a textové ještě seřadí podle definované šířky.

```
SORT VAR BY type.
```

FORMÁT PROMĚNNÝCH

FORMATS

Tímto příkazem můžeme nastavit formát existujících numerických proměnných. Formát textových proměnných tímto způsobem nastavit nelze.

```
FORMATS var_list1 (format1) var_list2 (format2).
```

V závorce zadáváme formát proměnné.

Př1: Proměnná varX bude mít 5 cifer a z toho budou 2 pro desetinné místo.

```
FORMATS varX (F5.2).
```

Př2: Ukázka nastavení formátu více proměnných najednou.

```
FORMATS varX varY varZ (F5.2) varA varB (date).
```

ALTER TYPE

Tímto můžeme libovolně měnit typ a formát proměnných.

```
ALTER TYPE var_list1 (format_old=format_new).
```

Př1: Všechny proměnné ze seznamu var_list, které mají formát F2.0 se převedou na formát A2.

```
ALTER TYPE var_list (F2.0 = A2).
```

Př2: U všech proměnných varX varY varZ se nastaví formát A2, bez rozdílu toho, jaký formát měly.

```
ALTER TYPE varX varY varZ (A2).
```

NUMERIC

Takto vytvoříme novou prázdnou numerickou proměnnou v zadaném formátu. Použití je stejné jako u příkazu FORMATS.

```
NUMERIC var_list1 (format1) var_list2 (format2).
```

Př1: Ukázka vytvoření více proměnných najednou s různými formáty.

```
NUMERIC varX varY varZ (F5.2) varA varB (date).
```

STRING

Takto vytvoříme novou prázdnou textovou proměnnou v zadaném formátu.

```
STRING var_list1 (format1) var_list2 (format2).
```

Př1: Ukázka vytvoření více proměnných najednou s různými formáty. Proměnné varX varY varZ budou textové proměnné, které budou mít 20 znaků a proměnné varA varB budou textové proměnné, které budou mít 50 znaků.

```
STRING varX varY varZ (A20) varA varB (A50).
```

POPISKY PROMĚNNÝCH

VARIABLE LABEL

Příkaz pro popis proměnných. Maximální délka popisku je 250 znaků.

```
VAR LAB varX "Popis proměnné".
```

Př1: Každou proměnnou popisujeme samostatně.

```
VAR LAB varX "Popis proměnné varX".
```

Př2: Popis více proměnných v jednom příkaze. Výhodou je, že nemusíme opakovaně psát příkaz VAR LAB.

```
VAR LAB
  varX "Popis proměnné varX"
  varY "Popis proměnné varY"
  varZ "Popis proměnné varZ".
```

VALUE LABEL

Příkaz pro popis hodnot proměnných. Délka popisku může být maximálně 120 znaků. Pokud již proměnná měla nějaké popisky, pak spuštěním tohoto příkazu budou původní popisky nahrazeny.

```
VAL LAB
  varA1 ... varAx 1 "Popis hodnoty 1" ... N "Popis hodnoty N" /
  ...
  varZ1 ... varZx 1 "Popis hodnoty 1" ... N "Popis hodnoty N" .
```

Př1: Popisky pro jednu proměnnou.

```
VAL LAB varX 1 "Ano" 2 "Ne".
VAL LAB varX
  1 "Ano"
  2 "Ne".
```

Př2: Popisky pro více proměnných, které mají stejné popisky.

```
VAL LAB varX1 varX2 varX3
  1 "Ano"
  2 "Ne".
```

Př3: Popisky více proměnných v jednom příkaze, kde některé mají stejné popisky a některé ne..

```
VAL LAB
  varX1 varX2 varX3 1 "Ano" 2 "Ne" /
  varA 1 "Ano" 2 "Ne" 99 "Nevím" /
  varB1 varB2 1 "Modrá" 2 "Červená" .
```

ADD VALUE LABEL

Příkaz pro popis hodnot proměnných. Narozdíl od klasického VAL LAB, tento příkaz zachová popisky hodnot, které nejsou v tomto příkaze definovány.

```
ADD VAL LAB
  varA1 ... varAx 1 "Popis hodnoty 1" ... N "Popis hodnoty N" /
  ...
  varZ1 ... varZx 1 "Popis hodnoty 1" ... N "Popis hodnoty N" .
```

Př1: Nejprve klasickým VAL LAB popíšeme hodnoty 1, 2. a 99. Poté dalším příkazem ADD VAL LAB doplníme do stejné proměnné upravený popis hodnoty 99 a navíc popisek hodnoty 98. Proměnná pak bude mít popsány všechny hodnoty 1, 2, 98 a 99. Pokud bychom i podruhé použili pouze příkaz VAL LAB, pak by proměnná měla popsán pouze hodnoty 98 a 99 a hodnoty 1 a 2 by byly bez popisků.

```
VAL LAB varX 1 "Ano" 2 "Ne" 99 "Bla bla".
ADD VAL LAB varX 98 "Nevím" 99 "Odpověď odmítnuta".
```

DALŠÍ ATRIBUTY PROMĚNNÝCH

MISSING VALUE

Slouží k nadefinování uživatelských chybějících hodnot (tzv. USER MISSING). Tyto hodnoty pak nebudou zahrnovány do statistických výstupů.

```
MIS VAL var_list1 (values1) ... var_listN (valuesN).
```

Př1: U proměnných varX a varY je 99 nastaveno jako chybějící hodnota, u proměnné varZ jsou chybějící hodnoty všechny hodnoty menší nebo rovny nule a hodnota 99.

```
MIS VAL varX varY (99) varZ (LO thru 0, 99).
```

Př2: Pokud závorku necháme prázdnou, tak dojde ke zrušení nastavených chybějících hodnot.

```
MIS VAL varX varY varZ ( ).
```

VARIABLE ALIGNMENT

Slouží k nadefinování zarovnání při zobrazování v záložce Data View.

```
VAR ALI var_list1 (align1) ... var_listN (alignN).
```

Př: Ukázka všech možných typů zarovnání. Proměnnou varX zarovná do leva, varY do prava a varZ na střed.

```
VAR ALI varX (LEFT) varY (RIGHT) varZ (CENTER).
```

VARIABLE LEVEL

Slouží k nadefinování typu otázky.

```
VAR LEV var_list1 (level1) ... var_listN (levelN).
```

Př: Ukázka všech možných typů. Proměnná varX je typu SCALE (s měřitelným pořadím, např. věk v letech, příjem v korunách), varY typu NOMINAL (bez měřitelného pořadí, např. region, pohlaví) a varZ typu ORDINAL (kategorické otázky, např. škála spokojenosti).

```
VAR ALI varX (SCALE) varY (NOMINAL) varZ (ORDINAL).
```

VARIABLE WIDTH

Slouží k nadefinování šířky sloupce při zobrazování v záložce Data View.

```
VAR WID var_list1 (width1) ... var_listN (widthN).
```

Př: U proměnných varX a varY je šířka sloupce 10, u proměnné varZ je šířka sloupce 20.

```
VAR WID varX varY (10) varZ (20).
```

PŘEHLED PŘÍKAZŮ – SOUBOROVÉ

OTEVÍRÁNÍ / UKLÁDÁNÍ DATOVÝCH SOUBORŮ

CD

Slouží k definici pracovního adresáře. Pokud je takto definován pracovní adresář, je možno v příkazech GET a SAVE používat relativní cesty k souborům.

```
CD "C:/SPSS/DATA/".
```

GET

Slouží k definici datového souboru, který bude otevřen a aktivován pro další úpravy.

```
GET FILE = "file.sav".
```

Standardně musí být pro definici souboru, který chceme otevřít, použita absolutní cesta. Pokud chceme používat relativní cesty, můžeme si pomoci příkazem CD.

Možnými parametry příkazu jsou:

```
FILE "file.sav" – povinný parametr, kterým definujeme soubor, který má být otevřen  
/KEEP var_list – při otevírání datového souboru vybere pouze proměnné definované v tomto seznamu  
/DROP var_list – při otevírání datového souboru vyhodí proměnné definované v tomto seznamu  
/RENAME (var_list_old = var_list_new) – při otevírání přejmenuje příslušné proměnné
```

Parametry KEEP a DROP nelze použít současně. Buď definuji, co vybrat, nebo co nevybrat.

Př: Otevřeme soubor file.sav, přitom vyhodíme proměnnou varX a proměnné varA1 a varA2 přejmenujeme na varB1 a varB2.

```
GET FILE = "C:/file.sav"  
/DROP varX  
/RENAME (varA1 varA2 = varB1 varB2).
```

GET DATA

Slouží k otevření dat v jiném formátu než je SPSS.

```
GET DATA  
/TYPE = type  
/FILE= "file.sav".
```

Otvírá formáty ODBC, OLEDB, XLS (verze 95 – 2003), XLSX, XLSM, TXT.

Př: Otevřeme soubor file.xls a načteme data z listu sheet1.

```
GET DATA  
/TYPE = XLS  
/FILE = "C:/file.xls"  
/SHEET = NAME "sheet1".
```

GET TRANSLATE

Slouží k otevření dat v jiném formátu než je SPSS. Oproti GET DATA otvírá spíše starší datové typy.

```
GET TRANSLATE FILE = "file.sav"  
/TYPE = type.
```

Otvírá formáty WK,WK1,WKS,SYM,SLK, XLS (verze nižší než 95), DBF, TAB, SYS.

Př: Otevřeme soubor file.xls a načteme data z listu sheet1.

```
GET DATA  
/TYPE = XLS  
/FILE = "C:/file.xls"  
/SHEET = NAME "sheet1".
```

SAVE

Slouží k definici datového souboru, do kterého budou uložena aktuální data.

```
SAVE OUTFILE = "file.sav".
```

Standardně musí být pro definici souboru, který chceme otevřít, použita absolutní cesta. Pokud chceme používat relativní cesty, můžeme si pomoci příkazem CD.

Možnými parametry příkazu jsou:

OUTFILE "file.sav" – povinný parametr, kterým definujeme soubor, do kterého budou uložena data

/KEEP var_list – při ukládání dat vybere pouze proměnné definované v tomto seznamu

/DROP var_list – při ukládání dat vyhodí proměnné definované v tomto seznamu

/RENAME (var_list_old = var_list_new) – při ukládání přejmenuje příslušné proměnné

Parametry KEEP a DROP nelze použít současně. Buď definuji, co vybrat, nebo co nevybrat.

Př: Data ukládáme do souboru file.sav, přitom vyhodíme proměnnou varX a proměnné varA1 a varA2 přejmenujeme na varB1 a varB2.

```
SAVE OUTFILE = "C:/file.sav"  
  /DROP varX  
  /RENAME (varA1 varA2 = varB1 varB2).
```

SAVE TRANSLATE

Slouží k uložení dat v jiném formátu než je SPSS.

```
SAVE TRANSLATE OUTFILE = "file"  
  /TYPE = type.
```

Ukládá formáty CSV, DB2, DB3, DB4, ODBC, PC, SAS, STATA, SYM, SLK, TAB, WKS, WK1, WK3, XLS.

Př: Otevřeme soubor file.xls a načteme data z listu sheet1.

```
GET DATA  
  /TYPE = XLS  
  /FILE = "C:/file.xls"  
  /SHEET = NAME "sheet1".
```

WRITE

Slouží k zapsání dat v textovém formátu (tzv. strojově čitelném formátu). Je užitečný především pro vytváření ASCII datových souborů, kdy u každé proměnné samostatně definujeme její výstupní formát. Každá proměnná musí mít formát nadefinovaný samostatně. Výhodou je, že můžeme jednu proměnnou zapsat i vícekrát a to klidně v různých formátech.

```
WRITE OUTFILE = "output_file.txt"  
  / var1 (format1) ... varN (formatN).
```

Př: Do souboru file.txt zapíše nejprve proměnnou var1 ve formátu N5, pak znovu proměnnou var1 ve formátu F5 a pak proměnnou var2 ve formátu A10.

```
WRITE OUTFILE = "file.txt"  
  /var1 (N5) var1 (F5) var2 (A10).
```

Formáty a vlastnosti zápisu na příkladu

- N5.2:** Zapíše číselnou proměnnou na 5 pozic, z toho 3 pozice jsou pro celou část a 2 pozice pro desetinnou část. Desetinný oddělovač se nezapíše. Pokud má celá část čísla méně cifer, tak nejprve zapíše nuly a pak teprve celou část. Pokud má desetinná část méně cifer, pak zbylé doplní nulami. Takže číslo 2.3 zapíše takto 00230. Pokud má celá část čísla více cifer, pak místo každé cifry zapíše hvězdičku. Pokud má desetinná část více cifer, pak ji zaokrouhlí. Takže číslo 123.456 zapíše takto 12346, ale číslo 1234.5 zapíše takto *****.
- F5.2:** Zapíše číselnou proměnnou na 5 pozic, z toho 2 pozice případně použije pro desetinnou část. Desetinný oddělovač se zapisuje a taky ubírá jednu pozici. Pokud má celá část čísla méně cifer, tak nejprve zapíše mezery a pak teprve celou část. Pokud má desetinná část méně cifer, pak zbylé doplní nulami. Takže číslo 2.3 zapíše takto -2.30. Pokud má celá část čísla více cifer než 5, pak místo každé cifry zapíše hvězdičku. Pokud má desetinná část více cifer, pak ji zaokrouhlí. Takže číslo 123.456 zapíše takto 123.5, ale číslo 1234.5 zapíše takto -1234, ale číslo 123456 zapíše takto *****.
- A5:** Zapíše textovou proměnnou na 5 pozic. Pokud by text byl kratší než na 5 pozic, zapíše nejprve text a na konci ho doplní mezerami. Pokud by text byl delší než na 5 pozic, zapíše prvních 5 znaků a zbytek usekne.

SPOJOVÁNÍ DATOVÝCH SOUBORŮ

ADD FILES

Spojuje soubory tak, že k jednomu souboru přidává řádky z dalšího. Pokud v některém souboru chybí číselné proměnné, které jsou v jiném, tak se v této proměnné přiřadí řádkům z tohoto souboru hodnota SYSMIS. Pokud mají stejné číselné proměnné ve spojovaných souborech jiný formát, použije se formát z prvního spojovaného souboru. Pokud však mají jiný formát textové proměnné, vypíše to chybu a operace neproběhne.

```
ADD FILES
  /FILE = file1.sav
  ...
  /FILE = fileN.sav.
```

Př: Hvězdička místo prvního souboru znamená, že při spojování se jako první použije právě aktivní soubor. Takže nám příkaz spojí aktivní soubor se soubory fileX.sav a fileY.sav.

```
ADD FILES
  /FILE = *
  /FILE = "fileX.sav"
  /FILE = "fileY.sav".
```

MATCH FILES

Spojuje soubory tak, že k jednomu souboru přidává proměnné z dalšího. Pro správné přiřazení hodnot odpovídajícím řádkům musí být v každém souboru jedna společná proměnná, podle které musí být řádky vzestupně seřazeny. Podle této proměnné se pak soubory spojují. Pokud se pokusíme připojit soubor s proměnnou, která už je v předchozích souborech obsažena, tak v této proměnné zůstanou původní hodnoty, nepřepíší se hodnotami z připojovaného souboru.

```
MATCH FILES
  /FILE = file1.sav
  ...
  /FILE = fileN.sav
BY var.
```

Př: Hvězdička místo prvního souboru znamená, že při spojování se jako první použije právě aktivní soubor. Takže nám příkaz k aktivnímu souboru připojí proměnné ze souborů fileX.sav a fileY.sav a jako propojovací proměnnou použije idnumber.

```
MATCH FILES
  /FILE = *
  /FILE = "fileX.sav"
  /FILE = "fileY.sav"
BY idnumber.
```

TRANSFORMACE DATOVÉHO SOUBORU

SORT CASES

Setřídí řádky v datovém souboru podle hodnot proměnných uvedených v seznamu proměnných. Nejprve třídí podle první uvedené proměnné, pak podle druhé uvedené proměnné, atd.

```
SOR CAS var_list1 (D) ... var_listN (A).
```

Volba (A) nebo (D) určuje, jak se podle seznamu proměnných, za kterými je tato volba uvedena, má třídít. Volba (A) znamená vzestupně (ascending), volba (D) znamená sestupně (descending).

Př1: Setřídí řádky vzestupně podle hodnot proměnné var1. V tomto případě jsme mohli vynechat parametr (A), protože ten je nastavený implicitně.

```
SOR CAS var1.
```

Př2: Budou vybrány všechny interview, které splňují podmínku, že proměnná varX je rovna hodnotě 1.

```
SEL IF (varX = 1).
```

STRUKTURA DATOVÉHO SOUBORU

Nejčastěji se pomocí těchto příkazů nastavují data pro statistické výstupy. Provádějí se např. výběry částí interview z datového souboru, případně se data váží apod.

SELECT IF

Slouží k výběru řádků, které splňují podmínku nadefinovanou v závorce. Výběr je trvalý, tzn. že řádky, které nebyly vybrány, jsou smazány.

```
SEL IF (podmínka).
```

Př: Budou vybrány všechny interview, které splňují podmínku, že proměnná varX je rovna hodnotě 1.

```
SEL IF (varX = 1).
```

N OF CASES

Slouží k výběru zadaného počtu interview ze začátku datového souboru. Výběr je, podobně jako u SEL IF, trvalý.

```
N number.
```

Př: Z datového souboru bude vybráno prvních 59 interview.

```
N 59.
```

SAMPLE

Slouží k náhodnému výběru zadaného počtu interview ze začátku datového souboru. Jsou 2 varianty použití tohoto příkazu. Výběr je také trvalý.

V prvním případě volíme desetinné číslo z intervalu (0,1), které nám určuje, kolik procent interview má být vybráno.

```
SAMPLE decimal_number.
```

V druhém případě zadáváme konkrétní počet number1 interview, který má být vybrán a velikost souboru, ze kterého má být vybíráno, parametrem number2. Pokud je parametr number2 menší než je celkový počet interview, tak je výběr proveden z prvních number2 interview. Pokud je parametr number2 větší, než je celkový počet interview, tak je počet vybíraných interview proporcionálně snížen.

```
SAMPLE number1 FROM number2.
```

Př1: Z datového souboru bude náhodně vybráno 65% ze všech interview.

```
SAMPLE 0.65.
```

Př2: Pokud má datový soubor 200 interview, pak z něj bude náhodně vybráno 50 interview. Pokud by měl více než 200 interview, tak těchto 50 interview je vybráno z prvních 200. Pokud by měl např. pouze 100 interview, pak by bylo vybráno pouze 25.

```
SAMPLE 50 FROM 200.
```

FILTER

Slouží k výběru řádků na základě hodnot uvedené proměnné (tzv. filtrovací proměnné). Jedná se o výběr řádků, které mají být zahrnuty do statistických výstupů. Transformační příkazy jsou i nadále prováděny na všech řádcích. Tento výběr je dočasný, na rozdíl od předchozích příkazů. Výběr je platný tak dlouho, dokud není vypnutý ukončovacím příkazem.

```
FILTER BY var_filter.  
FILTER OFF.
```

Správně vytvořená filtrovací proměnná by měla obsahovat pouze hodnoty 1 (řádky budou vybrány) a hodnoty 0 (řádky nebudou vybrány). Prakticky to funguje tak, že jsou vybrány všechny řádky, ve kterých má filtrovací proměnná nenulovou hodnotu (tj. množina $R - \{0\}$). Naopak všechny řádky, kde filtrovací proměnná má hodnotu 0 nebo SYSMIS, nebudou vybrány.

Př: Nejprve je zapnutý filtr podle proměnné var_filter. Pak se vyjede frekvence proměnné varX, pouze z řádků, které splňují podmínku filtru. Nakonec filtr vypínáme příkazem FILTER OFF.

```
FILTER BY var_filter.  
FRE varX.  
FILTER OFF.
```

SPLIT FILE

Slouží k rozdělení datového souboru na podskupiny se stejnými hodnotami v uvedené proměnné (tzv. splitovací proměnné). Rozdělení je platné tak dlouho, dokud není vypnuto ukončovacím příkazem.

```
SPLIT FILE BY var_split.  
SPLIT FILE OFF.
```

Před použitím tohoto příkazu **je nutné mít řádky seřazeny podle splitovací proměnné** příkazem SORT CASES, jinak rozdělení neproběhne korektně. Pozor, pokud zapneme SPLIT a pak znovu sortujeme, tak dojde k nekorektnímu ukončení platnosti rozdělení.

Př: Nejprve je zapnuto rozdělení podle proměnné var_split. Pak se vyjede frekvence proměnné varX, zvlášť pro jednotlivé podskupiny. Nakonec je rozdělení vypnuto příkazem SPLIT FILE OFF.

```
SPLIT FILE BY var_split.  
FRE varX.  
SPLIT FILE OFF.
```

WEIGHT

Slouží k navažování datového souboru podle zadané proměnné (tzv. vážící proměnné). Při statistických výjezdech pak odpověď konkrétního interview má takovou váhu, jaká je hodnota ve vážící proměnné. Prakticky je daná odpověď započítaná tolikrát, kolik je hodnota ve vážící proměnné. Vážení je platné tak dlouho, dokud není vypnuto ukončovacím příkazem.

```
WEIGHT BY var_weight.  
WEIGHT OFF.
```

Př: Nejprve jsou zapnuty váhy podle proměnné var_weight. Pak se vyjede vážená frekvence proměnné varX. Nakonec vážení vypínáme příkazem WEIGHT OFF.

```
WEIGHT BY var_weight.  
FRE varX.  
WEIGHT OFF.
```

DALŠÍ MANIPULACE SE SOUBORY

ERASE

Z pevného disku smaže definovaný soubor. Hodí se, když si potřebujeme vytvářet pomocné soubory, které po použití již nejsou potřeba

```
ERASE FILE = "file".
```

Př: Smaže soubor file.sav.

```
ERASE FILE = "C:\DATA\file.sav".
```

DATASET

Slouží k manipulaci s jednotlivými používanými datovými soubory. Pro různé manipulace jsou různé příkazy.

Nastavení souboru s příslušným jménem jako aktivního.

```
DATASET ACTIVATE name.
```

Zavření souboru s příslušným jménem.

```
DATASET CLOSE name.
```

Vytvoření nového souboru zkopírováním souboru s příslušným jménem.

```
DATASET COPY name.
```

Definování nového datového souboru.

```
DATASET DECLARE name.
```

Výpis všech dostupných (otevřených) datových souborů.

```
DATASET DISPLAY.
```

Pojmenování aktivního datového souboru zadaným jménem.

```
DATASET NAME name.
```

Př: Příklad, kdy potřebujeme v souboru zkopírovat proměnnou var_for_copy do nové proměnné var_copied, na což zatím není v SPSS příkaz.

```
DATASET CLOSE all.  
GET FILE = "fileX.sav" /KEEP ID var_for_copy.  
REN VAR (var_for_copy = var_copied).  
DATASET NAME copied.  
GET FILE = "fileX.sav".  
DATASET NAME master.  
DATASET ACTIVATE master.  
MATCH FILES  
  /FILE = master  
  /FILE = copied  
BY ID.  
EXE.
```

PŘEHLED PŘÍKAZŮ – STATISTICKÉ

FREQUENCIES

Základní frekvenční tabulka.

FRE variables.

Př. `FRE variables`
`/STATISTICS = STDDEV VARIANCE MINIMUM MAXIMUM MEAN MEDIAN SUM`
`/FORMAT = DFRE.`

Parametr STATISTICS zajistí, že se vypíše i další požadované statistiky. Nejčastěji to bývá MEAN (průměr), VARIANCE (rozptyl) případně ALL (všechny dostupné). Na tyto statistiky je však lepší použít příkaz DES nebo MEAN.

Parametr FORMAT definuje, jak má být tabulka seřazena. Nejužitečnější je DFRE (sestupně podle frekvencí). Automaticky je nastaveno AVAL (vzestupně podle kódů proměnné).

DESCRIPTIVES

Výstup popisných statistik.

DES variables.

Př. `DES variables`
`/STATISTICS = STDDEV VARIANCE MINIMUM MAXIMUM MEAN MEDIAN SUM.`

Parametr STATISTICS definuje, které statistiky chceme vyjet. Nejčastěji to bývá MEAN, VARIANCE případně ALL.

CROSSTABS

Jednoduchá křížová tabulka.

CRO variables by variables by variables by variables.

Př. `CRO varX by varY`
`/CELLS = COL.`

Parametr CELLS definuje charakter výstupu. Nejčastěji můžeme chtít COL (sloupcová procenta), ROW (řádková procenta) nebo ALL (všechny dostupné typy jako procenta, total, residua, atd.). Nejčastěji křížíme 2 proměnné, při křížení více proměnných začíná být tabulka nepřehledná.

MEAN

Základní jednoduchý výjezd průměrů.

MEAN variables.

Př. `MEAN variables`
`/CELLS = STDDEV VARIANCE MINIMUM MAXIMUM MEAN MEDIAN SUM.`

Parametr CELLS definuje, které statistiky chceme vyjet. Nejčastěji to bývá MEAN, VARIANCE případně ALL.

MULT RESPONSE

Frekvenční výstupy pro otázky s více možnými odpověďmi. Nevýhodou tohoto příkazu je, že neumožňuje seřadit odpovědi podle frekvence.

```
MULT RESP GROUPS = $var_multi "Label" (var1 ... varN (min_val,max_val))  
/FRE = $var_multi.
```

Př: Proměnné a1_1 až a1_30, ve kterých jsou zapsány kódy značek, které si respondent vybavil, seskupíme pomocí parametru GROUP do proměnné \$a1 (vždy se používá na začátku znak \$) a nastavíme, že zahrnuty budou pouze řádky obsahující odpovědi v intervalu 1 až 99 a pouze odpovědi z tohoto intervalu.

```
MULT RESP  
GROUPS = $a1 "Spontánní znalost značek" (a1_1 TO a1_30 (1,99))  
/FRE = $a1.
```

Parametrem GROPUS definujeme multi proměnné, které zastupují skupinu jednotlivých proměnných s odpověďmi na multi otázku.

Parametr FRE definuje, které multi proměnné se mají vyjet.