Kapitel 44

Funktionen

In mehreren Prozeduren bei SPSS können Sie zur Transformation bestehender sowie zum Erstellen neuer Variablen eine Reihe arithmetischer und statistischer Funktionen verwenden. Unter anderem stehen diese Funktionen in den Prozeduren *Variable berechnen*, *Werte in Fällen Zählen* und *Umkodieren* aus dem Menü *Transformieren* sowie in der Prozedur *Daten, Fälle auswählen* zur Verfügung. In diesem Kapitel wird die Bedeutung dieser Funktionen erläutert. Der folgende Abschnitt gibt einen nach inhaltlichen Aspekten gegliederten Überblick über die verfügbaren Funktionen, bevor anschließend sämtliche Funktionen in alphabetischer Reihenfolge aufgeführt und mit Beispielen erläutert werden.

44.1 Thematischer Überblick

Arithmetische Funktionen

ABS(*Zahl*) Liefert den absoluten Wert von *Zahl*.

ARSIN(*Zahl*) Ergibt den inversen Sinus von *Zahl*.

ARTAN(*Zahl*) Ergibt den inversen Tangens von *Zahl*.

COS(*Zahl*) Ergibt den Kosinus von *Zahl*.

EXP(*Zahl*) Potenziert den Wert e (≈2,718282) mit *Zahl*.

LG10(*Zahl*) Ergibt den Logarithmus von *Zahl* zur Basis 10.

LN(*Zahl*) Ergibt den natürlichen Logarithmus von *Zahl*.

MOD(*Zahl, Nenner*) Ergibt den bei einer Division von *Zahl* durch *Nenner* entstehenden Rest.

RND(*Zahl*) Rundet die angegebene *Zahl*.

SIN(*Zahl*) Ergibt den Sinus von *Zahl*.

SQRT(*Zahl*) Ergibt den Betrag der Quadratwurzel von *Zahl*.

SUM(*Zahl, Zahl* [*, Zahl,...*]) Gibt die Summe der angegebenen Parameter aus.

TRUNC(*Zahl*) Schneidet die Dezimalstellen von *Zahl* ab und gibt den verbleibenden ganzzahligen Wert aus.

Statistische Funktionen

CFVAR(*Zahl, Zahl* [, *Zahl,...*]) Errechnet den Variationskoeffizienten.

LAG(*Variable*) Liefert den Wert von *Variable*, der dem aktuellen Fall in der Datendatei vorausgeht.

LAG(*Variable, Fallzahl*) Liefert den Wert von Variable, der dem aktuellen Fall in der Datendatei um die angegebene *Fallzahl* vorausgeht.

MAX(*Wert, Wert*[,...]) Gibt den größten der angegebenen Werte aus.

MEAN(*Zahl, Zahl*[,...]) Ergibt das arithmetische Mittel der angegebenen Werte.

MEDIAN(*Zahl, Zahl*[,...]) Ermittelt den Median der angegebenen Werte.

MIN(*Wert, Wert*[,...]) Gibt den kleinsten Wert der angegebenen Werte aus.

SD(*Zahl, Zahl*[, *Zahl,...*]) Ergibt die Standardabweichung der Parameter.

VARIANCE(*Zahl, Zahl* [, *Zahl,...*]) Errechnet die Varianz der Parameter.

Verteilungs- und Zufallsfunktionen

CDF.Verteilung(*Zahl,...*) Ergibt die Wahrscheinlichkeit, mit der eine Zufallsvariable, die der angegebenen *Verteilung* folgt, einen Wert kleiner *Zahl* annimmt.

IDF.Verteilung(*Wahrscheinlichkeit, weitere Parameter der Verteilung*) Die IDF-Funktionen ergeben den Wert, den eine der angegebenen Verteilung folgende Zufallsvariable mit einer bestimmten kumulierten *Wahrscheinlichkeit* annimmt.

NCDF.Verteilung(*Zahl, ..., nc*) Ergibt die Wahrscheinlichkeit, mit der eine Zufallsvariable, der die angegebene nicht zentrale Verteilung zugrunde liegt, einen Wert kleiner oder gleich *Zahl* annimmt.

NPDF.Verteilung(*Zahl, ..., nc*) Die NPDF-Funktionen beziehen sich auf nichtzentrale Verteilungen und berechnen den Wert der Dichtefunktion einer ausgewählten Verteilung für einen vorgegebenen Wert.

PDF.Verteilung(*Zahl, ...*) Die PDF-Funktionen berechnen den Wert der Dichtefunktion einer ausgewählten Verteilung für einen vorgegebenen Wert.

RV.Verteilung(...) »Zieht« Zufallszahlen aus einer vorgegebenen Verteilung.

SIG.CHISQ(*Zahl, df*) Berechnet die Signifikanz für den Wert *Zahl* einer Chi-Quadrat-verteilten Zufallsvariablen.

SIG.F(*Zahl, df1, df2*) Berechnet die Signifikanz für den Wert *Zahl* einer F-verteilten Zufallsvariablen.

Logische Funktionen

ANY(*Testwert, Wert* [, *Wert,...*]) Ergibt 1, wenn *Testwert* mit einem der angegebenen Werte übereinstimmt, und 0, wenn dies nicht der Fall ist.

MISSING(*Variable*) Ergibt 1 bei fehlendem Wert und 0 bei gültigem Wert.

RANGE(*Testwert, Unten, Oben*[, *Unten, Oben,...*]) Ergibt 1, wenn *Testwert* innerhalb eines der angegebenen Bereiche liegt, und 0, wenn dies nicht der Fall ist.

SYSMIS(*Variable*) Ergibt 1 für systemdefinierten fehlenden Wert und 0 für gültigen oder benutzerdefinierten fehlenden Wert.

Funktionen zur Behandlung fehlender Werte

MISSING(*Variable*) Ergibt 1 bei fehlendem Wert und 0 bei gültigem Wert.

NMISS(*Wert* [, *Wert,...*]) Gibt die Anzahl der fehlenden Werte unter den angegebenen *Werten* aus.

NVALID(*Wert* [, *Wert...*]) Gibt die Anzahl der gültigen Werte unter den angegebenen *Werten* aus.

SYSMIS(*Variable*) Ergibt 1 bei systemdefiniertem fehlenden Wert und 0 bei gültigem oder benutzerdefiniertem fehlenden Wert.

VALUE(*Variable*) Ergibt den Wert von *Variable*, wobei Benutzerdefinitionen über fehlende Werte aufgehoben werden.

Textfunktionen

CHAR.INDEX(*Text, Suchtext*) Liefert die erste Position, an der sich *Suchtext* in *Text* befindet.

CHAR.INDEX(*Text, Suchtext, Anzahl*) Nennt die erste Position, an der sich einer von mehreren *Suchtexten* befindet.

CHAR.LENGTH(*Text*) Ermittelt die Länge von *Text*.

CHAR.LPAD(*Text, Länge*) Fügt vor *Text* so viele Leerzeichen ein, dass dieser die vorgegebene *Länge* erreicht.

CHAR.LPAD(*Text, Länge, Zeichen*) Fügt vor *Text* so viele Zeichen von *Zeichen* ein, dass *Text* die vorgegebene *Länge* erreicht.

CHAR.RINDEX(*Text, Suchtext*) Ergibt die letzte Position, an der sich *Suchtext* in *Text* befindet.

CHAR.RINDEX(*Text, Suchtext, Anzahl*) Nennt die letzte Position, an der sich einer von mehreren *Suchtexten* befindet.

CHAR.RPAD(*Text, Länge*) Hängt an *Text* so viele Leerzeichen an, dass dieser die vorgegebene *Länge* erreicht.

CHAR.RPAD(*Text, Länge, Zeichen*) Hängt an *Text* so viele Zeichen der Art *Zeichen*, dass dieser die vorgegebene *Länge* erreicht.

CHAR.SUBSTR(*Text, Position*) Gibt von *Text* die letzten Zeichen ab *Position* aus.

CHAR.SUBSTR(*Text, Position, Länge*) Gibt von *Text* ab *Position* so viele Zeichen aus, wie durch *Länge* angegeben.

CONCAT(*Text, Text* [,...]) Verknüpft einzelne *Texte* zu einem einzigen Text.

LOWER(*Text*) Wandelt Großbuchstaben von *Text* in Kleinbuchstaben um.

LTRIM(*Text*) Ergibt *Text* ohne vorausgehende Leerzeichen.

LTRIM(*Text, Zeichen*) Ergibt *Text*, wobei die ersten Zeichen abgeschnitten werden, sofern diese mit *Zeichen* identisch sind.

NUMBER(*Text, Format*) Wandelt einen *Textwert* in einen numerischen Wert um.

REPLACE(*Text, Suchtext, Neutext* [*Anzahl*]) Ersetzt in dem angegebenen *Text* die Zeichenfolge *Suchtext* durch die Zeichenfolge *Neutext*. Ist der *Suchtext* mehrfach in *Text* enthalten, kann die Anzahl der Ersetzungen mit dem optionalen Parameter *Anzahl* begrenzt werden.

RTRIM(*Text*) Gibt *Text* ohne abschließende Leerzeichen aus.

RTRIM(*Text, Zeichen*) Gibt *Text* ohne abschließende Zeichen der Art *Zeichen* aus.

STRING(*Zahl, Format*) Wandelt einen numerischen Wert in einen Textwert um.

UPCASE(*Text*) Gibt *Text* ausschließlich in Großbuchstaben wieder.

Zeit- und Datumsfunktionen

CTIME.DAYS(*Zahl*) Rechnet den seriellen Wert *Zahl* in Anzahl von Tagen um.

CTIME.HOURS(*Zahl*) Rechnet den seriellen Wert *Zahl* in Anzahl von Stunden um.

CTIME.MINUTES(*Zahl*) Rechnet den seriellen Wert *Zahl* in Anzahl von Minuten um.

CTIME.SECONDS(*Zahl*) Rechnet den seriellen Wert *Zahl* in Anzahl von Sekunden um.

DATE.XXX(*Datum*) Rechnet *Datum* in einen seriellen Wert um. Für *XXX* können verschiedene Spezifikationen vorgenommen werden, die es erlauben, ein Datum wahlweise auf den Tag, Monat, Quartal etc. genau anzugeben.

DATEDIFF(*Zeitpunkt1, Zeitpunkt2, Einheit*) Berechnet die Zeitspanne, die zwischen zwei Zeitpunkten liegt.

DATESUM(*Zeitpunkt, Zeitspanne, Einheit, Methode*) Addiert zu dem angegebenen Zeitpunkt die vorgegebene Zeitspanne hinzu und gibt als Ergebnis den resultierenden neuen Zeitpunkt aus.

TIME.DAYS(*Tage*) Rechnet einen in *Tagen* angegebenen Zeitraum in eine serielle Zahl um.

TIME.HMS(*Stunden* [, *Minuten* [, *Sekunden*]]) Ergibt die serielle Zahl, die einem in *Stunden, Minuten, Sekunden* angegebenen Zeitraum entspricht.

XDATE.XXX(*Datumswert*) Rechnet eine serielle Zahl in einen Zeitpunkt um. Durch die Spezifikation *XXX* können Sie festlegen, welche Dimension (Tagesdatum, Stunde, Monat etc.) des Zeitpunktes ausgegeben werden soll.

YRMODA(*Jahr, Monat, Tag*) Ergibt die Anzahl an Tagen, die seit dem 15. Oktober 1582 bis zu dem angegebenen Datum vergangen sind.

44.2 Funktionen in alphabetischer Reihenfolge

$CASENUM Systemvariable. Gibt für jeden Fall der Datendatei die Fallnummer aus.

$DATE Systemvariable. Gibt das aktuelle Datum mit zweistelliger Jahresangabe im Format A9 (tt-mmm-jj, also zum Beispiel 19-MAR-75) aus.

$DATE11 Systemvariable. Gibt das aktuelle Datum mit vierstelliger Jahresangabe im Format A11 (tt-mmm-jjjj, also zum Beispiel 19-MAR-1975) aus.

$JDATE Systemvariable. Gibt die Anzahl der Tage aus, die seit dem 14. Oktober 1582 bis zum aktuellen Datum vergangen sind.

$SYSMIS Systemvariable. Gibt einen systemdefinierten fehlenden Wert aus. Diese Systemvariable ist insbesondere hilfreich, um in einer Bedingung zu prüfen, ob eine Variable einen systemdefinierten fehlenden Wert enthält.

$TIME Gibt die Anzahl der Sekunden aus, die seit dem 14. Oktober 1582 bis zum aktuellen Datum und der aktuellen Uhrzeit vergangen sind. Dies ist der numerische Wert, den SPSS dem aktuellen Zeitpunkt zuweist.

ABS(*Zahl*) Numerisch. Gibt den absoluten Wert von *Zahl* aus.

ABS(-3) = ABS(3) = 3

ANY(*Testwert, Wert* [, *Wert,...*]) Numerisch (Logisch). Ergibt 1 (wahr), wenn *Testwert* mit irgendeinem der angegebenen *Werte* übereinstimmt, und 0 (falsch), wenn keine Übereinstimmung vorliegt. Sie können sowohl numerische Werte als auch Textwerte angeben. Textwerte müssen zwischen Anführungszeichen (oder Hochkommata) geschrieben werden.

ANY(0.1, 1/10, 5) = 1 ANY("0.1", "1/10", "5") = 0

ARSIN(*Zahl*) Numerisch. Das Argument *Zahl* ist ein Sinuswert, zu dem die Funktion den Winkel im Bogenmaß errechnet. Der Winkel wird im Bogenmaß zwischen $-\pi/2$ und $+\pi/2$ angegeben. *Zahl* muss zwischen -1 und +1 liegen.

ARSIN(0.5) $\approx$ 0,5236 = $\frac{1}{6}\pi$ ARSIN(1) $\approx$ 1,5708 = $\frac{1}{2}\pi$

ARTAN(*Zahl*) Numerisch. Das Argument *Zahl* ist ein Tangenswert, zu dem die Funktion den Winkel errechnet. Der Winkel wird im Bogenmaß zwischen $-\pi/2$ und $+\pi/2$ angegeben.

ARTAN(1) $\approx$ 0,7854 = $\frac{1}{4}\pi$ ARTAN(0) = 0

CDF.Verteilung(*Zahl, ...*) Numerisch. Ergibt die Wahrscheinlichkeit, mit der eine Zufallsvariable, die der angegebenen Verteilung folgt, einen Wert kleiner oder gleich *Zahl* annimmt. Die Funktion ermittelt somit die kumulierte Wahrscheinlichkeit für das Auftreten eines Wertes unter der Annahme einer bestimmten Verteilung. Diese Funktion steht für 25 Verteilungen zur Verfügung, die Sie jeweils mit den für die Verteilung relevanten Parametern spezifizieren können. Im Folgenden werden die CDF-Funktionen mit den unterschiedlichen Verteilungen auf-

geführt. Der Parameter *Zahl* gibt dabei jeweils den Wert an, für den die kumulierte Wahrscheinlichkeit errechnet werden soll.

Hinweis

Für nichtzentrale Verteilungen siehe auch die NCDF-Funktionen.

CDF.BERNOULLI(*Zahl, p*) Geben Sie mit *p* die konstante Wahrscheinlichkeit an, mit der ein Ereignis bei einer Versuchsdurchführung eintritt. *p* muss ein Wert zwischen 0 und 1 sein. Gilt nicht, dass $0 \leq p \leq 1$, ergibt die Funktion einen fehlenden Wert.

CDF.BERNOULLI(Zahl, p) ist identisch mit CDF.BINOM(Zahl, 1, p).

CDF.BERNOULLI(0.4, 0.3) = 0,7 CDF.BERNOULLI(1, p) = 1

CDF.BETA(*Zahl, y, z*) Liefert die Wahrscheinlichkeit, dass eine mit den Parametern *y* und *z* Beta-verteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.BINOM(*Zahl, n, p*) Geben Sie mit *n* die Anzahl der Versuchsdurchführungen und mit *p* die in jeder Durchführung geltende Wahrscheinlichkeit für das Eintreten eines Ereignisses an. Die Funktion liefert die Wahrscheinlichkeit, mit der bei *n* Versuchen das mit einer Wahrscheinlichkeit von *p* eintretende Ereignis höchstens mit der Häufigkeit von *Zahl* eintritt. *p* muss einen Wert zwischen 0 und 1 haben, und *n* muss ganzzahlig sein. Ist eine dieser beiden Regeln verletzt, gibt die Funktion einen fehlenden Wert aus.

CDF.BINOM(1, 3, 1/6) ≈ 0,925926. Dies ist die Wahrscheinlichkeit, dass bei drei Würfen mit einem sechsseitigen Würfel (bei dem jede Seite mit der gleichen Wahrscheinlichkeit von 1/6 erscheint) höchstens einmal eine Sechs fällt (dies gilt natürlich auch für die Eins, Zwei etc.).

CDF.BINOM(n, n, p) ergibt 1.

CDF.BINOM(Zahl<n, n, 1) ergibt 0.

CDF.BINOM(Zahl, n, 0) ergibt 1.

CDF.BINOM(Zahl, 1, p) ist identisch mit CDF.BERNOULLI(Zahl, p).

CDF.BVNOR(*Zahl1, Zahl2, r*) Diese Funktion liefert für eine bivariate Standardnormalverteilung mit dem Korrelationsparameter *r* die Wahrscheinlichkeit, dass die beiden Zufallsvariablen Werte kleiner *Zahl1* bzw. *Zahl2* annehmen.

CDF.CAUCHY(*Zahl, k, Stufe*) Ergibt die Wahrscheinlichkeit, dass eine mit den Parametern *k* und *Stufe* Cauchy-verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

CDF.CHISQ(*Zahl, Freiheitsgrade*) Ergibt die Wahrscheinlichkeit, mit der eine Zufallsvariable, die χ^2(*Freiheitsgrade*)-verteilt ist, einen Wert kleiner *Zahl* annimmt. Geben Sie für *Freiheitsgrade* einen Wert größer als 0 an. Sie können die *Freiheitsgrade* auch mit Dezimalstellen angeben.

CDF.CHISQ(40, 20) ergibt 0,995.

CDF.EXP(*Zahl, λ*) Gibt die Wahrscheinlichkeit an, mit der eine Exp(λ)-verteilte Zufallsvariable einen Wert kleiner als Zahl annimmt. λ muss größer als 0 sein.

CDF.Exp(2, 1) liefert ≈ 0,86467.

Allgemein liefert die Funktion *CDF.Exp(Zahl, λ)* das Ergebnis $1 - e^{-\lambda \cdot Zahl}$.

CDF.F(*Zahl, Freiheitsgrad1, Freiheitsgrad2*) Ergibt die Wahrscheinlichkeit, mit der eine F-verteilte Zufallsvariable mit den Freiheitsgraden *Freiheitsgrad1* im Zähler und *Freiheitsgrad2* im Nenner einen Wert kleiner als *Zahl* annimmt. Die beiden Angaben für die Freiheitsgrade müssen größer als 0 sein und dürfen Dezimalstellen aufweisen.

CDF.F(2,15,20) = 0,926056

CDF.GAMMA(*Zahl, k, λ*) Ergibt die Wahrscheinlichkeit, mit der eine GA(k, λ)-verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt. *λ* und *k* müssen größer als 0 sein.

CDF.GEOM(*Zahl, p*) Geben Sie mit *p* die Wahrscheinlichkeit für das Eintreten eines Ereignisses bei einem Versuch an. *p* muss größer als 0 und kleiner oder gleich 1 sein. Die Funktion liefert die Wahrscheinlichkeit, mit der eine geometrisch mit dem Parameter *p* verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

CDF.GEOM(2, 1/6) liefert 0,305. Dies entspricht der Wahrscheinlichkeit, dass man beim Würfeln mit einem gerechten sechsseitigen Würfel spätestens beim zweiten Versuch eine Sechs würfelt (eigentlich, dass man höchstens einen erfolglosen Versuch vergibt, bevor man Erfolg hat).

Allgemein liefert *CDF.Geom(Zahl, p)* das Ergebnis $1 - (1 - p)^{Zahl}$.

CDF.HALFNRM(*Zahl, Mittelwert, Standardabweichung*) Berechnet die Wahrscheinlichkeit, dass eine halbnormalverteilte Zufallsvariable mit den angegebenen Parametern einen Wert kleiner *Zahl* annimmt.

CDF.HYPER(*Zahl, Total, Ziehungen, mögliche Treffer*) Ergibt die Wahrscheinlichkeit, dass eine hypergeometrisch mit den Parametern *Total*, *Ziehungen* und *mögliche Treffer* verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt. Geben Sie für *Total*, *Ziehungen* und *mögliche Treffer* ganzzahlige Werte größer 0 an.

CDF.HYPER(2, 1000, 10, 100) ≈ 0,9308. Dies entspricht der Wahrscheinlichkeit, beim Ziehen von zehn Losen aus einer Gesamtmenge von 1.000 Losen, unter denen sich 100 Volltreffer befinden, weniger als zwei Volltreffer zu erwischen.

Mit einer Wahrscheinlichkeit von *1 – CDF.HYPER(2, 1000, 10, 100) ≈ 6,92 %* zieht man also zwei oder mehr Volltreffer.

CDF.IGAUSS(*Zahl, Mittel, Skal*) Gibt die Wahrscheinlichkeit an, dass eine Zufallsvariable, die einer inversen Normalverteilung mit den angegebenen Parametern folgt, einen Wert kleiner *Zahl* annimmt.

CDF.LAPLACE(*Zahl, Mittelwert, Stufe*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern *Mittelwert* und *Stufe* Laplace-verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

CDF.LNORMAL(*Zahl, μ_l, σ_l*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern μ_l und σ_l logarithmisch-normalverteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

CDF.LOGISTIC(*Zahl, Mittelwert, Stufe*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern *Mittelwert* und *Stufe* Logistik-verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

CDF.NEGBIN(*Zahl, Erfolge, p*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern *Erfolge* und *p* negativ-binomialverteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt. Dies ist die Wahrscheinlichkeit, mit der weniger als *Zahl* Versuche benötigt werden, um *Erfolge* Erfolge zu erzielen, wenn die Wahrscheinlichkeit eines Erfolgs bei jedem Versuch *p* beträgt.

CDF.NEGBIN(20, 5, 0.2) ≈ 0,37035. Dies entspricht der Wahrscheinlichkeit, beim Ziehen von Losen nach spätestens 19 Ziehungen den fünften Volltreffer gezogen zu haben, wenn man bei jedem Versuch mit einer Wahrscheinlichkeit von 0,2 einen Volltreffer zieht.

CDF.NORMAL(*Zahl, Mittelwert, Standardabweichung*) Ergibt die Wahrscheinlichkeit, dass eine mit den Parametern *Mittelwert* und *Standardabweichung* normalverteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.NORMAL(3, 3, 17) = 0,5

CDF.PARETO(*Zahl, k, Stufe*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern *k* und *Stufe* Pareto-verteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.POISSON(*Zahl, Mittelwert*) Ergibt die Wahrscheinlichkeit, mit der eine mit dem Parameter *Mittelwert* Poisson-verteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.SMOD(*Zahl, Größe, df*) Ergibt die Wahrscheinlichkeit, mit der eine Variable, die dem studentisierten Maximalmodul mit den angegebenen Parametern entspricht, einen Wert kleiner *Zahl* liefert.

CDF.SRANGE(*Zahl, Größe, df*) Berechnet die Wahrscheinlichkeit, mit der ein Wert aus der studentisierten Spannweitenstatistik mit den angegebenen Parametern kleiner *Zahl* ist.

CDF.T(*Zahl, Freiheitsgrade*) Ergibt die Wahrscheinlichkeit, mit der eine mit den vorgegebenen *Freiheitsgraden* t-verteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.T(1.752, 15) ≈ 0,95

CDF.UNIFORM(*Zahl, Min, Max*) Ergibt die Wahrscheinlichkeit, mit der eine in dem Intervall zwischen *Min* und *Max* gleichverteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDF.Uniform(3, 0, 9) = 1/3

CDF.WEIBULL(*Zahl, y, z*) Ergibt die Wahrscheinlichkeit, mit der eine mit den Parametern *y* und *z* Weibull-verteilte Zufallsvariable einen Wert kleiner *Zahl* annimmt.

CDFNORM(*Z-Wert*) Numerisch. Ergibt die Wahrscheinlichkeit, mit der eine standardnormalverteilte Zufallsvariable (normalverteilt mit Mittelwert 0 und Standardabweichung 1) einen Wert kleiner oder gleich *Z-Wert* annimmt.

CDFNORM(0) = 0,5 CDFNORM(–1.28) = 0,1

CFVAR(*Zahl, Zahl* [*, Zahl, ...*]) Numerisch. Gibt den Variationskoeffizienten der angegebenen gültigen Werte aus. Unter den angegebenen Werten müssen mindestens zwei gültige Werte sein, andernfalls ergibt sich ein fehlender Wert. Ist insgesamt nur ein Wert angegeben, wird die Funktion nicht ausgeführt.

Tipp

Sie können selbst eine Mindestanzahl gültiger Werte festlegen, unterhalb derer die Funktion einen fehlenden Wert liefert. Schreiben Sie die entsprechende Mindestzahl durch einen Punkt getrennt direkt hinter den Funktionsnamen. Soll die Funktion für den Fall, dass sich unter den Parametern weniger als fünf gültige Werte befinden, einen fehlenden Wert ausgeben, schreiben Sie die Funktion in der Form *CFVAR.5(...)*.

Der Variationskoeffizient ergibt sich als:

$$\frac{\text{Standardabweichung}}{\text{Mittelwert}} = \frac{\sqrt{\frac{(x_i - \bar{x})^2}{n-1}}}{\bar{x}}$$

Dabei bezeichnet x_i die gültigen (nicht fehlenden) Werte, $\bar{x}$ deren Mittelwert und *n* die Anzahl der gültigen Werte.

CHAR.INDEX(*Text, Suchtext*) Numerisch. Diese Funktion vergleicht, ob der *Suchtext* in *Text* enthalten ist, und gibt als Ergebnis die Position aus, an der *Suchtext* das erste Mal in *Text* auftaucht. Ist *Suchtext* nicht in *Text* enthalten, ergibt die Funktion 0. Die Funktion unterscheidet dabei zwischen Groß- und Kleinbuchstaben.

CHAR.INDEX("Capuccino", "ci") ergibt 6, *CHAR.INDEX("Espresso", "S")* ergibt 0, da der Großbuchstabe *S* nicht in »*Espresso*« enthalten ist.

Wenn Sie für *Text* oder *Suchtext* eine Textvariable angeben und einzelne Werte der Variablen kürzer sind, als es die definierte Variablenbreite zulässt, wird die Differenz zwischen der Länge des Wertes und der Breite der Variablen mit Leerzeichen aufgefüllt. Steht in einem Feld der Variablen *var1* mit der Länge 2 lediglich ein »*a*«, so gibt die Funktion *CHAR.INDEX("Capuccino", var1)* den Wert 0 aus, da in »*Capuccino*« nirgendwo die Zeichenfolge »*a* « (»*a*« mit darauffolgendem Leerzeichen) enthalten ist. Diesen Effekt können Sie mit der Funktion *RTRIM* vermeiden. *CHAR.INDEX("Capuccino", RTRIM(var1))* liefert 2.

CHAR.INDEX(*Text, Suchtext, Teillänge*) Numerisch. Diese Funktion entspricht weitgehend der vorhergehenden *CHAR.INDEX*-Funktion. Der Unterschied besteht darin, dass Sie mehrere (gleich lange) Argumente als Suchtext angeben können. Mit dem Parameter *Teillänge* zerlegen Sie den Suchtext in mehrere einzelne Suchtexte, die anschließend unabhängig voneinander in *Text* gesucht werden. Wählen Sie die Teillänge so, dass die Länge von *Suchtext* ein ganzzahliges Vielfaches von *Teillänge* ist. *Suchtext* wird dann in einzelne Suchtexte der Länge *Teillänge* zerlegt.

Geben Sie für Suchtext »*Hai*« und für Teillänge *1* an, so wird getrennt nach den Suchtexten »*H*«, »*a*« und »*i*« gesucht. Die Funktion gibt die erste Position innerhalb von *Text* aus, die einen der gesuchten Teilstücke aufweist. Lässt sich Suchtext nicht in gleich lange Ausdrücke der Länge *Teillänge* zerlegen, gibt die Funktion einen fehlenden Wert aus.

CHAR.INDEX("Capuccino","ci",1) = 5 CHAR.INDEX("Capuccino","si",1) = 7

CHAR.LENGTH(*Text*) Numerisch. Diese Funktion ermittelt die Länge (Anzahl der Zeichen) von *Text*. Wenn Sie für *Text* eine Variable angeben, ergibt die Funktion stets die für die Variable definierte Breite, unabhängig von der tatsächlichen Länge einzelner Werte.

CHAR.LENGTH("unendlich") = 9

Tipp

Ist der Wert »*unendlich*» in der Variablen *var1* enthalten, wobei für *var1* eine Länge von 16 definiert ist, gibt die Funktion *CHAR.LENGTH(var1)* den Wert 16 aus. Die Funktion *CHAR.LENGTH(RTRIM(var1))* liefert dagegen 9, da die Funktion *RTRIM* die Leerzeichen am Ende der Werte von *var1* abschneidet.

CHAR.LPAD(*Text, Länge*) String. Liefert den Textwert *Text*, der am Anfang um so viele Leerzeichen ergänzt wurde, dass sich ein Text der angegebenen *Länge* ergibt. Umfasst *Text* bereits die angegebene Länge (oder sogar mehr Zeichen), wird der Text unverändert ausgegeben. *Länge* muss ein Wert zwischen 1 und 255 sein.

CHAR.LPAD("Vier", 5) ergibt » *Vier*» (»*Vier*« mit Leerzeichen davor).

Hinweis

Wenn Sie nicht einen Text, sondern den Bezug auf eine Textvariable angeben, sollten Sie beachten, dass alle Werte der Variablen die für die Variable definierte Breite besitzen. Ist der Wert »*Aal*« in einer Textvariablen *textvar* mit der Breite 8 gespeichert, umfasst der einzelne Textwert ebenfalls acht Zeichen, da »*Aal*« automatisch um fünf Leerzeichen ergänzt wird. Die Funktion *CHAR.LPAD(Textvar, 8)* würde vor keinen Wert der Variablen ein Leerzeichen einfügen, da alle Werte bereits acht Zeichen lang sind.

CHAR.LPAD(*Text, Länge, Zeichen*) String. Ist der angegebene *Text* kürzer als die angegebene *Länge*, so werden vor *Text* so viele Zeichen der Art *Zeichen* eingefügt, bis sich ein Text der vorgegebenen *Länge* ergibt. Andernfalls wird *Text* unverändert als Ergebnis der Funktion ausgegeben. *Länge* ist ein ganzzahliger Wert zwischen 1 und 255, *Zeichen* darf lediglich ein einziges Zeichen umfassen.

CHAR.LPAD("Ja", 4, "#") ergibt »##Ja«.

CHAR.MBLEN.BYTE(*Text, Position*) Numerisch. In einigen asiatischen Sprachen belegen einzelne Zeichen mehr als ein Byte. Diese Funktion gibt die Bytezahl aus, die das Zeichen aus *Text* belegt, das sich an der vorgegebenen Byte-*Position* befindet.

CHAR.RINDEX(*Text, Suchtext*) Numerisch. Diese Funktion sucht *Suchtext* in *Text* und gibt als Ergebnis die Position aus, an der *Suchtext* das letzte Mal in *Text* enthalten ist. Ist *Suchtext* mehrere Zeichen lang, gibt die Funktion die Position des ersten Zeichens aus. Es wird zwischen Groß- und Kleinbuchstaben unterschieden.

CHAR.RINDEX("Massachusetts", "s") = 13
CHAR.RINDEX("Massachusetts", "as") = 2

CHAR.RINDEX(*Text, Suchtext, Teillänge*) Numerisch. Wie die andere *CHAR.RINDEX*-Funktion sucht auch diese Funktion nach einer Zeichenfolge in *Text*. Mit der zusätzlichen Angabe *Teillänge* können Sie den *Suchtext* in einzelne gleich lange Zeichenfolgen der Länge *Teillänge* zerlegen. Anschließend wird jedes dieser Teilstücke innerhalb von *Text* gesucht. Die Funktion gibt die letzte Position aus, an der einer der gesuchten Texte auftaucht. *Teillänge* muss so gewählt werden, dass sich der *Suchtext* in mehrere Zeichenfolgen der Länge *Teillänge* zerlegen lässt. Ist dies nicht der Fall, liefert die Funktion einen systemdefinierten fehlenden Wert.

Geben Sie für *Suchtext* »*Cuba*« und für *Teillänge 2* an, wird der angegebene Text nach den Zeichenfolgen »*Cu*« und »*ba*« abgesucht:

CHAR.RINDEX("Barrabas", "Cuba", 2) = 6

CHAR.RPAD(*Text, Länge*) String. Hängt (rechts) an den *Text* so viele Leerzeichen an, dass dieser die angegebene *Länge* erreicht. Ist *Text* ohnehin länger als *Länge*, wird der *Text* jedoch nicht entsprechend gekürzt. *Länge* muss eine ganze Zahl zwischen 1 und 255 sein, andernfalls liefert die Funktion keine Ergebnisse.

Hinweis

Ein Text, der als Wert einer String-Variablen gespeichert ist, jedoch nicht die gesamte für die Variable definierte Länge umfasst, wird von SPSS automatisch mit der entsprechenden Anzahl an Leerzeichen aufgefüllt. Diese werden weder in den einzelnen Feldern der Tabelle noch in der Bearbeitungszeile dargestellt, sind jedoch Bestandteil des Wertes und wirken sich somit bei einigen Text-Funktionen (so zum Beispiel bei der Funktion *CHAR.INDEX*) aus. Die *CHAR.RPAD*-Funktion kann daher verwendet werden, um einen Text so umzuwandeln, dass er anschließend mit den Werten einer Variablen wirklich identisch ist und sich auch nicht mehr durch fehlende Leerzeichen unterscheidet.

CHAR.RPAD(*Text, Länge, Zeichen*) String. Ist der angegebene *Text* kürzer als die bezeichnete *Länge*, so wird der *Text* (rechts) mit so vielen Zeichen der Art *Zeichen* aufgefüllt, bis *Text* die vorgegebene *Länge* erreicht hat. *Länge* muss eine ganze Zahl zwischen 1 und 255 sein, *Zeichen* darf lediglich ein Zeichen umfassen. Wenn Sie für *Zeichen* mehrere Zeichen angeben oder *Länge* außerhalb des erlaubten Bereichs liegt, gibt die Funktion keinen Wert aus.

CHAR.RPAD("xx", 4, "y") ergibt »xxyy«.

CHAR.RPAD("Känguruh", 4, "y") ergibt »Känguruh«.

CHAR.SUBSTR(*Text, Position*) String. Gibt von *Text* die letzten Zeichen beginnend mit dem Zeichen an der Stelle *Position* aus.

CHAR.SUBSTR("Dinosaurier", 5) ergibt »saurier «.

CHAR.SUBSTR(*Text, Position, Länge*) String. Gibt einen Ausschnitt des Ausdrucks *Text* aus. Der Ausschnitt beginnt mit dem durch *Position* angegebenen Zeichen und umfasst insgesamt *Länge* Zeichen.

CHAR.SUBSTR("Dinosaurier", 5, 3) ergibt »sau«.

CONCAT(CHAR.SUBSTR("Tatjana", 1, 3), CHAR.SUBSTR("Pieper", 2, 1)) ergibt »Tati«.

CONCAT(*Text, Text* [, ...]) String. Verknüpft die angegebenen Argumente zu einem Textwert. Es müssen mindestens zwei Argumente angegeben werden.

Hinweis

Wenn als Argumente Variablen angegeben werden, so dass die Inhalte einzelner Felder der Datendatei verknüpft werden, hängt das Verknüpfungsergebnis unter anderem von der festgelegten Variablenbreite der Quellvariablen ab. Sind die zu verknüpfenden Textwerte kürzer als die entsprechende Variablenlänge, werden die Zeichenfolgen mit Leerzeichen aufgefüllt. Damit die zusammengesetzten Werte in die Zielvariable geschrieben werden können, muss auch diese die erforderliche Variablenbreite aufweisen.

Stammen die Texte *Hai* und *Fisch* aus den Variablen *var1* und *var2*, wobei *var1* eine Variable der Länge 8 ist, liefert die Funktion *CONCAT(var1, var2)* den Text »*Hai Fisch*» als Ergebnis.

Die zusammengesetzte Funktion *CONCAT(RTRIM(var1), LOWER(var2))* liefert »*Haifisch*« als Ergebnis, wobei *RTRIM* die Leerzeichen des Wertes »*Hai* « abschneidet und *LOWER* den Großbuchstaben von »*Fisch*« in einen Kleinbuchstaben umwandelt.

COS(*Zahl*) Numerisch. Berechnet den Kosinus von *Zahl*; *Zahl* ist ein Winkel im Bogenmaß.

COS(0.5 * π) = 0

COS(0.7854) ≈ 0,7071 = $\frac{1}{2}\sqrt{2}$, denn 0,7854 ≈ $\frac{1}{4}\pi$

CTIME.DAYS(*Zahl*) Numerisch. Ergibt die Anzahl der Tage, die den seriellen Wert *Zahl* hat.

Hinweis

Der serielle Wert eines Zeitintervalls ergibt sich aus der Anzahl der in dem Zeitintervall liegenden Sekunden. Einem Tag entspricht somit die serielle Zahl 86.400. Diese Funktion gibt nicht nur ganzzahlige Werte, sondern auch Bruchteile eines Tages aus. Ist *Zahl* kleiner als 0, gibt die Funktion eine negative Anzahl von Tagen aus.

CTIME.DAYS(129600) = 1,5

CTIME.DAYS(TIME.DAYS(3)) = 3

CTIME.DAYS(TIME.DAYS(YRMODA(94,10,1))-TIME.DAYS(YRMODA(94,9,1)))=30

CTIME.HOURS(*Zahl*) Numerisch. Ergibt die Anzahl an Stunden, die dem seriellen Wert *Zahl* entspricht. Die Stundenanzahl wird ggf. mit Dezimalstellen angegeben. Ist *Zahl* ein negativer Wert, gibt auch die Funktion einen negativen Wert aus.

Hinweis

Der serielle Wert eines Zeitintervalls ergibt sich aus der entsprechenden Anzahl an Sekunden. Eine Stunde hat den seriellen Wert 3.600.

CTIME.HOURS(60) = $0{,}1\overline{6} = \frac{1}{60}$

CTIME.HOURS(TIME.HMS(15, 30) – TIME.HMS(10)) = 5,5

CTIME.MINUTES(*Zahl*) Numerisch. Ergibt die Anzahl an Minuten, deren serieller Wert gleich *Zahl* ist. Der Ausgabewert kann Dezimalstellen aufweisen und auch negativ sein.

Hinweis

Der serielle Wert eines Zeitintervalls entspricht der Anzahl an Sekunden, die das Zeitintervall umfasst. Eine Minute hat somit den seriellen Wert 60.

CTIME.MINUTES(333) = 5,55

CTIME.MINUTES(TIME.HMS(14, 56) – TIME.HMS(15, 30, 30)) = -34,5

CTIME.SECONDS(*Zahl*) Numerisch. Ergibt die Anzahl an Sekunden, die dem seriellen Wert *Zahl* entsprechen. Das Ergebnis kann negativ sein und weist ggf. Dezimalstellen auf.

Hinweis

Jedem Zeitintervall entspricht ein serieller Wert, der mit der Anzahl der in dem Zeitintervall liegenden Sekunden identisch ist. Eine Sekunde hat also den seriellen Wert 1.

CTIME.SECONDS(-17.3) = -17,3

CTIME.SECONDS(TIME.HMS(17, 15, 20) – TIME.HMS(17, 20, 30)) = -310

DATE.XXX(*Datum*) Numerisch. Datumsfunktionen berechnen den seriellen Wert des angegebenen Datums. Das Datum darf nicht vor dem 15. Oktober 1582 liegen, da der Kalender von SPSS erst mit diesem Tag beginnt. Jedem Zeitpunkt nach dem Anfangsdatum des Kalenders entspricht ein serieller Wert. Der 15. Oktober 1582 hat (um 0:00:00,00 Uhr) den seriellen Wert 86.400 (Anzahl der Sekunden eines Tages). Der serielle Wert eines späteren Zeitpunktes ergibt sich aus der Summe des Anfangswertes (86.400) und der Anzahl an Sekunden, die seit dem Anfangszeitpunkt vergangen sind. Der 15. Oktober 1582, 0:01:00,00 hat somit den seriellen Wert 86.460. Serielle Werte unter 86.400 werden von SPSS als Uhrzeitangaben ohne Datumsbezeichnung interpretiert. Der Uhrzeit 0:00:01 (äquivalent dem Zeitintervall von einer Sekunde) entspricht der serielle Wert 1. Dieser

Wert wird für höhere Zeitangaben entsprechend hochgezählt, bis schließlich die Uhrzeit 23:59:59 den seriellen Wert 86.399 ergibt.

Als serieller Wert eines Datums ohne nähere Uhrzeitangaben wird der Wert des Datums zu Beginn des Tages (0:00:00 Uhr) ausgegeben. Auf entsprechende Weise wird bei noch gröberen Datumsangaben verfahren. Geben Sie lediglich ein Jahr und einen Monat an, so wird der serielle Wert für den Beginn des ersten Tages des Monats errechnet. Eine Jahresangabe ohne weitere Bezeichnungen entspricht dem Beginn des Jahres genau in dem Zeitpunkt des Jahreswechsels.

Insgesamt stehen sechs Datumsfunktionen zur Verfügung, die jeweils den seriellen Wert des angegebenen Datums ausgeben. Die einzelnen Funktionen unterscheiden sich lediglich in der Form und der Genauigkeit, in der das Datum angegeben wird.

DATE.DMY(*Tag, Monat, Jahr*) Numerisch. Geben Sie ein Datum bis auf den Tag genau in der in Deutschland üblichen Form *Tag, Monat, Jahr* an.

DATE.DMY(11+8, 3, 99) ergibt die serielle Zahl 13.330.569.600. In einer Variablen mit dem Datumsformat *dd-mmm-yyyy* wird diese Zahl in der Form 19-MAR-2005 dargestellt.

DATE.MDY(*Monat, Tag, Jahr*) Numerisch. Geben Sie ein Datum in der in den USA gängigen Form Monat, Tag, Jahr an.

Haben Sie die Angaben eines Datums getrennt in den drei Variablen *monat*, *tag*, und *jahr* gespeichert, können Sie diese Funktion benutzen, um die einzelnen Angaben in einer neuen Variablen zu einem gemeinsamen Datum zusammenzuführen. Schreiben Sie *DATE.MDY(monat, tag, jahr)*, um die serielle Zahl des gesamten Datums zu erhalten. Wählen Sie anschließend für die Zielvariable ein geeignetes Datumsformat (zum Beispiel *mm/dd/yy*), mit dem der neue Wert als interpretierbares Datum dargestellt wird.

DATE.MOYR(*Monat, Jahr*) Numerisch. Das Datum wird lediglich durch den Monat und das Jahr beschrieben.

Für die Variablen *Datum1 = 15.6.2005* und *Datum2 = 1.1.2006* ergibt die Funktion

DATE.MOYR(XDATE.MONTH(datum1), XDATE.YEAR(datum2))

das Datum Juni 2006 (die resultierende serielle Zahl entspricht dem 1. Juni 2006).

DATE.QYR(*Quartal, Jahr*) Numerisch. Geben Sie das Quartal und das Jahr an, dessen seriellen Wert Sie errechnen möchten.

DATE.QYR(3,05) ergibt die Zahl 13.339.555.200. In dem Datumsformat *dd.mm.yy* wird diese Zahl in der Form 01.07.05 dargestellt, das Format *q Q yyyy* stellt die Zahl in der Form 3 Q 2005 dar.

DATE.WKYR(*Wochenzahl, Jahr*) Numerisch. Geben Sie an, für welche *Woche* von *Jahr* der serielle Wert errechnet werden soll.

DATE.WKYR(2, 05) = 13.324.521.600 (Dies entspricht dem 8. Januar 2005.)

DATE.YRDAY(*Jahr, Tageszahl*) Numerisch. Beschreiben Sie ein Datum, für das der serielle Wert berechnet werden soll, durch die Angabe des Jahres und die Nummer des Tages innerhalb des Jahres (1, 2, ..., 366).

DATE.YRDAY(2005, 417) ergibt den 21. Februar 2006 (13.359.859.200). Das gleiche Ergebnis liefert auch die Funktion *DATE.YRDAY(2006, 52)*.

DATEDIFF(*Zeitpunkt1, Zeitpunkt2, »Einheit«*) Berechnet die Zeitspanne, die zwischen zwei Zeitpunkten liegt. Das Ergebnis wird in der vorgegebenen *Einheit* als ganze Zahl ohne Nachkommastellen ausgegeben. Um die Einheit festzulegen, verwenden Sie eines der folgenden Schlüsselwörter, die zwischen Anführungszeichen gesetzt werden müssen: *years*, *quarters*, *months*, *weeks*, *days*, *hours*, *minutes*, *seconds*. Für *Zeitpunkt1* und *Zeitpunkt2* müssen zwei Variablen angegeben werden, die als Variablentyp ein Datums- oder Zeitformat haben.

DATESUM(*Zeitpunkt, Zeitspanne, »Einheit«* [, *»Methode«*]) Addiert zu dem angegebenen Zeitpunkt die vorgegebene Zeitspanne hinzu und gibt als Ergebnis den resultierenden neuen Zeitpunkt aus. Als *Zeitpunkt* muss eine Variable angegeben werden, die als Variablentyp ein Datums- oder Zeitformat hat. Die *Zeitspanne* wird als numerischer Wert angegeben. Mit dem Parameter *Einheit* legen sie fest, in welcher Einheit die Zeitspanne gemessen wird. Verwenden Sie hierzu eines der folgenden Schlüsselwörter, das zwischen Anführungszeichen angegeben werden muss: *years*, *quarters*, *months*, *weeks*, *days*, *hours*, *minutes*, *seconds*.

Die Angabe einer *Methode* ist optional; sie steuert den Umgang mit Schaltjahren. Sie können zwischen den Methoden *closest* und *rollover* wählen. Mit *rollover* werden »überschüssige Tage« in einen Folgemonat verschoben, so dass der 29. Februar + 1 Jahr den 1. März ergibt. Mit *closest* wird bei Schaltjahr-Konflikten jeweils das nächstliegende zulässige Datum innerhalb des betreffenden Monats gewählt; bei dieser Methode ergibt 29. Februar + 1 Jahr somit den 28. Februar. Das Schlüsselwort für die Methode muss stets zwischen Anführungszeichen angegeben werden.

EXP(*Zahl*) Numerisch. Potenziert den Wert *e* (≈2,718282) mit *Zahl*.
EXP(*Zahl*) = e^{Zahl}

IDF.Verteilung(*Wahrscheinlichkeit, weitere Parameter der Verteilung*) Numerisch. Die IDF-Funktionen berechnen den Wert, den eine Zufallsvariable, die der angegebenen Verteilung folgt, mit der vorgegebenen kumulierten *Wahrscheinlichkeit* annimmt. Mit dieser Wahrscheinlichkeit liegt der Wert dieser Zufallsvariablen also unter dem Ergebnis der IDF-Funktion.

Im Folgenden werden die 18 IDF-Funktionen für die unterschiedlichen Verteilungen aufgeführt. Der Parameter *Wahrscheinlichkeit* gibt jeweils die kumulierte Wahrscheinlichkeit an, für die der entsprechende Wert berechnet wird. Die übrigen Parameter spezifizieren die jeweilige Verteilung.

IDF.BETA(*Wahrscheinlichkeit, y, z*) Beta-Verteilung.

IDF.CAUCHY(*Wahrscheinlichkeit, k, Stufe*) Cauchy-Verteilung.

IDF.CHISQ(*Wahrscheinlichkeit, Freiheitsgrade*) Chi-Quadrat-Verteilung.

IDF.EXP(*Wahrscheinlichkeit, λ*) Exponential-Verteilung.

IDF.F(*Wahrscheinlichkeit, Freiheitsgrad1, Freiheitsgrad2*) F-Verteilung.

IDF.GAMMA(*Wahrscheinlichkeit, k, λ*) Gamma-Verteilung.

IDF.HALFNR(*Wahrscheinlichkeit, Mittelwert, Standardabweichung*) Halbnormalverteilung.

IDF.IGAUSS(*Wahrscheinlichkeit, Mittel, Skal*) Inverse Normalverteilung.

IDF.LAPLACE(*Wahrscheinlichkeit, Mittelwert, Stufe*) Laplace-Verteilung.

IDF.LNORMAL(*Wahrscheinlichkeit, m_l , s_l*) Logarithmische Normalverteilung.

IDF.LOGISTIC(*Wahrscheinlichkeit, Mittelwert, Stufe*) Logistische Verteilung.

IDF.NORMAL(*Wahrscheinlichkeit, Mittelwert, Standardabweichung*) Normalverteilung.

IDF.PARETO(*Wahrscheinlichkeit, k, Stufe*) Pareto-Verteilung.

IDF.SMOD(*Wahrscheinlichkeit, Größe, df*) Studentisiertes Maximalmodul.

IDF.SRANGE(*Wahrscheinlichkeit, Größe, df*) Studentisierte Spannweite.

IDF.T(*Wahrscheinlichkeit, Freiheitsgrade*) T-Verteilung.

IDF.UNIFORM(*Wahrscheinlichkeit, Min, Max*) Gleich-Verteilung.

IDF.WEIBULL(*Wahrscheinlichkeit, y, z*) Weibull-Verteilung.

INDEX(*Text, Suchtext*) Numerisch. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.INDEX(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

LAG(*Variable*) Numerisch oder String. Ergibt den Wert, den die angegebene Variable im vorhergehenden Fall aufweist. Für den ersten Fall gibt die Funktion bei numerischen Variablen einen systemdefinierten fehlenden Wert aus, bei Textvariablen bleibt das erste Feld leer.

LAG(alter) ergibt für Fall 23 den Wert, den die Variable *alter* in Fall 22 aufweist.

LAG(*Variable, nFälle*) Numerisch oder String. Liefert den Wert, den die angegebene Variable in dem um *nFälle* vorausgehenden Fall aufweist. Für die ersten *nFälle* Fälle ergeben sich für numerische Variablen fehlende Werte und für Textvariablen leere Felder.

Die Funktion *LAG(alter, 3)* liefert für Fall 23 den Wert, den die Variable *alter* in Fall 20 aufweist.

LENGTH(*Text*) Numerisch. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.LENGTH(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

LG10(*Zahl*) Numerisch. Berechnet den Logarithmus von *Zahl* zur Basis 10. Für Zahl ≤ 0 liefert die Funktion einen systemdefinierten fehlenden Wert.

LG10(100) = 2 LG10(10**6) = 6

LN(*Zahl*) Numerisch. Berechnet den natürlichen Logarithmus von *Zahl* (der Wert, mit dem *e* potenziert werden muss, damit sich *Zahl* ergibt). Für Zahl ≤ 0 ergibt sich ein systemdefinierter fehlender Wert.

LN(2.71828) ≈ 1 LN(1) = 0

LNGAMMA(*Zahl*) Numerisch. Berechnet den Logarithmus der vollständigen Gamma-Funktion für *Zahl*. *Zahl* muss ein positiver Wert sein.

LOWER(*Text*) String. Gibt den angegebenen Text ausschließlich in Kleinbuchstaben wieder.

LOWER("Harbour Lights") ergibt »harbour lights«.

LPAD(*Text, Länge*) String. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.LPAD(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

LTRIM(*Text*) String. Dies ist die Umkehrfunktion zu *CHAR.LPAD(Text, Länge)*. Sie gibt den angegebenen Text ohne vorausgehende Leerzeichen aus.

LTRIM(" Box") ergibt »Box«.

LTRIM(*Text, Zeichen*) String. Diese Funktion bewirkt die Umkehrung der Funktion *CHAR.LPAD(Text, Länge, Zeichen)*. Sie liefert den angegebenen *Text* ohne führende mit *Zeichen* identische Zeichen. Für Zeichen können Sie dabei lediglich ein einzelnes Zeichen angeben. Dabei wird zwischen Groß- und Kleinbuchstaben unterschieden.

LTRIM("Aal","A") ergibt »al«.

LTRIM(CHAR.LPAD(Text, Länge, Zeichen), Zeichen) ergibt *Text*.

MAX(*Wert, Wert* [, ...]) Numerisch oder String. Gibt den größten der angegebenen Werte aus. Sie können sowohl numerische als auch Textwerte angeben, innerhalb einer Funktion müssen jedoch alle Werte vom gleichen Typ sein. Sie können numerische Werte auch dann nicht mit Textwerten kombinieren, wenn die Textwerte als Zahlen interpretierbar sind. Als Argumente müssen mindestens zwei Werte angegeben werden. Fehlende Werte (auch benutzerdefinierte fehlende Werte) werden nicht berücksichtigt.

Tipp

Wenn die Funktion erst ab einer bestimmten Anzahl gültiger Werte ausgeführt werden soll, können Sie eine entsprechende Mindestanzahl vorgeben. Schreiben Sie die gewünschte Mindestanzahl gültiger Werte durch einen Punkt getrennt direkt hinter den Funktionsnamen *MAX*. Wenn die Funktion nur für drei oder mehr gültige Werte ausgeführt werden soll, schreiben Sie *MAX.3(Wert1, Wert2 [, Wert3...])*. Wird die Mindestanzahl gültiger Werte nicht erreicht, gibt die Funktion dann einen systemdefinierten fehlenden Wert aus.

MAX(3, -17) = 3

MAX("Zimbabwe", "Zaire") = »Zimbabwe«

MAX(3, "4") wird nicht ausgeführt, aber *MAX(3, NUMBER("4", E1))* ergibt 4, wobei die *NUMBER*-Funktion den Textwert "*4*" in den numerischen Wert *4* umwandelt.

MBLEN.BYTE(*Text, Position*) Numerisch. In einigen asiatischen Sprachen belegen einzelne Zeichen mehr als ein Byte. Diese Funktion gibt die Bytezahl aus,

die das Zeichen aus *Text* belegt, das sich an der vorgegebenen Byte-*Position* befindet.

MEAN(*Zahl, Zahl* [, ...]) Numerisch. Berechnet das arithmetische Mittel der angegebenen Werte. Es müssen mindestens zwei ausschließlich numerische Werte angegeben werden. Fehlende Werte (auch benutzerdefinierte) werden für den betreffenden Fall ausgeschlossen. Geben Sie als Parameter Variablen an, kann der Mittelwert also für verschiedene Fälle auf einer unterschiedlichen Anzahl an Werten beruhen.

Tipp

Sie können vorgeben, dass eine Mindestanzahl gültiger Werte vorhanden sein muss, damit die Funktion ausgeführt wird. Ist die tatsächliche Anzahl gültiger Werte geringer, gibt die Funktion einen systemdefinierten fehlenden Wert aus. Schreiben Sie die betreffende Zahl durch einen Punkt getrennt direkt hinter den Funktionsnamen. Um die Funktion nur für fünf oder mehr gültige Werte auszuführen, schreiben Sie *MEAN.5(Zahl,Zahl,...)*.

MEAN(1, 3, 5, 3/0 , 7, 9) = 5

MEAN.6(1, 3, 5, 3/0, 7, 9) gibt einen systemdefinierten fehlenden Wert aus.

MEDIAN(*Zahl, Zahl* [, ...]) Numerisch. Gibt den Median (das 50%-Perzentil) der angegebenen Argumente aus. Es müssen mindestens zwei ausschließlich numerische Werte angegeben werden. Fehlende Werte (auch benutzerdefinierte) werden für den betreffenden Fall ausgeschlossen.

MIN(*Wert, Wert* [, ...]) Gibt den kleinsten Wert der angegebenen Argumente aus. Sie können sowohl numerische als auch Textwerte angeben, aber nicht beide Arten in einer Funktion kombinieren. Es müssen mindestens zwei Argumente angegeben werden. Werte, die vom Benutzer als fehlende Werte gekennzeichnet sind, werden nicht berücksichtigt.

Tipp

Soll die Funktion erst ab einer bestimmten Anzahl gültiger Werte ausgeführt werden, können Sie die entsprechende Zahl durch einen Punkt getrennt hinter den Funktionsnamen schreiben. Soll die Funktion nur für mehr als fünf gültige Werte ausgeführt werden, geben Sie die Funktion in der Form *MIN.6(Wert, Wert, ...)* an. Wird die Mindestanzahl gültiger Werte nicht erreicht, gibt die Funktion einen systemdefinierten fehlenden Wert aus.

MIN(ABS(-3), 2) = 2

MIN("schwarz", "weiß") = »schwarz«

Tipp

Um Textwerte und numerische Werte miteinander zu vergleichen, können Sie die Funktionen *NUMBER* und *STRING* verwenden.

MISSING(*Variable*) Numerisch (Logisch). Ergibt 1 (wahr), wenn *Variable* einen fehlenden Wert aufweist, und 0 (falsch), wenn *Variable* einen gültigen Wert enthält. Fehlende Werte sind dabei nicht nur systemdefinierte, sondern auch benutzerdefinierte fehlende Werte. *Variable* ist ein Variablenname der Datendatei.

Tipp

Um systemdefinierte fehlende Werte von allen übrigen fehlenden Werten zu unterscheiden, verwenden Sie die Funktion *SYSMIS*.

MOD(*Zahl, Nenner*) Numerisch. Dividiert *Zahl* durch *Nenner* und liefert den Rest, der sich nicht ganzzahlig durch *Nenner* dividieren lässt. Geben Sie für *Nenner* eine Null ein, so wird ein systemdefinierter fehlender Wert ausgegeben.

MOD(5.4, 3) = 2,4 MOD(-7, 3) = -1

NCDF.Verteilung(*Zahl, ..., nc*) Numerisch. Ergibt die Wahrscheinlichkeit, mit der eine Zufallsvariable, der die angegebene Verteilung zugrunde liegt, einen Wert kleiner oder gleich *Zahl* annimmt. Die Funktion ermittelt also die kumulierte Wahrscheinlichkeit für das Auftreten eines Wertes unter der Annahme einer bestimmten Verteilung. Die Funktion steht für vier nichtzentrale Verteilungen zur Verfügung, die Sie jeweils mit den für die Verteilung relevanten Parametern näher spezifizieren können. Der Parameter *Zahl* gibt bei den im Folgenden aufgeführten NCDF-Verteilungen jeweils den Wert an, für den die kumulierte Wahrscheinlichkeit errechnet werden soll, mit *nc* können Sie den Grad der Nichtzentralität festlegen.

Hinweis

Für weitere Verteilungen schauen Sie unter den CDF-Funktionen nach.

NCDF.BETA(*Zahl, shape1, shape2, nc*) Ergibt die Wahrscheinlichkeit, mit der eine nichtzentrale mit den Parametern *shape1* und *shape2* sowie der Nichtzentralität *nc* Beta-verteilte Zufallsvariable einen Wert kleiner als *Zahl* annimmt.

NCDF.CHISQ(*Zahl, Freiheitsgrade, nc*) Ergibt die Wahrscheinlichkeit, mit der eine nichtzentrale χ^2-verteilte Zufallsvariable mit einer Nichtzentralität von *nc* und den vorgegebenen *Freiheitsgraden* einen Wert kleiner *Zahl* annimmt.

NCDF.F(*Zahl, Freiheitsgrad1, Freiheitsgrad2, nc*) Ergibt die Wahrscheinlichkeit, mit der eine nichtzentrale F-verteilte Zufallsvariable mit einer Nichtzentralität *nc* und *Freiheitsgrad1* Freiheitsgraden im Zähler sowie *Freiheitsgrad2* Freiheitsgraden im Nenner einen Wert kleiner *Zahl* annimmt.

NCDF.T(*Zahl, Freiheitsgrade, nc*) Ergibt die Wahrscheinlichkeit, mit der eine nichtzentrale T-verteilte Zufallsvariable mit den vorgegebenen *Freiheitsgraden* und einer Nichtzentralität von *nc* einen Wert kleiner *Zahl* annimmt.

NMISS(*Numwert* [, *Numwert, ...*]) Numerisch. Ermittelt die Anzahl der fehlenden Werte unter den angegebenen numerischen Werten. Sie können einen oder mehrere numerische Werte angeben.

NMISS(12, 1/0, 56, SQRT(-3), 8) = 2

Werden als Argumente Variablen angeführt, wird für jeden Fall einzeln ermittelt, wie viele fehlende Werte die angegebenen Variablen in dem betreffenden Fall aufweisen. Dabei werden sowohl systemdefinierte als auch benutzerdefinierte fehlende Werte gezählt.

NORMALIZE(Text) String. Gibt den *Text* in normalisierter Form aus, im Unicode-Modus in Unicode NFC. Im Codepage-Modus wird der *Text* unverändert ausgegeben.

NPDF.BETA(*Zahl, Form1, Form2, nc*) Numerisch. Berechnet für *Zahl* den Wert der Dichtefunktion einer nichtzentralen Betaverteilung mit den vorgegebenen *Form*-Parametern und dem Nichtzentralitätsgrad *nc*.

NPDF.CHISQ(*Zahl, df, nc*) Numerisch. Berechnet für *Zahl* den Wert der Dichtefunktion einer nichtzentralen Chi-Quadrat-Verteilung mit *df* Freiheitsgraden und dem Nichtzentralitätsgrad *nc*.

NPDF.F(*Zahl, df1, df2, nc*) Numerisch. Berechnet für *Zahl* den Wert der Dichtefunktion einer nichtzentralen F-Verteilung mit *df1* und *df2* Freiheitsgraden und dem Nichtzentralitätsgrad *nc*.

NPDF.T(*Zahl, df, nc*) Numerisch. Berechnet für *Zahl* den Wert der Dichtefunktion einer nichtzentralen T-Verteilung mit *df* Freiheitsgraden und dem Nichtzentralitätsgrad *nc*.

NTRIM(*Variable*). Gibt den Wert der *Variablen* ohne die Entfernung führender Leerzeichen aus. Als *Variable* muss direkt der Name einer vorhandenen Variablen angegeben werden. Andere Ausdrücke sind hier nicht zulässig.

NUMBER(*Text, Format*) Numerisch. Wandelt einen Wert, der in Textform angegeben ist, aber gleichzeitig als numerischer Wert interpretiert werden kann, in einen numerischen Wert um. Mit *Format* geben Sie das Variablenformat an, das als Grundlage zur Interpretation von *Text* herangezogen wird. Der *Text* wird damit so gelesen, als wäre er in dem Variablenformat *Format* dargestellt. Dabei werden Dezimaltrennzeichen ggf. dort ergänzt, wo nach dem angegebenen Format solche zu erwarten wären. Sind in dem Text bereits Dezimaltrennzeichen enthalten, werden diese korrekt umgesetzt. Kann *Text* nicht im Sinne des *Formats* interpretiert werden, so gibt die Funktion einen fehlenden Wert aus.

NUMBER("1234567", E8.2) = NUMBER("12345,67", E8.2) = 12345,67

NUMBER("1234,567", E8.2) = 1234,567

Sie können bei dieser Funktion nicht nur den Variablentyp *Numerisch* vorgeben, sondern auch alle anderen numerischen Variablentypen:

NUMBER("30Mar1993", Date11) liefert die Zahl 12.952.828.800. Diese kann in einem Datumsformat wieder als Datum gelesen werden. In dem Format *ddmmyy* wird sie beispielsweise in der Form *30.03.93* dargestellt.

NVALID(*Numwert* [, *Numwert, ...*]) Numerisch. Bestimmt die Anzahl der gültigen (nicht fehlenden) Werte unter den angegebenen Parametern. Als Parameter der Funktion können Sie einen oder mehrere numerische Werte angeben.

NVALID(12, 1/0, 56, SQRT(3), 8) = 3

Geben Sie als Parameter Variablen an, wird für jeden Fall die Anzahl der in den angegebenen Variablen enthaltenen fehlenden Werte ermittelt. Dabei werden sowohl systemdefinierte als auch benutzerdefinierte fehlende Werte als ungültig betrachtet. Beinhaltet eine Variable dagegen einen numerischen Wert, der grundsätz-

lich gültig ist, in dem aktuellen Format der Variablen jedoch nicht dargestellt werden kann, so wird dieser Wert als gültig gewertet.

PDF.Verteilung(...) Numerisch. Die PDF-Funktionen berechnen den Wert der Dichtefunktion einer ausgewählten Verteilung für einen vorgegebenen Wert. Die Verteilung lässt sich mit den jeweiligen Parametern spezifizieren; zur Verfügung stehen die folgenden Verteilungen:

PDF.BERNOULLI(*Zahl, p*) Bernoulli-Verteilung.

PDF.BETA(*Zahl, y, z*) Beta-Verteilung.

PDF.BINOM(*Zahl, n, p*) Binomial-Verteilung.

PDF.BVNOR(*Zahl1, Zahl2, r*) Bivariate Standardnormalverteilung.

PDF.CAUCHY(*Zahl, k, Stufe*) Cauchy-Verteilung.

PDF.CHISQ(*Zahl, Freiheitsgrade*) Chi-Quadrat-Verteilung.

PDF.EXP(*Zahl, λ*) Exponentialverteilung.

PDF.F(*Zahl, Freiheitsgrad1, Freiheitsgrad2*) F-Verteilung.

PDF.GAMMA(*Zahl, k, λ*) Gamma-Verteilung.

PDF.GEOM(*Zahl, p*) Geometrische Verteilung.

PDF.HALFNRM(*Zahl, Mittelwert, Standardabweichung*) Halbe Normalverteilung.

PDF.HYPER(*Zahl, Total, Ziehungen, mögliche Treffer*) Hypergeometrische Verteilung.

PDF.IGAUSS(*Zahl, Mittel, Skal*) Gauß-Verteilung.

PDF.LAPLACE(*Zahl, Mittelwert, Stufe*) Laplace-Verteilung.

PDF.LNORMAL(*Zahl, μ_l, σ_l*) Log-Normalverteilung.

PDF.LOGISTIC(*Zahl, Mittelwert, Stufe*) Logistische Verteilung.

PDF.NEGBIN(*Zahl, Erfolge, p*) Negativ-Binomialverteilung.

PDF.NORMAL(*Zahl, Mittelwert, Standardabweichung*) Normalverteilung.

PDF.PARETO(*Zahl, k, Stufe*) Pareto-Verteilung.

PDF.POISSON(*Zahl, Mittelwert*) Poisson-Verteilung.

PDF.T(*Zahl, Freiheitsgrade*) T-Verteilung.

PDF.UNIFORM(*Zahl, Min, Max*) Gleichverteilung.

PDF.WEIBULL(*Zahl, y, z*) Weibull-Verteilung.

RANGE(*Testwert, Unten, Oben* [, *Unten, Oben, ...*]) Numerisch (Logisch). Überprüft, ob *Testwert* innerhalb eines der durch die Grenzen *Unten* und *Oben* beschriebenen Wertebereichs liegt. Dabei zählen auch die Grenzen selbst zu dem Bereich. Ist dies der Fall, ergibt die Funktion 1, andernfalls ergibt sie 0. *Testwert*, *Unten* und *Oben* müssen entweder durchgehend numerische Werte oder Textwerte sein. Sie können einen oder mehrere Bereich(e) angeben. Die Funktion liefert bereits 1 (wahr), wenn *Testwert* in nur einem der Bereiche liegt.

RANGE(3, 2, 4) = 1 RANGE("b", "c" ,"f" ,"b" ,"d") = 1

Hinweis

Die untere Grenze muss stets kleiner oder gleich der korrespondierenden oberen Grenze sein; ist dies nicht der Fall, liefert die Funktion einen fehlenden Wert. So liefert *RANGE("drei", "zwei", "vier")* einen fehlenden Wert, da die untere Grenze über der oberen liegt.

REPLACE(*Text, Suchtext, Neutext* [*Anzahl*]) Ersetzt in dem angegebenen *Text* die Zeichenfolge *Suchtext* durch die Zeichenfolge *Neutext*. Ist der *Suchtext* mehrfach in *Text* enthalten, kann die Anzahl der Ersetzungen mit dem optionalen Parameter *Anzahl* begrenzt werden.

REPLACE("Barrabas", "a", "u") = »Burrubus«

REPLACE("Barrabas", "a", "u", 2) = »Burrubas«

RINDEX(*Text, Suchtext*) Numerisch. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.RINDEX(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

RND(*Zahl*) Numerisch. Rundet *Zahl* zu einem ganzzahligen Wert.

RND(-7.5) = -8

RPAD(*Text, Länge*) String. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.RPAD(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

RTRIM(*Text*) String. Gibt *Text* ohne rechts angehängte Leerzeichen aus.

Tipp

Werte, die in einer String-Variablen enthalten sind, haben stets die für die Variable definierte Länge. Enthält ein Feld einen Text mit weniger Zeichen, so werden die fehlenden Zeichen automatisch mit Leerzeichen aufgefüllt. Diese Leerzeichen werden nicht angezeigt und sind damit in der Datendatei nicht zu erkennen. Die Leerzeichen können jedoch sehr störend sein, wenn die Werte der Variablen als Parameter einer Text-Funktion verwendet werden. Geben Sie beispielsweise in ein Feld der Textvariablen *text1* mit der Länge 2 lediglich den Buchstaben »*o*« ein und wollen anschließend mit der Funktion

CHAR.INDEX("Coconut", text1)

nach dem Buchstaben »*o*“ in dem Wort »*Coconut*« suchen, wird die Funktion mitteilen, dass der gesuchte Wert nicht in dem Wort enthalten ist, da die Funktion tatsächlich nicht nach »*o*«, sondern nach »*o* « (»*o*« mit nachfolgendem Leerzeichen) sucht. Um dies zu verhindern, geben Sie als Suchwert die Funktion *RTRIM(text1)* an, so dass die gesamte Suchfunktion

CHAR.INDEX("Coconut",RTRIM(text1))

lautet. Diese Funktion wird den Wert *2* ausgeben, da der zweite Buchstabe von »*Coconut*« dem gesuchten Text entspricht.

RTRIM(*Text, Zeichen*) String. Diese Funktion schneidet von *Text* die letzten Zeichen ab, sofern diese mit *Zeichen* übereinstimmen. Damit ist diese Funktion der Funktion *RTRIM(Text)* sehr ähnlich. Der Unterschied besteht darin, dass die

letzten Zeichen von *Text* nicht dann gelöscht werden, wenn es sich dabei um Leerzeichen handelt, sondern genau dann, wenn diese Zeichen mit *Zeichen* übereinstimmen. *Zeichen* darf nur ein Zeichen lang sein.

RTRIM("Kaffee", "e") ergibt »Kaff«

RV.Verteilung(...) Numerisch. Mit den RV-Funktionen erzeugen Sie eine Folge von Zufallszahlen, die in der Grundgesamtheit einer vorgegebenen Verteilung entsprechen. Die RV-Funktion steht für 22 Verteilungen zur Verfügung, die Sie jeweils mit den entsprechenden Verteilungsparametern näher spezifizieren können.

Tipp

Zur Berechnung der Zufallszahlen wird ein Pseudo-Zufallszahlengenerator verwendet. Sie können eine Folge von Zufallszahlen identisch reproduzieren, indem Sie jeweils vor dem Start der Prozedur den gleichen Ausgangswert für die Berechnung der Zufallszahlen vorgeben. Siehe hierzu Kapitel 11, Abschnitt 11.2.

Für folgende Verteilungen steht eine RV-Funktion zur Verfügung:

RV.BERNOULLI(*p*) Bernoulli-Verteilung.

RV.BETA(*y, z*) Beta-Verteilung.

RV.BINOM(*n, p*) Binomial-Verteilung.

RV.CAUCHY(*k, Stufe*) Cauchy-Verteilung.

RV.CHISQ(*Freiheitsgrade*) Chi-Quadrat-Verteilung.

RV.EXP(λ) Exponential-Verteilung.

RV.F(*Freiheitsgrad1, Freiheitsgrad2*) F-Verteilung.

RV.GAMMA(*k,* λ) Gamma-Verteilung.

RV.GEOM(*p*) Geometrische Verteilung.

RV.HALFNRM(*Mittelwert, Standardabweichung*) Halbe Normalverteilung.

RV.HYPER(*Total, Ziehungen, mögl. Treffer*) Hypergeometrische Verteilung.

RV.IGAUSS(*Mittel, Skal*) Inverse Normalverteilung.

RV.LAPLACE(*Mittelwert, Stufe*) Laplace-Verteilung.

RV.LNORMAL(μ_1, σ_1) Logarithmische Normalverteilung.

RV.LOGISTIC(*Mittelwert, Stufe*) Logistische Verteilung.

RV.NEGBIN(*Erfolge, p*) Negative Binomialverteilung.

RV.NORMAL(*Mittelwert, Standardabweichung*) Normalverteilung.

RV.PARETO(*k, Stufe*) Pareto-Verteilung.

RV.POISSON(*Mittelwert*) Poisson-Verteilung.

RV.T(*Freiheitsgrade*) T-Verteilung.

RV.UNIFORM(*Min, Max*) Gleichverteilung.

RV.WEIBULL(*y, z*) Weibull-Verteilung.

SD(*Zahl, Zahl* [*, Zahl, ...*]) Numerisch. Berechnet die Standardabweichung der angegebenen (gültigen) Parameter. Es müssen mindestens zwei Parameter angegeben werden.

Die Standardabweichung wird nach der Formel

$$\text{Standardabweichung} = \sqrt{\frac{(x_i - \bar{x})^2}{n-1}}$$

berechnet. Dabei bezeichnet x_i die gültigen (nicht fehlenden) Werte, $\bar{x}$ deren Mittelwert und n die Anzahl der gültigen Werte.

Tipp

Sie können zusätzlich eine Mindestanzahl gültiger Parameter festlegen, unterhalb derer die Funktion nicht ausgeführt wird. Schreiben Sie die entsprechende Zahl, durch einen Punkt getrennt, unmittelbar hinter den Funktionsnamen. Soll die Funktion nur dann ausgeführt werden, wenn sich unter den angegebenen Parametern mindestens 20 gültige Werte befinden, schreiben Sie die Funktion in der Form *SD.20(...)*.

SIG.CHISQ(*Zahl, df*) Numerisch. Berechnet die Signifikanz für den Wert *Zahl* einer mit *df* Freiheitsgraden Chi-Quadrat-verteilten Zufallsvariablen. Dies ist die Wahrscheinlichkeit, dass eine solche Zufallsvariable einen Wert größer *Zahl* liefert.

SIG.F(*Zahl, df1, df2*) Numerisch. Berechnet die Signifikanz für den Wert *Zahl* einer mit *df1* und *df2* Freiheitsgraden F-verteilten Zufallsvariablen. Dies ist die Wahrscheinlichkeit, dass eine solche Variable einen Wert größer *Zahl* annimmt.

SIN(*Zahl*) Numerisch. Berechnet den Sinus von *Zahl*. *Zahl* ist ein Winkel, der im Bogenmaß angegeben werden muss.

SIN(1.5798) = 1, denn 1,5708 ≈ π/2.

SQRT(*Zahl*) Numerisch. Gibt den Betrag der Quadratwurzel von *Zahl* aus. Zahl muss positiv sein.

SQRT(9) = 3 SQRT(-9) ergibt einen fehlenden Wert.

STRING(*Zahl, Format*) String. Wandelt einen numerischen Wert in einen Textwert um. Die Funktion gibt die *Zahl* so, wie sie in dem vorgegebenen *Format* dargestellt wird, als Text aus.

STRING(1.234567, F8.4) ergibt »1,2346«, denn die Zahl würde in dem Format *F8.4* gerundet dargestellt werden.

STRING(1.234567, E9.2) ergibt »1,23E+0«.

STRING(12986784000, Date11) liefert den Text »27APR1994«.

STRUNC(*Text, Länge*) String. Gibt den auf die angegebene *Länge* gekürzten *Text* aus. Die Länge wird in Byte angegeben, was bei europäischen Sprachen meistens der Anzahl der Zeichen entspricht.

SUBSTR(*Text, Position*) String. Diese Funktion wird nicht mehr aktiv unterstützt. Sie wurde ersetzt durch die Funktion CHAR.SUBSTR(), die bei der Verwendung einiger Sonderzeichen und asiatischer Sprachen präziser ist.

SUM(*Zahl, Zahl* [*, Zahl, ...*]) Numerisch. Berechnet die Summe der angegebenen Werte. Sie können beliebig viele Parameter angeben, darunter müssen sich jedoch mindestens zwei gültige Werte befinden. Enthalten die angegebenen Parameter unter anderem ungültige (fehlende) Werte, werden diese nicht berücksichtigt, so dass die Summe nur für die verbleibenden gültigen Werte berechnet wird.

Tipp

Sie können eine Mindestanzahl gültiger Werte vorgeben, unterhalb derer die Funktion nicht ausgeführt wird. Schreiben Sie die entsprechende Zahl durch einen Punkt getrennt direkt hinter den Funktionsnamen. Soll die Prozedur nur für mindestens 17 gültige Werte ausgeführt werden, schreiben Sie die Funktion in der Form *SUM.17(...)*. Wird die geforderte Anzahl gültiger Werte nicht erreicht, liefert die Funktion einen systemdefinierten fehlenden Wert.

SUM(2, 7, 9) = 18 SUM(2, 7, SQRT(-9)) = 9

SUM.3(2, 7, SQRT(-9)) ergibt einen fehlenden Wert.

SYSMIS(*Variable*) Numerisch (Logisch). Ergibt 1 (wahr), wenn *Variable* einen systemdefinierten fehlenden Wert enthält, und 0 (falsch), wenn *Variable* einen gültigen oder einen benutzerdefinierten fehlenden Wert enthält. *Variable* ist der Name einer numerischen Variablen.

Tipp

Sie können die Funktion *MISSING* verwenden, wenn auch für benutzerdefinierte fehlende Werte eine 1 (wahr) ausgegeben werden soll.

TIME.DAYS(*Tage*) Numerisch. Bestimmt die serielle Zahl, die dem in Tagen angegebenen Zeitraum entspricht. Die serielle Zahl ergibt sich aus der Anzahl der Sekunden, die mit der Anzahl der angegebenen Tage äquivalent ist.

TIME.DAYS(1) = 86.400 (=24 · 60 · 60)

TIME.DAYS(*Tage*) = 86.400 · *Tage*

TIME.HMS(*Stunden*) Numerisch. Ergibt die serielle Zahl, die der angegebenen Anzahl an *Stunden* entspricht. Vgl. *TIME.HMS(Stunden, Minuten, Sekunden)*.

TIME.HMS(*Stunden, Minuten*) Numerisch. Berechnet die serielle Zahl, die der angegebenen Anzahl an *Stunden* und *Minuten* entspricht. Vgl. *TIME.HMS(Stunden, Minuten, Sekunden)*.

TIME.HMS(*Stunden, Minuten, Sekunden*) Numerisch. Ergibt die serielle Zahl, die der angegebenen Anzahl an *Stunden*, *Minuten* und *Sekunden* entspricht. Die serielle Zahl ist identisch mit dem in Sekunden ausgedrückten angegebenen Zeitraum. Keiner der Werte darf negativ sein. Die Werte für *Minuten* und *Sekunden* dürfen zudem nicht größer als 60 sein. Für *Sekunden* können Sie auch einen Wert mit Dezimalstellen angeben. Wollen Sie für *Sekunden* eine Null angeben, können Sie diesen Parameter auch fortlassen, vgl. *TIME.HMS(Stunden, Minuten)*. Um nur

Stunden zu berücksichtigen, können entsprechend die Parameter *Sekunden* und *Minuten* ausgelassen oder mit dem Wert 0 angeführt werden.

TIME.HMS(6, 23, 21.3) = 23.001,3 ($6 \cdot 60^2 + 23 \cdot 60 + 21{,}3$)

TIME.HMS(std, min, sek) = $std \cdot 60^2 + min \cdot 60 + sek$

TRUNC(*Zahl*) Numerisch. Schneidet sämtliche Dezimalstellen von *Zahl* ab und gibt den verbleibenden ganzzahligen Wert aus.

TRUNC(-3.7) = -3 TRUNC(5.9) = 5

UPCASE(*Text*) String. Gibt *Text* ausschließlich in Großbuchstaben wieder.

UPCASE("Kleinbuchstaben") ergibt »KLEINBUCHSTABEN«.

VALUE(*Variable*) Numerisch oder String. Ergibt den Wert von *Variable*, wobei auch für benutzerdefinierte fehlende Werte der ursprüngliche Wert und nicht ein fehlender Wert ausgegeben wird. Daher können Sie mit dieser Funktion benutzerdefinierte fehlende Werte in arithmetische und logische Operationen einbeziehen, ohne die Definition fehlender Werte außer Kraft zu setzen.

VALUELABEL(*Variable*) String. Gibt das Wertelabel des jeweiligen Wertes der *Variablen* aus. Ist für den jeweiligen Wert kein Wertelabel definiert, gibt die Funktion einen leeren Textwert aus. Als *Variable* muss direkt ein Variablenname angegeben werden; andere Ausdrücke sind hier nicht zulässig.

VARIANCE(*Zahl, Zahl* [*, Zahl, ...*]) Numerisch. Errechnet die Varianz der angegebenen (gültigen) Werte. Die Funktion erfordert mindestens zwei gültige Werte unter den Parametern. Ist nur ein Parameter angegeben, wird die Funktion nicht ausgeführt, und es erscheint ein entsprechender Hinweis. Sind zwei oder mehr Parameter angegeben, unter denen sich jedoch nur ein gültiger Wert befindet, wird die Funktion ausgeführt und gibt einen systemdefinierten fehlenden Wert aus.

Tipp

Darüber hinaus können Sie selbst eine Mindestanzahl gültiger Parameter vorgeben. Die Funktion liefert dann einen fehlenden Wert, wenn die angegebenen Parameter zu wenig gültige Werte beinhalten. Schreiben Sie die gewünschte Mindestzahl durch einen Punkt getrennt direkt hinter den Funktionsnamen. Soll die Funktion nur ausgeführt werden, wenn sie auf mindestens fünf gültige Werte zurückgreifen kann, geben Sie die Funktion in der Form *VARIANCE.5(...)* an.

Die Varianz berechnet sich als:

$$\text{Varianz} = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

Dabei bezeichnet x_i die gültigen Parameter, $\bar{x}$ deren Mittelwert und n deren Anzahl.

VARIANCE(1, 3, 5, 7) = $6{,}\overline{6}$

VARIANCE.4(1, 3, 5, SQRT(-7)) liefert einen fehlenden Wert.

XDATE.XXX(*Datumswert*) Numerisch. *Datumswert* ist eine serielle Zahl, die von SPSS in ein Datum oder eine Uhrzeit (oder beides zusammen) übersetzt wird.

(Zum Zusammenhang zwischen Datumsangaben und seriellen Zahlen bei SPSS siehe die Ausführungen zur Funktion *DATE*.) Die *XDATE*-Funktionen filtern aus dem (in Form einer seriellen Zahl) angegebenen Zeitpunkt einzelne Komponenten heraus. Die Funktion *XDATE.HOUR* ergibt beispielsweise die Stunde des angegebenen Zeitpunktes (zum Beispiel 23 für den Zeitpunkt 17.10.2004 23:11:05), während die Funktion *XDATE.MONTH* für diesen Zeitpunkt den Wert *10* ausgibt und damit den Monat herausgreift.

XDATE.DATE(*Datumswert*) Numerisch. Filtert aus dem angegebenen Zeitpunkt das Datum mit Tag, Monat und Jahr heraus.

XDATE.HOUR(*Datumswert*) Numerisch. Gibt die Stunde des angegebenen Zeitpunktes aus. Das Ergebnis ist ein Wert zwischen 0 und 23.

XDATE.JDAY(*Datumswert*) Numerisch. Gibt an, welchen Tag im Jahr das angegebene Datum beschreibt. Das Ergebnis ist ein Wert zwischen 1 und 366.

XDATE.MDAY(*Datumswert*) Numerisch. Gibt von dem angegebenen Datum den Tag des Monats aus. Die Werte liegen somit zwischen 1 und 31.

XDATE.MINUTE(*Datumswert*) Numerisch. Filtert aus dem angegebenen Zeitpunkt die Minutenangabe heraus. Das Ergebnis ist ein Wert zwischen 0 und 59.

XDATE.MONTH(*Datumswert*) Numerisch. Ergibt den Monat des zu *Datumswert* gehörenden Datums. Das Ergebnis ist eine Zahl zwischen 1 und 12.

XDATE.QUARTER(*Datumswert*) Numerisch. Gibt das Jahresquartal aus, in dem der angegebene Zeitpunkt liegt. Das Quartal wird als Zahl zwischen 1 und 4 ausgegeben.

XDATE.SECOND(*Datumswert*) Numerisch. Ermittelt die Sekundenangaben des durch *Datumswert* bezeichneten Zeitpunktes. Das Ergebnis ist ein Wert zwischen 1 und 59.

XDATE.TDAY(*Zeitspanne*) Numerisch. Die serielle Zahl *Zeitspanne* ist hier die Anzahl an Sekunden, die ein zu betrachtender Zeitraum umfasst. Die Funktion errechnet die Anzahl der vollen Tage, die in der *Zeitspanne* enthalten sind.

XDATE.TDAY(86399) = 0 XDATE.TDAY(86400) = 1

XDATE.TIME(*Datumswert*) Numerisch. Diese Funktion betrachtet von dem durch *Datumswert* angegebenen Zeitpunkt nur die Uhrzeit. Das Ergebnis der Funktion ist die Anzahl der Sekunden, die von Beginn des Tages (Mitternacht) bis zur betreffenden Uhrzeit vergangen sind. Bezeichnet die serielle Zahl *Datumswert* den Zeitpunkt 19.3.2003 0:01:07, liefert die Funktion den Wert 67. Der Ausgabewert liegt zwischen 0 und $86.399,\overline{9}$, denn ein Tag enthält 86.400 Sekunden.

XDATE.WEEK(*Datumswert*) Numerisch. Gibt an, in welcher Woche des Jahres das angegebene Datum liegt. Es ergeben sich daher Werte zwischen 1 und 53.

XDATE.WKDAY(*Datumswert*) Numerisch. Bestimmt den Wochentag des angegebenen Zeitpunktes. Die Wochentage werden als Zahlen zwischen 1 und 7 ausgegeben, wobei eine 1 dem Sonntag und eine 7 dem Samstag entspricht.

XDATE.YEAR(*Datumswert*) Numerisch. Filtert das Jahr aus dem angegebenen Zeitpunkt heraus. Das Jahr wird als vierstellige Zahl (wie *2005*) ausgegeben.

Die folgende Übersicht gibt die Ergebnisse der Funktionen als serielle Zahl sowie in der Form eines geeigneten Datums oder Zeitformats wieder. Als Argument für die Funktionen wurde stets die serielle Zahl des Zeitpunktes 23. Juli 2005 15:37:22,78 (13.341.512.242,78) verwendet.

Funktion	Ergebnis	Format	Darstellung
XDATE.DATE	1334156000	dd-mmm-yyyy	23-JUL-2005
XDATE.HOUR	15		
XDATE.JDAY	204		
XDATE.MDAY	23		
XDATE.MINUTE	37		
XDATE.MONTH	7	Jan, Feb, Mar...	JUL
XDATE.QUARTER	3		
XDATE.SECOND	22,78	hh:mm:ss.ss	0:00:22,78
XDATE.TDAY	154.415		
XDATE.TIME	56242,78	hh:mm:ss.ss	15:37:22,78
XDATE.WEEK	30		
XDATE.WKDAY	7	Monday, Tuesday...	SATURDAY
XDATE.YEAR	2005		

YRMODA(*Jahr, Monat, Tag*) Numerisch. Ergibt die Anzahl an Tagen, die seit dem 15. Oktober 1582 bis zu dem angegebenen Datum vergangen sind. (Der Kalender von SPSS beginnt erst mit dem 15. Oktober 1582, dem Tag, an dem der gregorianische Kalender eingeführt wurde.) Für die Argumente der Funktion gelten folgende Regeln:

Jahr. Geben Sie einen Wert an, der größer oder gleich 1582 ist. Um ein Jahr aus dem Zeitraum von 69 Jahren vor dem aktuellen Datum bis 30 Jahre nach dem aktuellen Datum anzugeben, genügt es, die beiden letzten Ziffern des Jahres zu nennen.

Monat. Für *Monat* können Sie einen Wert zwischen 1 und 13 angeben. Mit 13 beziehen Sie sich auf den ersten Monat des auf *Jahr* folgenden Jahres. Die Angaben (2004, 13, 25) und (2005, 1, 25) sind somit äquivalent.

Tag. Die Werte von *Tag* können zwischen 0 und 31 liegen. Der Wert 0 entspricht dem letzten Tag des vor *Monat* liegenden Monats. (2005, 8, 31) ist somit identisch mit (2005, 9, 0). Wenn Sie den Tag *31* für einen Monat angeben, der nur 30 Tage umfasst, so wird dies als erster Tag des folgenden Monats interpretiert. Für Februar werden die Tage 30 und ggf. 29 als 2. bzw. 3. März aufgefasst.

Wird für einen der Parameter ein ungültiger Wert angegeben, liefert die Funktion einen systemdefinierten fehlenden Wert.