Wie erstellt man eine .xlsx Listen-Spezifikation, um allen Benutzeranforderungen zu genügen?

Véronique Bourcier
Director - xxformat GmbH
c/o STARTPLATZ, Im Mediapark 5
50670 Köln
veronique.bourcier@xxformat.com

Zusammenfassung

Kommunikation zwischen verschiedenen Gruppen mit unterschiedlichem Hintergrund kann sehr schwierig und missverständlich sein. In diesem Beitrag werden sie sehen, wie man eine Spezifikation für den medizinischen Review strukturieren kann, um die unterschiedlichen Anforderungen von SAS Programmierern und nicht Programmierern in einem Dokument ab zu decken.

Schlüsselwörter: ODS EXCEL, styles.excel, RGB, CX, resolve

1 Einführung

1.1 Warum sollten Sie mit maschinenlesbaren Spezifikationen arbeiten?

Seitdem die ODS Syntax eingeführt wurde, steht in SAS neue Funktionalität zu Verfügung. Wenn man eine Report Spezifikation erstellen muss, wird solche Funktionalität allerdings entweder an den Programmierer abgegeben oder gar nicht benutzt.

Es gibt zwei Szenarien: Entweder die Person, die die Spezifikation erstellen muss, kennt die notwendigen Funktionalitäten nicht oder sie **weiß nicht, wie Anforderungen aufgeschrieben sein sollten, um diese für Programmierer und Nicht-Programmierer klar darzustellen**.

Die für die Spezifikation zuständige Person ist in der Regel **nicht daran interessiert jeden einzelnen Aspekt der Ausgabe eine Entscheidung zu treffen**. Zum Beispiel hat sie vielleicht kein Interesse am Font Name, Font Große, Output Datei Name, usw. Diese Priorität ist wahrscheinlicher: W**ie können wir Daten betrachten, um den Review schneller zu erledigen?**

In der Regel werden Outputs nicht automatisch aus der Spezifikation generiert. Aber man muss **sicherstellen, dass die Spezifikation und das finale Ergebnis überein stimmen**. Es beginnt schwieriger zu werden, wenn die Spezifikation viele Details enthält. Noch komplizierter wird es, wenn es Änderungen in der Spezifikation gibt.

Wenn man eine Spezifikation schreibt, ist es schwierig herauszufinden, wie das finale Ergebnis aussehen wird. Wenn der Output nicht wie erwartet ist, gibt es vor und zurück mit dem Programmierer. Dies benötigt Zeit und Energie von beiden beteiligten Seiten.

In diesem Beitrag sind die generierten Ausgaben Excel Dateien. Sie sind mit `ods excel` (seit SAS 9.4 verfügbar) erstellt. Die Syntax ist ähnlich der von `ods tagsets.excelxp`, aber es gibt zwei Vorteile: die Größe der Datei ist reduziert und Grafiken können hinzugefügt werden (SAS Beta Version).

1.2 Welche Aspekte behandelt dieser Beitrag?

Zuerst werde ich Ihnen zeigen, wie man den **Prozess standardisieren** kann. Dafür werden wir **Firmen spezifische Farbenpaletten und SAS Styles definieren**. Farben können eine Herausforderung sein. Ich zeige Ihnen, wie man Farben für Programmierer und Nicht-Programmierer leicht zugänglich machen kann.

Anschließend werden wir der fiktiven Person JR zuhören. Er ist zuständig für die Erstellung der Spezifikation eines Projektes namens FORECAST. JR wird uns die verschiedenen Schritte beschreiben, die er dafür machen muss. Er wird die Bedürfnisse in drei Datensätzen aufschreiben. Diese drei Datensätzen enthalten drei unterschiedliche Informationsebenen: Datei, Tab und Variablen Informationen. Anschließend wird er mehrere Makros aufrufen, um drei Excel Dateien zu generieren: ein Report, basiert auf Standard und Codelist Werten, ein Report basiert auf realen Daten und abschließend die Report Spezifikation Dokumentation.

2 Firmen-spezifische Regeln

Man braucht nicht das Rad für jedes einzelne Projekt noch einmal neu erfinden. Farben und SAS Styles können auf Firmenebene festgelegt werden.

2.1 Firmen spezifische Regeln: Farben

Farben sind Schlüsselkomponenten, wenn man von Report Darstellungen spricht. Früher haben Leute Farben, die mit Schlüsselworten wie `blue`, `red` oder `green` generiert wurden, akzeptiert. Diese Zeit ist vorbei. Wir müssen ein breites Spektrum an Farben anbieten, die leicht verfügbar und kohärent in unserer Firma sind.

Eine Lösung dazu ist eine Firmen-spezifische Farbenpalette. Sie ist in einem SAS Datensatz definiert. Der Programmierer kann leicht darauf zugreifen und einen Extrakt in einen anderen Dateityp (.xlsx, .rtf, .pdf, etc. file) für nicht Programmierer erstellen.

2.1.1 Microsoft Farbenpalette

Schauen wir in die vorgegebene Farbenpalette von Microsoft Produkten.

Hinter jeder Farbe gibt es drei Nummern, die von 0 bis 255 gehen, d.h. eine Nummer für rot, eine für grün und eine für blau. Wir sprechen von RGB (Red Green Blue) Werten.

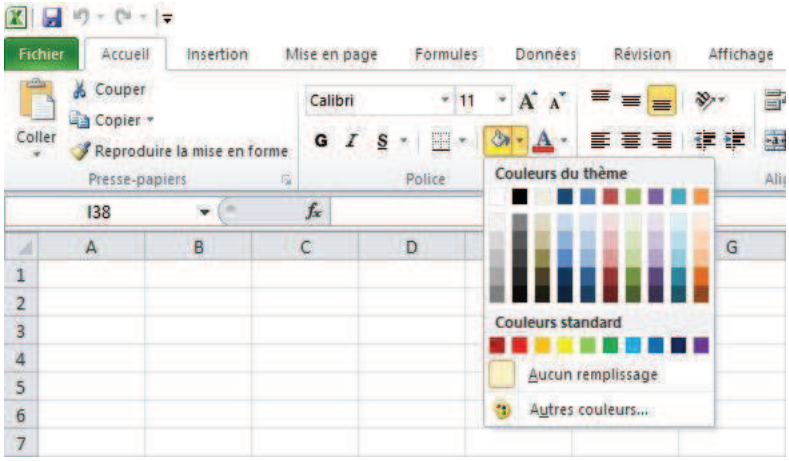


Abbildung 1: Farbenpalette in Excel

So ! Die obigen Farben entsprechen folgenden Nummern.

255,255,255	0,0,0	238,236,225	31,73,125	79,129,189	192,80,77	155,187,89	128,100,162	75,172,198	247,150,70
242,242,242	128,128,128	221,217,196	197,217,241	220,230,241	242,220,219	235,241,222	228,223,236	218,238,243	253,233,217
217,217,217	89,89,89	196,189,151	141,180,226	184,204,228	230,184,183	216,228,188	204,192,218	183,222,232	252,213,180
191,191,191	64,64,64	148,138,84	83,141,213	149,179,215	218,150,148	196,215,155	177,160,199	146,205,220	250,191,143
166,166,166	38,38,38	73,69,41	22,54,92	54,96,146	150,54,52	118,147,60	96,73,122	49,134,155	226,107,10
128,128,128	13,13,13	29,27,16	15,36,62	36,64,98	99,37,35	79,98,40	64,49,81	33,89,103	151,71,6
192,0,0	255,0,0	255,192,0	255,255,0	146,208,80	0,176,80	0,176,240	0,112,192	0,32,96	112,48,160

Abbildung 2: Farbepalette in Excel und entsprechende RGB Werte

Falls wir weitere Farben benötigen, selektieren wir diese in dem 'Personalisiert' (*Abbildung 3*) Tab. Die Werte für rot, grün und blau sind dort dargestellt.

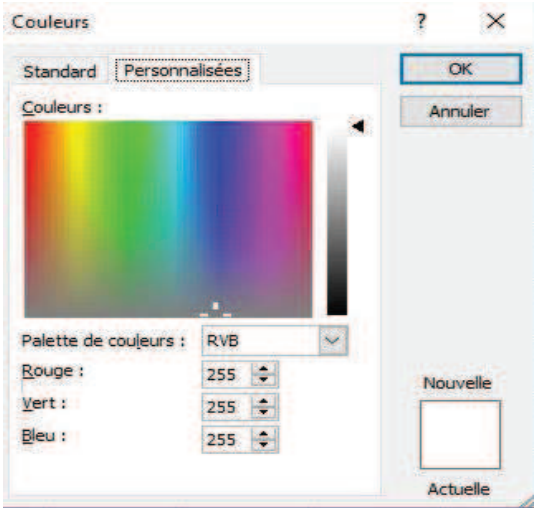
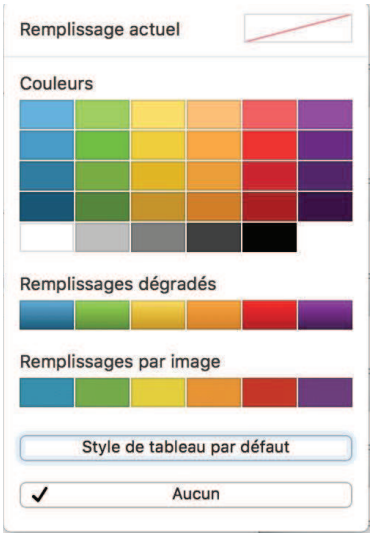


Abbildung 3: Personalisiert Farben in Excel

2.1.2 Mac Farbenpalette

Das Vorgehen ist bei Mac ähnlich.



100,178,223	156,225,89	255,224,97	255,192,114	255,95,94	157,69,184
73,155,201	110,192,56	241,209,48	255,169,58	255,45,33	108,32,133
54,125,162	121,174,61	226,184,1	236,159,46	207,35,43	85,19,108
23,87,120	87,135,38	198,147,0	209,127,21	174,25,22	60,10,73
255,255,255	191,191,191	127,127,127	64,64,64	0,0,0	

0,0,0	33,33,33	66,66,66	94,94,94	121,121,121	145,145,145	146,146,146	169,169,169	192,192,192	214,214,214	235,235,235	255,255,255
148,17,0	148,82,0	146,144,0	79,143,0	0,143,0	0,144,81	0,145,147	0,84,147	1,25,147	83,27,147	148,33,147	148,23,81
255,38,0	255,147,0	255,251,0	142,250,0	0,249,0	0,250,146	0,253,255	0,150,255	4,51,255	148,55,255	255,64,255	255,47,146
255,126,121	255,212,121	255,252,121	212,251,121	115,250,121	115,252,214	115,253,255	118,214,255	122,129,255	215,131,255	255,133,255	255,138,216

Abbildung 4: Farben in Mac

2.1.3 Konvertieren von RGB Werten in CX<rot><grün><blau> Werte

Wie können wir als Programmierer diese rot, grün und blau Nummern benutzen? Es gibt ein kleines Makro, das ich vor ein paar Jahren gefunden habe. Ich liebe es. Es sieht wie folgt aus:

```
%macro rgb (rr,gg,bb);
  CX%sysfunc(putn(&rr.,hex2.))%sysfunc(putn(&gg.,hex2.))%sy
sfunc(putn(&bb.,hex2.))
%mend rgb;
```

Diese drei Zeilen sind der Trick. Für die schwarze Farbe wird das Makro den `CX000000` Wert erstellen. Die zwei ersten Stellen sind der hexadezimal Wert für rot; die zwei folgenden Stellen sind der hexadezimal Wert für grün ; die letzte zwei Stellen sind der hexadezimal Werte für blau.

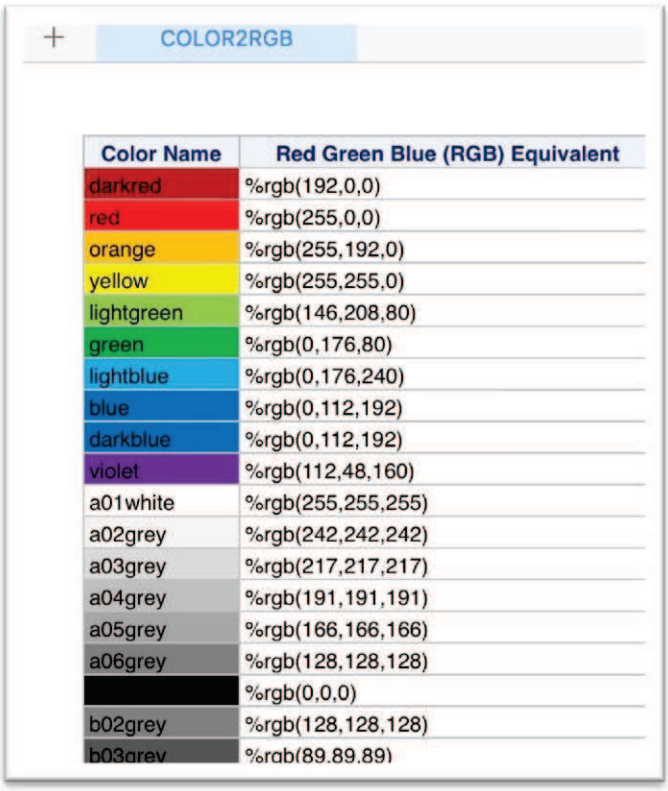
```
%let black=%rgb(0,0,0);
```

Falls Du mehr über dezimale Werte lernen möchtest, kannst du folgenden Beitrag lesen: Wie kann man Sonderzeichen mit SAS Sprache identifizieren und wegnehmen?

Da wir jetzt wissen, wie man Farben mit Nummern definiert und diese mit SAS liest, können wir diese jetzt an Nicht-Programmierer zur Verfügung stellen.

Option 1: benutze die Liste aus *Abbildung 2* (Excel)
und
Abbildung 4 (Mac).
Die Spezifikation wird drei Nummern pro Farbe enthalten.

Option 2: stelle eine Liste mit beidem, Nummer und Name zur Verfügung. Die Spezifikation wird drei Nummern oder einen Spitzname pro Farbe nach Benutzerpräferenz enthalten
(Siehe *Abbildung 5*).



COLOR2RGB

Color Name	Red Green Blue (RGB) Equivalent
darkred	%rgb(192,0,0)
red	%rgb(255,0,0)
orange	%rgb(255,192,0)
yellow	%rgb(255,255,0)
lightgreen	%rgb(146,208,80)
green	%rgb(0,176,80)
lightblue	%rgb(0,176,240)
blue	%rgb(0,112,192)
darkblue	%rgb(0,112,192)
violet	%rgb(112,48,160)
a01white	%rgb(255,255,255)
a02grey	%rgb(242,242,242)
a03grey	%rgb(217,217,217)
a04grey	%rgb(191,191,191)
a05grey	%rgb(166,166,166)
a06grey	%rgb(128,128,128)
	%rgb(0,0,0)
b02grey	%rgb(128,128,128)
b03grey	%rgb(89,89,89)

Abbildung 5: COLOR2RGB

Ich möchte in diesem Code die `resolve ()` Funktion hervorheben, die den Wert der `label` Variablen von `%rgb(...)` auf `CX...` ändern würde. Farben werden im alphanumerischen Format `color2rgb` gespeichert. Er wird mit `background` Option in der Prozedur `proc print` verwendet.

```
data biz.color2rgb;
   length fmtname $32 type $1 start $40 label $200;
   retain fmtname 'COLOR2RGB' type 'C';
   start='darkred';    label='%rgb(192,0,0)';     output;
   start='red';        label='%rgb(255,0,0)';     output;
   …
run;

data tmp.color2rgb;
   set biz.color2rgb;
   label=resolve(label);
run;

proc format cntlin=tmp.color2rgb;
run;
ods _all_ close;
ods excel file="&out_path_biz./&out_filename_biz..xlsx";
ods excel options(sheet_name='COLOR2RGB');
   proc print data=biz.color2rgb noobs;
      var start / style=[background=$color2rgb.];
      var label;
      label start='Color Name'
            label='Red Green Blue (RGB) Equivalent';
   run;
ods excel close;
```

2.2 Firmen-spezifische Regeln: Styles

Als erste Firmen-Standardreferenz haben wir über Farbpaletten gesprochen. Als zweites betrachten wir SAS Styles. Styles definieren das Layout der Ausgabe. SAS bietet den Style `Excel` an. Wir werden den als Basis verwenden, um einen neuen Style namens `xlsx1` zu erstellen.

Tabelle 1: `Standard/1_Biz/StyleDef` (Datensatz)

StyleParentName	StyleName	StyleCat	StyleAttr	StyleVal
styles.excel	styles.xlsx1	SystemTitle	background	a01white
styles.excel	styles.xlsx1	SystemTitle	foreground	blue
styles.excel	styles.xlsx1	SystemTitle	font_size	14pt
styles.excel	styles.xlsx1	SystemTitle	font_face	'Arial'
styles.excel	styles.xlsx1	SystemTitle	font_weight	bold

StyleParentName	StyleName	StyleCat	StyleAttr	StyleVal
styles.excel	styles.xlsx1	SystemTitle	font_style	roman
styles.excel	styles.xlsx1	SystemFooter	…	…
styles.excel	styles.xlsx1	Report	bordercolor	grey
styles.excel	styles.xlsx1	Report	borderwidth	2cm
styles.excel	styles.xlsx1	Report	…	…
styles.excel	styles.xlsx1	Header	just	c
styles.excel	styles.xlsx1	Header	…	…
styles.excel	styles.xlsx1	Data	cellwidth	3cm
styles.excel	styles.xlsx1	Data	…	…
…	…	…	…	…

Style Attribute können in einem Datensatz gespeichert werden z.B. `StyleDef`. Dieser Datensatz kann gelesen werden, um einen neuen Style zu erstellen. Im folgenden Beispiel enthält der Datensatz den Namen des übergeordneten Style `Excel`, den Namen des neuen Styles `xlsx1` und die Attribute, die wir ändern möchten: Titel, Fußzeile, Bericht, Kopf und Daten.

Ich habe hier die Farbe `a01white` in `StyleVal` verwendet. Es ist eine der Firmenreferenzfarben (*Abbildung 5*), die per Programm in `CXFFFFFF` umgewandelt wird.

```
data _null_;
   set _style.&in_style_dsname.;
   by StyleName StyleCat;
   if first.StyleName then
      do;
         call execute('proc template;');
         call execute('   define style ' || StyleName ||
';');
         call execute('   parent = ' || StyleParentName ||
';');
      end;
   if first.StyleCat then
         call execute('   replace ' || StyleCat || ' /');
         call execute(    StyleAttr || '=' || StyleVal );
   if last.StyleCat then
         call execute('   ;');
```

```
    if last.StyleName then
       do;
          call execute('   end;');
          call execute('run;');
       end;
run;
```

3 Es ist Zeit, die Spezifikation zu schreiben

Jetzt haben wir die Firmen-Farbpalette und Style. Ich übergebe an unsere fiktive Figur (JR), der verantwortlich für die Erstellung der Spezifikation ist.

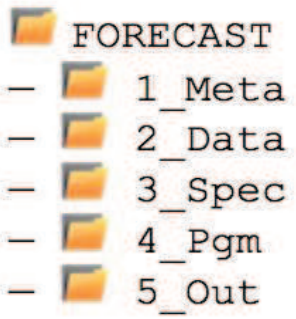


Abbildung 6: FORECAST Ordner

3.1 Die Projektordnerstruktur

Hallo, ich bin JR. Ich arbeite an dem Projekt FORECAST. In unserer Firma haben wir folgende Ordnerstruktur für Projekte (siehe Abbildung 6):

3.1.1 Metadata

In dem Ordner FORECAST/1_Meta haben wir die Metadaten der FORECAST Projektdatenbank. Es gibt einen einzelnen Datensatz, genannt MetaAll.

Tabelle 2: FORECAST/1_Meta/MetaAll (Datensatz)

DatasetName	Variable Name	Variable Type
weather	date	N
weather	alert	N
history	date	N
history	alert	N

3.1.2 Data

Im Ordner `FORECAST/2_Data` haben wir zwei Datensätze: `history` und `weather`, die die Projektdatenbank bilden. Beide haben ähnliche Inhalte für diese Demo.

Tabelle 3: `FORECAST/2_Data/weather` (Datensatz)

date	alert
…	0
…	1
…	0
…	2

3.1.3 Spezifikation

Meine Arbeit beginnt im Ordner `FORECAST/3_Spec`. Ich muss drei Datensätze aktualisieren: `SpecFile`, `SpecTab`, `SpecVar`.

Im Datensatz `SpecFile` kann ich den Dateiname(n) `FileName`, den gespeichert Ort(e) `FilePath` und der Style Name(n) für das Layout `FileStyle` angeben. Der Programmierer hat schon Standardwerte eingegeben. Normalerweise lasse ich es so wie es ist, denn es unserer Firmenstandardstruktur folgt.

Tabelle 4: `FORECAST/3_Spec/SpecFile` (Datensatz)

FileID	FilePath	FileName	FileStyle
1	/folders/myfolders/KSFE-simulation/1_Study/5_Out	report_weather	styles.xlsx1

Im Datensatz `SpecTab` spezifiziere ich den Tab Namen:

- Ich kann die Reihenfolge des Tab mit einem anderen Nummerierungssystem in der Variable `TabNum` ändern.
- Ich kann auch den Namen des Tab in `TabName` Variable ändern.
- Ich kann auch den Titel in `TabTitle1` Variable ändern, der in der ersten Zeile des Tabs angezeigt wird oder ich kann das Feld leer lassen, wenn ich keinen Titel anzeigen möchte.

Tabelle 5: `FORECAST/3_Spec/SpecTab` (Datensatz)

FileID	TabID	TabNum	TabName	TabTitle1
1	1	1	Weather	Weather Results
1	2	2	Histoy	History Results

Im Datensatz `SpecVar` liste ich die Variablen, die ich für jeden Tab haben möchte. Es gibt eine Zeile pro Variable.

- Wenn ich einen Variablennamen durch eine andere Beschriftung in der Datei ersetzen möchte, spezifiziere ich sie in der `VariableLabel` Spalte. Wenn ich sicherstellen möchte, dass ein Zeilenumbruch in der Mitte der Zeichenfolge aufgenommen wird, füge ich das Symbol `$` hinzu. Es ist das Symbol, das in unserem Unternehmen verwendet wird, um für Zeilenumbruch zu spezifizieren.

Tabelle 6: `FORECAST/3_Spec/SpecVar` (Datensatz)

TabID	Dataset Name	Variable Name	VariableLabel	Format	Background ColorFormat
1	weather	date	Date	e8601da	
1	weather	alert	Alert$Level		scale_gor
2	history	date		e8601da	
2	history	alert			scale_gor

- Ich weiß, dass Datumsangaben in unserer Datenbank als SAS Datum gespeichert sind (= nur die Maschine weiß, was es bedeutet). Also gebe ich in der `format` Variablen an, wie ich es angezeigt haben möchte. In unserer Firma verwenden wir das Format `e8601da` und da jeder daran gewöhnt ist, funktioniert es gut. In anderen Firmen könnte eine andere Variable verwendet werden, wie zum Beispiel eine `DateDisplay` Variable mit Werten im Form von `yyyy-mm-dd`.
- Im FORECAST-Projekt haben wir die Variable `alert`. Ich möchte den Hintergrund der Zelle entsprechend seinem Wert ändern. Es sollte mir für meinen Review helfen. Zum Glück hatte ich schon die gleiche Skala für ein anderes Projekt benutzt. Also existiert das Format im Standardordner schon. Ich gebe den Namen des Formats in der `BackgroundColorFormat` Spalte an.

Oh! Ich merke gerade, dass ich noch nicht die Struktur unseres `Standard` Ordners erwähnt habe. Ich hole das jetzt nach.

3.2 Die Standardordnerstruktur

Der `Standard` Ordner besteht aus drei Unterordnern.

Abbildung 7: Standard Ordner

Im Ordner `Standard/1_Biz` haben wir die `color2rgb` Tabelle, die die RGB Werte und Namen für die Farben enthält. Die Tabelle ist als SAS Datensatz und als Excel-Datei verfügbar. Ich benutze häufig `color2rgb.xlsx`. Wir haben auch den `StyleDef` Datensatz, der das Firmen spezifisches Layout beschreibt.

Im Ordner `Standard/2_Mcr` haben wir Firmenmakros.

Im `Standard/3_Cod` Ordner finden wir Kodelisten wie die, die ich für meine Hintergrundfarbe benötige. Zusätzlich haben wir eine vollständige Liste in einer Excel-Datei (siehe *Abbildung 8*). Es ist die Datei, die ich benutze.

	A	B	C
1	Format	Value	Color
2	scale_gor	0	CX00B050
3	scale_gor	1	CXF79646
4	scale_gor	2	CXFF0000

Abbildung 8: `Standard/3_Cod/format_all` (Excel)

Wenn eine Liste von Werten in den Standards fehlt, kann ich es einfach anfordern. Zum Beispiel, wenn `scale_gor` fehlen würde, würde ich eine Datei mit diesem Namen mit zwei Spalten erstellen: `Value` und `Color`. Es ist ähnlich wie *Abbildung 9*.

Value	Color
0	green
1	%rgb(247,150,70)
2	red

Abbildung 9: scale_gor (Datensatz)

Dann erzeugt ein Programm aus allen Listen eine Datei und konvertiert Farben in `CX------`. Aus `%rgb(247,150,70)` würde `CXF79646` werden.

Jetzt ist es Zeit für mich, einige Programme laufen zu lassen.

3.3 Programme: `main_include.sas` und `resetting.sas`

Wir arbeiten mit nur einem Programm, das mehrere Makros aufruft. Der erste Teil des Programms wird in eine eigene Datei (`main_include.sas`) geschrieben und der letzte Teil des Programms wird in ein anderes Programm (`resetting.sas`) geschrieben. Beide Programme werden mit der `%include` Anweisung eingelesen.

Das `main_include.sas` Programm enthält Informationen über Bibliotheken, Format- und Makro-Kataloge. Wir haben auch einen Makroaufruf zur Erstellung des Firmen Style auf Basis der Referenztabelle `StyleDef` (siehe *Abschnitt 2.2*).

Im `resetting.sas` Programm haben wir Anweisungen, um Bibliotheken, SAS Format- und Makro-kataloge zurückzusetzen. Die Anweisung **`%sysmstoreclear;`** läuft zuerst. Dann wird der `libname` für Makro(s) zurückgesetzt. Style(s), die in `main_include.sas` erstellt wurden, werden gelöscht und der Titel wird durch die `title` Anweisung leer gelassen.

Die Programme sind auf Projektebene im Order `FORECAST` gespeichert. In der Regel werden sie durch Programmierer vorbereitet. Hier ist nichts für mich zu tun!

3.4 Programm: `call_pgm.sas`

Der verbleibende Teil des Programms ist in `call_pgm.sas` verfügbar, die im Ordner `FORECAST/4_pgm` zur Verfügung steht. Es besteht aus Makroaufrufen. Ich kann jederzeit mit einem Stern `*` direkt nach dem `%` Zeichen, Aufrufe aktivieren oder deaktivieren.

3.4.1 Erstellen ein Datensatz aus die drei Spezifikation Datensatz

Das erste Makro, `%gen_spec`, erstellt einen neuen Datensatz im Ordner `FORECAST/3_spec`, der `SpecFile`-, `SpecTab`- und `SpecTab`-Dateien mit den `FileID`- und `TabID`-Schlüsselvariablen kombiniert. Es fügt auch den Variablentyp mit dem `MetaAll` Datensatz hinzu.

Die Makroparameter geben den Namen und den Speicherort der Dateien an. Einige der in `call_pgms.sas` aufgerufenen Makros verwenden temporäre Datasets, die standardmäßig in der Arbeitsbibliothek gespeichert sind. Das Ziel dieser temporären Datasets kann mit dem Makroparameter `out_tmp_lib` geändert werden. Ich benutze aber immer die gleichen Werte.

```
%gen_spec               (in_spec_lib        = spec
                        ,in_spec_file       = spec_file
                        ,in_spec_tab        = spec_tab
                        ,in_spec_var        = spec_var
```

```
,in_meta_lib          = meta
,in_meta_dsname       = meta_all
,out_spec_lib         = spec
,out_spec_dsname      = spec
,out_tmp_lib          = tmp
);
```

3.4.2 Erstellung eines Reports mit Dummy Daten

Das zweite Makro, `%gen_report_dummy`, erzeugt eine Excel-Datei mit dem finalen Design, das aber auf Dummy-Daten basiert.

- Wenn das Format `e8601da` in der Spezifikation verwendet wird, weiß das Programm, dass ein SAS-Datum erwartet ist. Es wird das Datum mit der `today()`-Funktion setzen.
- Wenn ein firmenspezifisches Format (Liste von Werten) in der Spezifikation angegeben ist, verwendet das Programm die Werte von dieser Liste.
- Ein zusätzlicher Tab könnte hinzugefügt werden, um die auf die Daten angewendeten Skalen anzuzeigen.

```
%gen_report_dummy(in_cod_lib          = cod
                 ,in_spec_lib         = spec
                 ,in_spec_dsname      = spec
                 ,out_suffix          = _dummy
                 ,out_tmp_lib         = tmp);
```

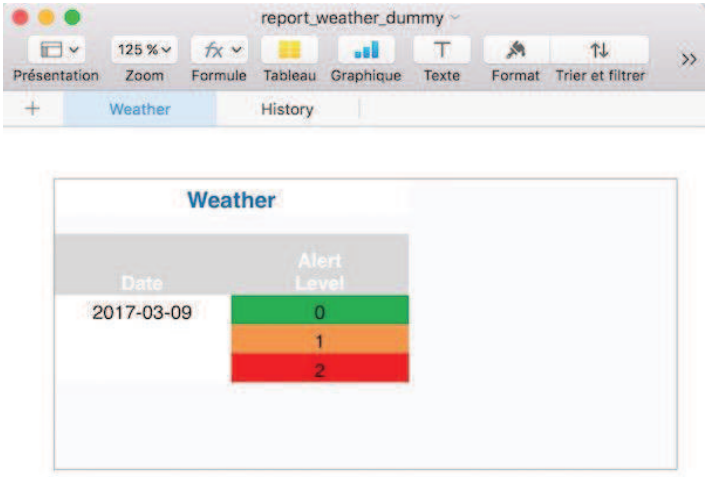


Abbildung 10: `FORECAST/5_Out/report_weather_dummy` (Excel)

Normalerweise brauche ich nichts in den Makroparametern, das durch Programmierer vorbereitetet war, ändern. Ich muss nur wissen, wo die Datei gespeichert ist!

Es ist im Datensatz `spec_file` angegeben. Die Information ist auch im Standardausgabeziel des Programms (HTML- oder Text-Datei abhängig von der SAS-Installation) angegeben.

In der Tat liefern die Makroparameter nur Details zu den Ein- und Ausgabedaten wie zum Beispiel Kodelisten- und Spezifikationsdatensätze.

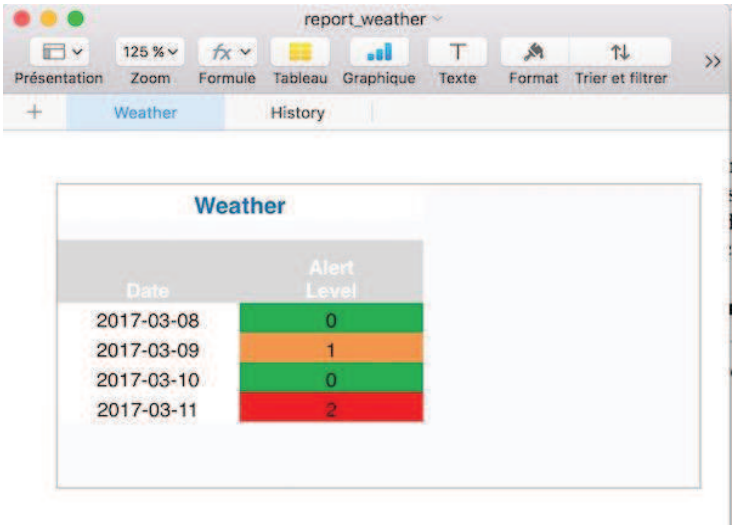
Wir erstellen insgesamt drei Reporte mit `call_pgm.sas`. Der Name dieser Dateien beginnt genauso, wie in `SpecFile`-Dataset definiert. Der Parameter `out_suffix`

wird verwendet, um **ein Suffix zu definieren**. Damit sind die **Dateinamen unterschiedlich**.

3.4.3 Erstellen eines Report mit realen Daten

Sobald ich mit dem Report mit Dummy Daten zufrieden bin, kann ich das nächste Makro ausführen, das mit echten Daten arbeitet. Es ist das Makro `%gen_report`.

```
%gen_report       (in_cod_lib          = cod
                  ,in_spec_lib         = spec
                  ,in_spec_dsname      = spec
                  ,in_data_lib         = data
                  ,out_suffix          =
                  );
```



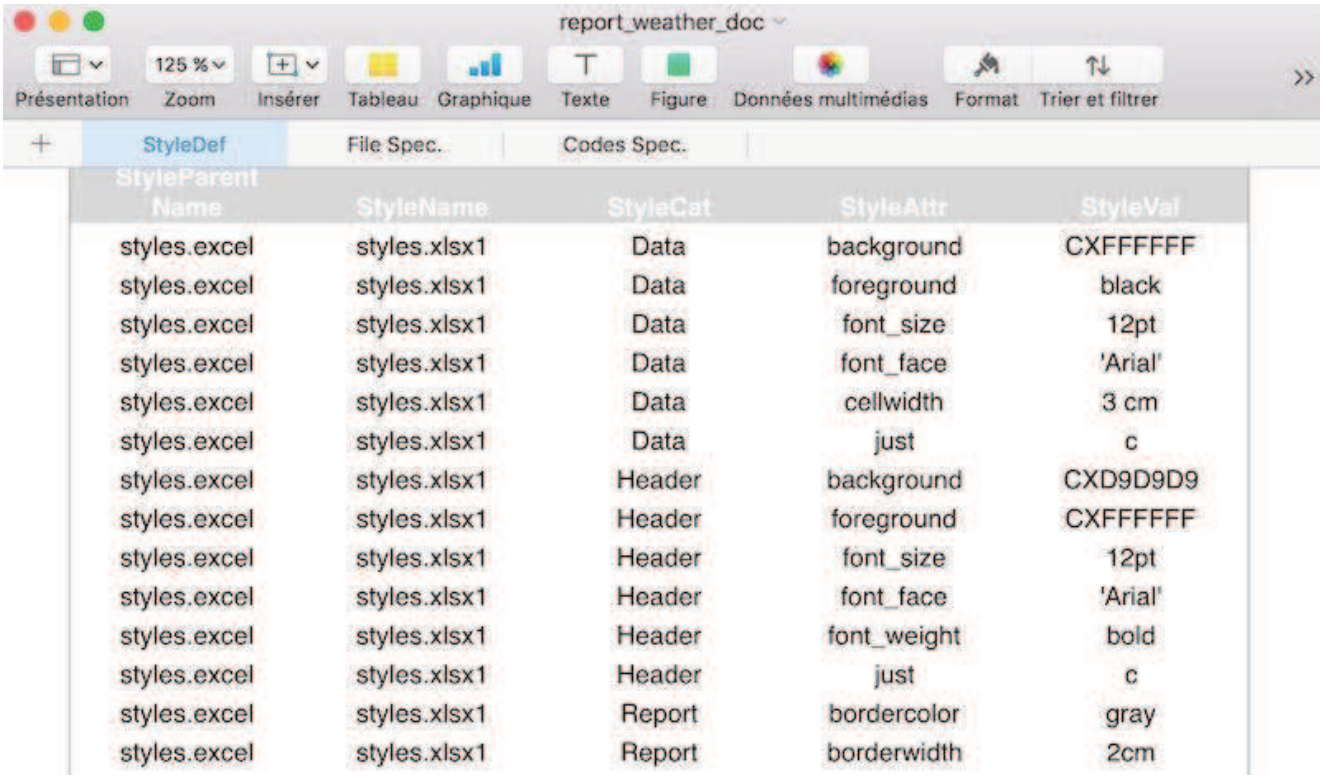
Der Code dieses Makros ist dem vorherigen sehr ähnlich. Es ist nur kürzer, weil reale Daten verwendet werden. Es besteht keine Notwendigkeit hier einen Dummy-Datensatz zu erstellen.

Abbildung 11: `FORECAST/5_Out/report_weather` (Excel)

3.4.4 Erstellen der Report Spezifikationsdokumentation

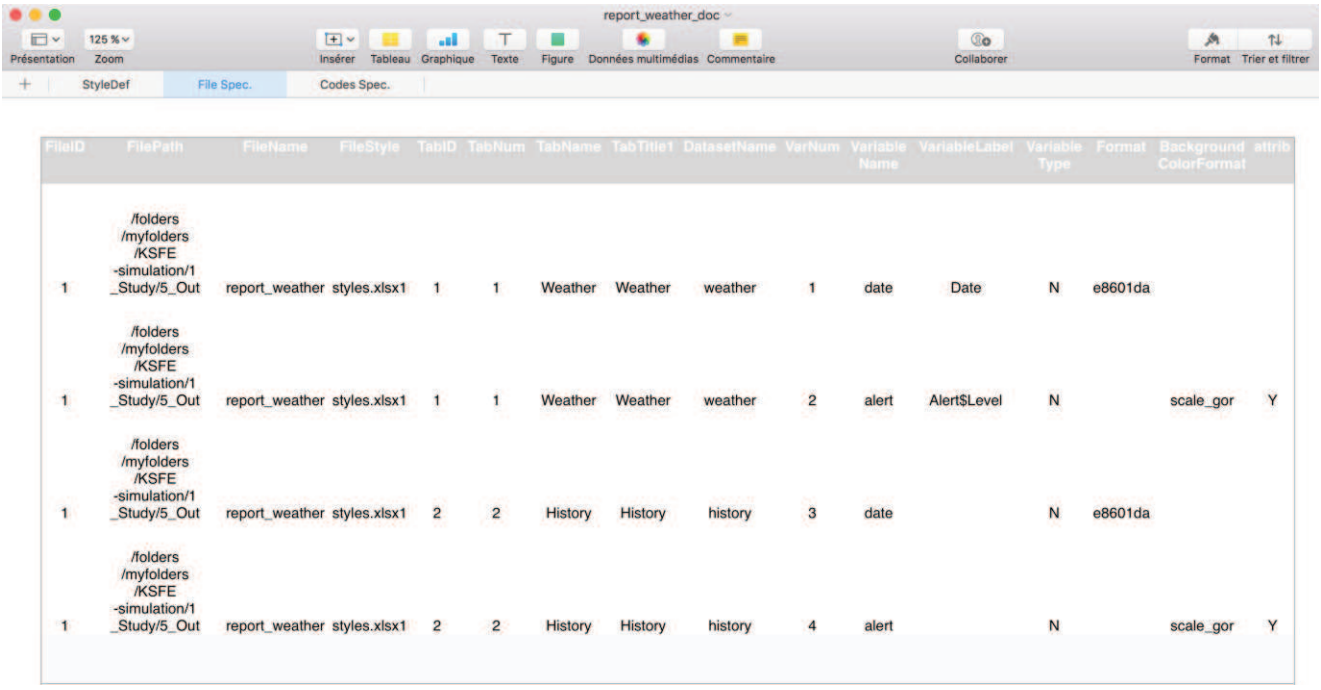
Die Dokumentation enthält die Informationen aus dem `spec` Datensatz, sowie die referenzierten Kodeliste(n) und Style-Standard-Datensätze.

```
%gen_report_doc   (in_style_lib        = style
                  ,in_style_dsname     = StyleDef
                  ,in_cod_lib          = cod
                  ,in_cod_dsname       = format_all
                  ,in_spec_lib         = spec
                  ,in_spec_dsname      = spec
                  ,out_suffix          = _doc
                  ,out_tmp_lib         = WORK
                  );
```



StyleParent Name	StyleName	StyleCat	StyleAttr	StyleVal
styles.excel	styles.xlsx1	Data	background	CXFFFFFF
styles.excel	styles.xlsx1	Data	foreground	black
styles.excel	styles.xlsx1	Data	font_size	12pt
styles.excel	styles.xlsx1	Data	font_face	'Arial'
styles.excel	styles.xlsx1	Data	cellwidth	3 cm
styles.excel	styles.xlsx1	Data	just	c
styles.excel	styles.xlsx1	Header	background	CXD9D9D9
styles.excel	styles.xlsx1	Header	foreground	CXFFFFFF
styles.excel	styles.xlsx1	Header	font_size	12pt
styles.excel	styles.xlsx1	Header	font_face	'Arial'
styles.excel	styles.xlsx1	Header	font_weight	bold
styles.excel	styles.xlsx1	Header	just	c
styles.excel	styles.xlsx1	Report	bordercolor	gray
styles.excel	styles.xlsx1	Report	borderwidth	2cm

Abbildung 12: `FORECAST/5_Out/report_weather_doc` (Excel) - Style



FileID	FilePath	FileName	FileStyle	TabID	TabNum	TabName	TabTitle1	DatasetName	VarNum	Variable Name	VariableLabel	Variable Type	Format	Background ColorFormat	attrib
1	/folders /myfolders /KSFE -simulation/1 _Study/5_Out	report_weather	styles.xlsx1	1	1	Weather	Weather	weather	1	date	Date	N	e8601da		
1	/folders /myfolders /KSFE -simulation/1 _Study/5_Out	report_weather	styles.xlsx1	1	1	Weather	Weather	weather	2	alert	Alert$Level	N		scale_gor	Y
1	/folders /myfolders /KSFE -simulation/1 _Study/5_Out	report_weather	styles.xlsx1	2	2	History	History	history	3	date		N	e8601da		
1	/folders /myfolders /KSFE -simulation/1 _Study/5_Out	report_weather	styles.xlsx1	2	2	History	History	history	4	alert		N		scale_gor	Y

Abbildung 13: `FORECAST/5_Out/report_weather_doc` (Excel) - Spec.

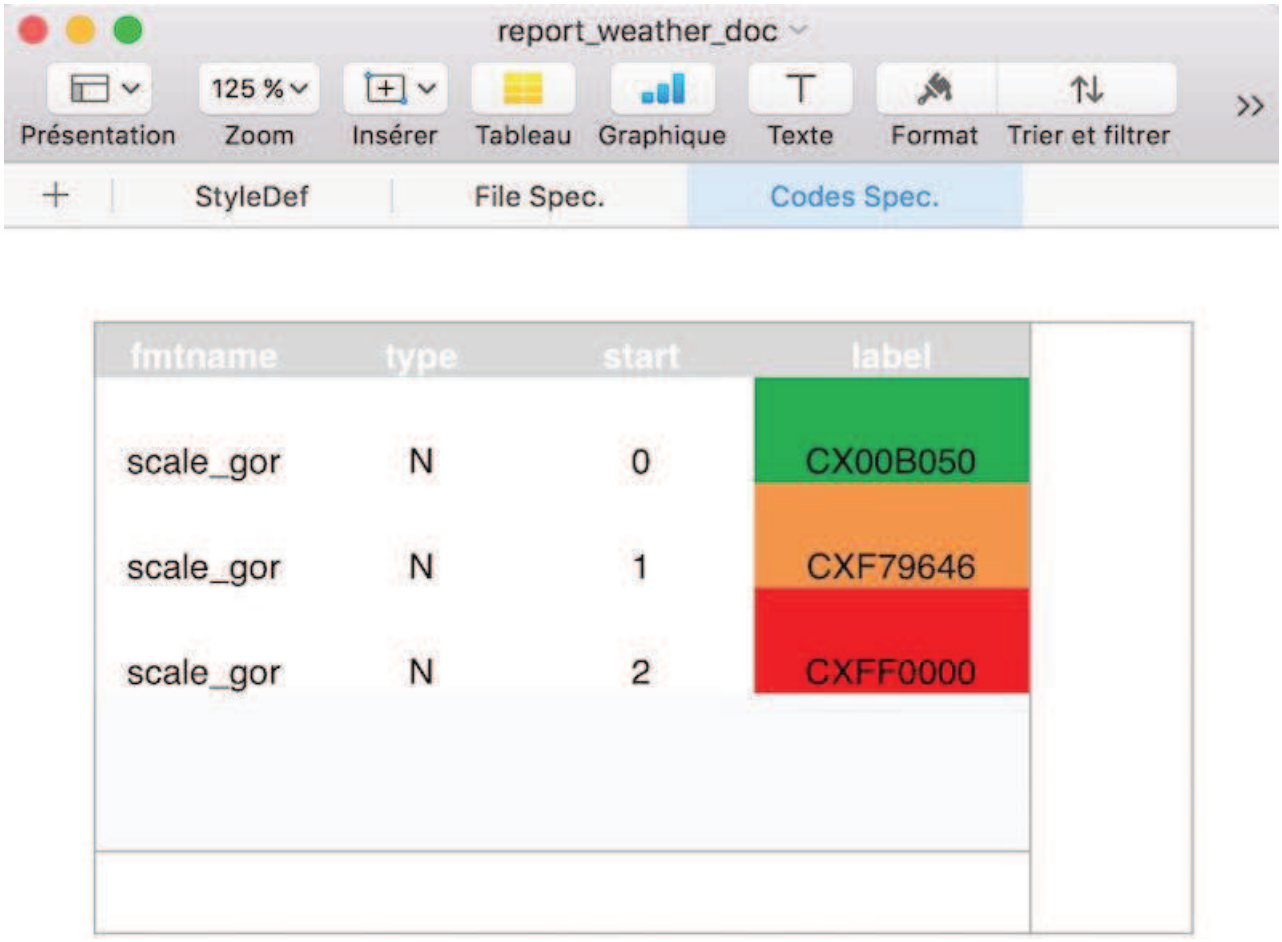


Abbildung 14: `FORECAST/5_Out/report_weather_doc` (Excel) - Codes

Das ist es! Ich hoffe, sie können jetzt nachvollziehen, wie ich meine Anforderungen spezifiziere. Der Hauptvorteil hierbei ist, dass sie maschinenlesbar sind. Auf diese Weise kann ich einen Report mit Dummy-Daten generieren, um das Ergebnis frühzeitig zu visualisieren und die Spezifikation entsprechend zu aktualisieren. Es vermeidet viel hin und her mit dem Programmierer. Beide müssen sich keine Sorgen um Synchronisierungsprobleme, zwischen der Spezifikation und den Programmen, machen. Es macht die Validierung einfacher und zuverlässiger.
Tschüss, JR

4 Nächste Schritte

Dieser Beitrag ist eine Einführung in das Konzept der Datensatz basierten Spezifikation, von der sowohl Programmierer als auch Nicht-Programmierer profitieren können. Es gibt Ihnen eine Basis um Ihre eigene Lösung zu entwickeln.

Um mit solch einem System zu arbeiten, benötigen wir so viele standardisierte Definitionen auf Firmenebene wie möglich:

- Datensätzen mit Farb- oder Style Definitionen
- Vorbelegte Makroparameterwerte
- Vorlagenprogramme, die kopiert und angepasst werden können

4.1 Mapping

Dieses Papier beschränkte sich auf Szenarien, in denen Metadaten und Daten ein Eins-zu-eins Verhältnis bilden. Sie können dies durch die Einführung von Mapping-Regeln aufbauen. Zum Beispiel brauchen Sie es für:

- erstellen einer Variablen aus vier Variablen und erklären, wie sie gruppieren werden,
- erstellen einer Datei aus verschiedener anderer Dateien,
- aufteilung der Information einer Variablen in zwei z.B. `yyyy-mm-ddThh:mm` in `yyyy-mm-dd` (Datum) und `hh:mm (Zeit)`,
- wählen einer Teilmenge eines Datensatzes aus z.B. nur Patienten mit einer bestimmten Randomisierungszahl.

Der Schlüsselpunkt hierbei ist, dass die Eingangsvariablen einerseits und auf der anderen Seite die Ausgangsvariablen vorbesetzt werden, wobei ein Zwischenfeld die Zuordnung von Regeln reflektiert.

Diese Mapping-Regeln können mit der Erfahrung von Projekten verallgemeinert und als Vorlagen gespeichert werden. Um eine gewisse Flexibilität zu ermöglichen, sollten zwei / drei Alternativen zur Verfügung gestellt werden, um die häufigsten Fälle zu abzudecken.

Für ganz neue Mapping-Regeln würde diese Technik ein gemeinsames Treffen zwischen dem Programmierer und dem Nicht-Programmierer erfordern.

4.2 ODS Excel Optionen

In diesem Beitrag betrachteten wir nur grundlegende ODS-Funktionalitäten. Allerdings ist die Liste der Möglichkeiten breiter. Jede Firma sollte Punkte, die für ihre eigenen Bedürfnisse relevanten Punkte, berücksichtigen. Nur um ein paar zu nennen:

- Hintergrundfarbe für Intervalle definieren
- Hintergrundfarbe über mehrere Spalten definieren
- Rahmenfarbe und Breite definieren
- Spaltenbreite definieren
- Tabulaturfarbe definieren (`tab_color=`)
- `Flyover`-Text (Excel Kommentar) hinzufügen, um eine Regel zu beschreiben, wenn einige Werte farbig dargestellt werden
- Hyperlink hinzufügen
- Filter automatisch hinzufügen
- Header gefrieren
- Zwei Header hinzufügen: eine für Variablen Label und eine für Variablennamen
- Horizontale und vertikale Ausrichtung (links, rechts, ausrichten, umbrechen, …) definieren.
- Zellen kombinieren
- Text drehen

- Druckbereich (Rand, Seitennummerierung, Ausrichtung, Druckbereich, Maßstab, …) definieren
- Grafik definieren

Sie können auch generische Ansätze definieren:

- Formatierung, die bei der Verwendung von Datum / Uhrzeit im Dateinamen gelten soll
- Darstellung von fehlenden numerischen Werten
- Darstellung von nicht gerundeten Werten

4.3 ODS-Format

Der Beitrag ist auf Excel beschränkt. Jede Art von ODS-Formaten (ODS CSV, ODS HTML, ODS RTF, ODS PDF in Verbindung mit ODS LAYOUT, ODS DOCUMENT und so weiter) hat eigene Optionen und Bedürfnisse, die weitere Untersuchungen erfordern.