Programmes used for constructing spell data

Dale T. Mortensen Centre, Aarhus University — Version 15/05/2025

Henning Bunzel and Mads Hejlesen

Department of Economics and Business Economics, Aarhus University, email hbunzel@econ.au.dk

Contents

Abstract This book documents programmes used to construct spell datasets

Chapter 1
Overview of spell data code

The spell datasets are generated on project 702728 and all code, pdf, and datasets are stored in directory *SPELL*. Table 1 shows the content of SPELL directory.

Table 1 SPELL directory

NAME	CONTENT
ARCHIVE	A copy of the last versions of spell code and datasets
DATASET	The final and intermediate datasets. These datasets are documented in LMDG datasets with the name SPELL_'spelldataset'or SUBSPELL_'subspelldataset'
DOCS	Technical notes and utilities
LIBNAME	Contains all subdirectories with datasets and program memes used for the current version.
PDF	All PDF files used for external documentation. Pdf files used for testing and internal documentation are stored in the relevant programme subdirectory.
PROGRAMMES	All codes used to generate the spell datasets. There exists a subdirectory for each task used in the construction of the spells.
TIL_ECONAU_DB	A copy of datasets to be shared with other projects

The directory PROGRAMMES has a subdirectory for each task in the construction of spells.

Table 2 shows the subdirectories of directory programmes .

And each subdirectory in the PROGRAMMES directory have the subdirectories shown in table 3. These subdirectories contain all jobs to construct, check and analyze the spells in this task.

Table 4 shows the spell datasets generated by project SPELL

Table 2 PROGRAMMES directory

Name	Title
EMPLOYMENT_SPELLS	The jobs create employment spells using datasets created by jobs in RAW_EMPLOYMENT_OBS
E_U_N_SPELLS	The jobs create spells showing transitions between employment, unemployment and NOT in labour market. The jobs use datasets created by jobs in EMPLOYMENT_SPELLS and in UNEMPLOYMENT_SPELLS.
FIRM_ID	The jobs create the firm_ids and workplace_ids used in spells.
RAW_EMPLOYMENT	The jobs create raw employment records using datasets *CONESR*, *RAS_CONESR* and *BFL*
PERSONS_SPELLS	The jobs create spells for persons with permanent address in Denmark
TIMELON	The jobs create and estimate of hourly wage and number of hours work.
UNEMPLOYMENT_SPELLS	The jobs create unemployment spells using datasets *LC*, *SHSS*, *OF*.

Table 3 Entries in SPELL subdirectories

NAME	CONTENT
ANALYSIS	jobs used to test and analyse source and final datasets
BEFORE2008	programmes processing data before 2008
CHECKS	programmes checking data for a new year
FROM2008	programmes processing data from 2008

Table 4 SPELL datasets

Name	Title	programme
SPELL_E	*Spells, Primary jobs with overlaps.*	
SPELL_E_N_YEARLY_SMOOTH	*Yearly spells, E-, N- spells without overlaps.*	
SPELL_E_U_N	*Spells, E-U-N spells with overlaps.*	
SPELL_E_U_N_YEARLY	*Yearly spells, E-, U-, N- spells with overlaps.*	
SPELL_E_U_N_YEARLY_SMOOTH	*Spells, E-, U-, N- spells with overlaps.*	
RAW_UNEMPLOYMENT	*Spells, Unemployment with overlaps.*	
SPELL_U	*Spells formed from RAW_UNEMPLOYMENT* without overlaps and gaps.	
SUBSPELL_E_SEC	*Subspells with secondary jobs in SPELL_E*	
SUBSPELL_LOEN	*Subspells with wage and salary for SPELL_PRIM_EMPLOY.*	
SUBSPELL_MULT_DSKOD	*Subspells with multiple DSKOD for employment spells*	

Chapter 2
Raw Employment Spells

1 Introduction

The construction of employment spells consists of two major tasks. The first task constructs *RAW_EMPLOYMENT, Intermediate dataset for employment spells formed from CONESR, RAS_CONESR and BFL*, which is used in the second task to construct *SPELL_E, Spells, Primary jobs with overlaps.* and *SUBSPELL_E_SEC, Subspells with secondary jobs in SPELL_E*.

RAW_EMPLOYMENT contains all data needed to construct the employment spells. It is constructed from *CONESR*, *RAS_CONESR* and *BFL*. The content has been checked and harmonized over the years and transformed into a monthly dataset. In a given month a worker may have one or more employments.

The statistical units are *workplaces* and *persons*. *SENR* identifies the unit that reports taxable income.

We distinguish between *SPELL_E, Spells, Primary jobs with overlaps.* and *SUBSPELL_E_SEC, Subspells with secondary jobs in SPELL_E*, where *primary employment* in a given month for a given worker is defined as employment at the firm at which the worker spends *most working hours, aggregated over the current and the two following months*. The primary employment spells are merged with *SPELL_U, Spells formed from RAW_UNEMPLOYMENT* without overlaps and gaps. resulting in *SPELL_E_U_N*.

2 Dataset *RAW_EMPLOYMENT*

This section describes how we extract and prepare monthly raw employment records from *LMDG* datasets *CONESR*(19884-2005), *RAS_CONESR*(2006-2007), and *BFL*(2008-) in order to have a unified dataset containing all relevant employment records from which we can then compute the employment spells.

Table 5 List of Variables in Raw Employment

Variable name	Title	Description
AAR	*Year for the reference period of the observation*	
ALDER	*Age, ultimo year*	
ANS_TYPE	*SPELL, Type of employment, 1: full-time, 2: employment shorter than a year, 3: two or moreemployments in a year.*	Only for 1984-2007. It is computed from ANSFRA, ANSTIL, HELARKOD, EJANSNOV, ANS2111k
ANS_ULTIMO_NOVEMBER	*In SPELLS a boolean defining if a spell of employment includes end-of-november defined as the day, 1121.*	If employed November 21.
BOOL_JOB_ATP_SATS_KODE_MISSING	*A boolean defining if the ATP rate code is missing.0: the value is valid1: the value is missing.*	If ATP contributions are missing
B_JOB_ATP_BELOEB	*ATP contributions converted to SATS B for all years.*	
BOOL_JOB_LOEN	*A value characterizing the value of JOB_LOEN_BELOEB_SMAL.*	Defines if *JOB_LOEN_BELOEB* is zero, negative or positive
CVRNR	*Encrypted legal unit identifier. Data break 1994*	
ERROR_DATES	*SPELL, identifies the changes of month and day in ANSFRA and ANSTIL in dataset CONESR*	
JOB_ARB_NR	*Unique 10-digit ID number for an establishment of a job*	
JOB_CVRNR	*CVRNR on tax reports for the job, see CVRNR.*	
JOB_CVRNR_ESR	*CVR match on the SPELL dataset using SENR in ESR_SE_CVR*	
JOB_LOEN_BELOEB_SMAL	*Basic wage amount comprising earned income without ATP contribution and fringe benefits*	
JOB_LOEN_TIMER_BEREGNET	*SPELL, Estimated number of working hours*	
JOB_LOEN_TIMER_BFL	*SPELL, Number of working hours in BFL*	
JOB_SENR	*SENR on tax reports*	
JOB_SLUT_DATO	*In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*	
JOB_START_DATO	*In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*	
PNR	*Person identifier, encrypted CPRNR number*	

RAS_CONESR is a selection from *RAS* covering the years 1985-1986. The *LMDG* datasets exist in project 702728 folder ECONAU.

The variables in *RAW_EMPLOYMENT* are shown in Table 5 and in the documentation of the dataset.

The core employment variables are shown in Table 6

The structure and quality of the datasets (*CONESR*, *RAS_CONESR*) and *BFL* are very different. Therefore two different programmes are used.

Table 6 Core EMPLOYMENT variables

Name	Title	Description
JOB_LOEN_TIMER	*Number of paid wage hours in a job*	
PNR	*Person identifier, encrypted CPRNR number*	
JOB_START_DATO	*In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*	
JOB_TIME_LOEN_SMAL	*Average hourly earnings in November employment. Data break 2008.*	
JOB_SLUT_DATO	*In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*	

2.1 Period 1985-2007

CONESR covers 1980-2005. The *CONESR* variables for the years 2006-2007 have been added to the usual *RAS*. For the spell project the *CONESR* variables have been extracted from *RAS* for the years 2006-2007. These variables are from the yearly *taxable income report*. These datasets do not contain *JOB_START_DATO*, *JOB_SLUT_DATO*, *JOB_LOEN_TIMER*, but a set of variables HELARKOD, ANSFRA, and ANSTIL which can be used to construct these variables.

JOB_START_DATO, JOB_SLUT_DATO and *JOB_LOEN_TIMER_BEREGNET* are imputed, see below. A lot of obvious errors are fixed in HELARKOD, ANSFRA, and ANSTIL and missing data is imputed. Finally the result is compared with the *ECONAU* dataset *MIAPNRM* for the period 1997-2007. *MIAPNRM* is a monthly dataset with an indicator for employment in a month for a person at a firm. Start-date and End-date are made consistent with the *MIAPNRM* data.

JOB_LOEN_TIMER_BEREGNET is imputed from the ATP contribution which is a step function of hours worked, see task TIMELOEN. The number of hours worked is used to identify the *primary employment* in cases where a person works at several establishments. For selecting primary employment, the number of hours worked is preferred to payment received, because payment in a period often includes special payments or payment earned in previous periods.

All records have a *SENR*.

The SPELL population is all persons 15 years and above living in Denmark. The relevant persons are selected in task EMPLOYMENT programmes.

Five programmes construct the *RAW_EMPLOYMENT* dataset. They are described below. Each job has several blocks.

The SAS libraries used for version 2021_v1 are listed in \ SPELL\Library\library.sas. This SAS code also contains macros for:

- PDFSTART, initialize a new pdf document, defines colours, SAS PROC template which defines format 10.0 for crosstabfreqs and proc onewayfreqs allowing all digits for our large datasets.
- PDFEND
- DeleteDataset

- DeleteDatasetOnOff
- CreateTempDatasetOnOff If ON a dataset is copied to TEMP
- ComputeDecentiles Create a dataset with percentiles for a varible from a dataset
- ComputeDecentilesByVar Create a dataset with percentiles for a varible BY another variables from a dataset
- TransposeComputedPercentiles

2.1.1 _00_driver_before 2008_2021_v1

The driver runs the programmes needed for generating the dataset *RAW_EMPLOYMENT*. The driver job can be edited using the following SAS macro variables:

- *INOBS*, controls how many records are read: max or an integer number.
- *BOOL_DELETE*, deletes temporary files after each step.
- *BOOL_PDF_ON*, controls generation of PDF files for description and debug.
- LIB_IN and NAME_IN_DATA define SAS library and file name for in data.
- LIB_OUT, library for storing permanent output data. Test and temporary data are stored in \temp. Files in \temp must be deleted manually when they are not needed in the job anymore.
- START_YEAR and END_YEAR determine the period for which the spells are generated

2.1.2 01-extracting_conesr_before 2008_2021_v1

The structure of the records in *CONESR* and *RAS_CONESR* does not differ, but they cover two different time periods.

- STEP 1 Extracts variables from *CONESR* and *RAS_CONESR*, see table 5.
- STEP 2 Computes *B_JOB_ATP_BELOEB*,
- STEP 3 Checks ANSFRA and ANSTIL. Records with no contradictions in the values of the variables ANSFRA, ANSTIL, EJANSNOV, HELARKOD, ANS2911K have ERROR_DATES set to 0 and *ANS_TYPE* is set to 1,2,3, see below.
  For all observations with a contradiction the most likely values for ANSFRA, ANSTIL have determined. ANSFRA, ANSTIL are not changed but four new variables are computed: NUM_ANSFRADD, NUM_ANSFRAMM, NUM_ANSTILDD, NUM_ANSTILMM. These variables get a value for all records, hence ANSFRA and ANSTIL cannot be used later on to test for missing values. The editing is done here to make sure that the best values for NUM_ANSFRA and NUM_ANSTIL are used for determining *ANS_TYPE* in STEP 4 and to make sure that the correct values are used in STEP 4. All changes are documented by the variable ERROR_DATES, see Table **??**
- STEP 4 The variables *ANS_TYPE*, *JOB_START_DATO* and *JOB_SLUT_DATO* are computed.

The final output dataset is *RAW_EMPLOYMENT_CONESR_RAS* in directory RAWEB07. The debug datasets are *CONS_NO_ANS_TYPE* and *NO_JOB_ATP_SATSKODE*.

See the documentation in the SAS code for temporary files generated.

It contains one macro:

Macro weekmax (aar) The macro determines the highest week number in a year. Currently it is not used.

STEP 1. Extracts records from ***CONESR*** **and** ***RAS_CONESR*.** The dataset *_01_raw-employment_1* is constructed from *CONESR* and *_01_raw-employment_ras_1* from *RAS_CONESR* by selecting variables for identifying persons and workplaces (*PNR*, *SENR*, *ARB_NR*, *DSKOD*) wage, atp contributions, and information about employment spells (HELARKOD,ANS2911K,ESANSNOV,ANSFRA and ANSTIL.)

_01_raw-employment_ras_1 is used as indata and *_01_raw-employment_ras_2* is created by changing ANSFRA and ANSTIL. ANSFRA and ANSTIL have type 'character $4' in 1980-2005, but type 'numeric, 4' in *RAS* 2006-2007. It is converted to character $4.

_01_raw-employment_2 is constructed by appending *_01_raw-employment_1* and *_01_raw-employment_ras_2* and computing boolean variables for debugging and description:

- *BOOL_JOB_LOEN*
- *BOOL_JOB_ATP_SATS_KODE_MISSING*

STEP 2. Computes the variable B_JOB_ATP_BELOEB The step has *_01_raw-employment_2* as indata and generates *_01_raw-employment_3*. For 1980-1990 different variables and rates are used over time, see *Field Note: ATP*. The ATP contribution rate changes in 1982, 1988, 1990, 1996, and 2006.

The \temp file NO_JOB_ATP_SATS_KODE contains records with missing values for *JOB_ATP_SATS_KODE*.

- For the period 1980-1981 *JOB_ATP_BELOEB_SATSA* is converted to SATSB.
- For the period 1982-1987 the programme uses *JOB_ATP_BELOEB_SATSA* witch is equal to SATSB.
- For the period 1988-1989 SATSB plus SATSA converted to SATSB.
- For the period 1990-1995 SATSB plus SATSA converted to SATSB. The rates have changed.
- For the period 1996-2005 the programme uses *JOB_ATP_BELOEB* and *JOB_ATP_BIDRAG_SATS_KODE*.
- For the period 2006-2007 the programme uses *JOB_ATP_BELOEB* and *JOB_ATP_BIDRAG_SATS_KODE*. The rates have changed.

STEP 3. Corrects ANSFRA, ANSTIL AND HELARKOD The step has *_01_raw-employment-3* as indata and generates *_01_raw-employment_4*.

In order to compute variables *JOB_START_DATO* and *JOB_SLUT_DATO* and *ANS_TYPE* in STEP 4 the ANSFRA, ANSTIL dates are checked.

Each record is identified by (*PNR*, *SENR*, *AAR*) and each of these will be classified as

1. *ANS_TYPE*= 1: full year employment.
2. *ANS_TYPE*= 2: not full year employment but continuous employment.
3. *ANS_TYPE*= 3: not full year employment but two or more employments.

In general, identifying the three different types of employment is easy as long as the employer follows the guidelines for reporting taxable income. Employment of type 1 is marked by variable HELARKOD = 1 and no values for ANSFRA and ANSTIL. Employment of type 2 is marked by filling out ANSFRA (employment start-date) and ANSTIL (employment end-date), and employment of type 3 has HELAARK=0 and ANSFRA and ANSTIL have no values.

A small fraction of records does not satisfy the guidelines. One example is HELARKOD = 1, ANSFRA = 0101 and ANSTIL =1231 which clearly is a full-year employment although ANSFRA and ANSTIL do not have missing values.

ANSFRA and ANSTIL have the format MMDD.

The variable *ERROR_DATES* identifies the changes made.

Finally, the dates are made valid by correcting DD to max value for a given month.

Together with variable AAR they are used to construct *JOB_START_DATO* and *JOB_SLUT_DATO*.

For the years 1998-2007 the dates are checked and updated using *MIAPNRM*, see the \raw_employment \job_02_correcting_dates_using_MIAPNRM_2021_v1. ANSFRA and ANSTIL are not changed but four new variables are created NUM_ANSFRADD, NUM_ANSFRAMM, NUM_ANSTILDD, and NUM_ANSTILMM.

The IF - THEN - ELSEIF .... structure used to check the ANSFRA, ANSTIL is:

* ANS_TYPE 1: Full employment *
IF

HELARKOD = 1

THEN

IF
at least one of ANSFRA, ANSTIL has a value
IF
ANSFRA has a missing value
ERROR_DATES = '1'
set NUM_ANSFRA to default value 0101.
* ANSTIL must have a value *
unpack ANSTIL
ELSEIF
ANSTIL has a missing value
ERROR_DATES = '2'
set NUM_ANSTIL to default value 1231
* ANSFRA must have a value *

unpack ANSFRA
ELSE
* Both ANSFRA and ANSTIL have a value *
unpack ANSFRA and ANSTIL

ELSE

* HELARKOD = 0 *

IF

both ANSFRA and ANSTIL have a missing value
ANS_TYPE = 3
set NUM_ANSFRA and NUM_ANSTIL to 0

ELSE

* One of or both ANSFRA and ANSTIL have a valid value *
ANS_TYPE = 2

IF

ANSFRA has a missing value
* ANSTIL must have a valid date *
ERROR_DATES= 3
set ANSFRA to default 0101.

ELSEIF

ANSTIL has a missing value
ANSFRA must have a valid date
ERROR_DATES = 4
set ANSTIL to default value 1231
unpack ANSFRA

ELSE

* both ANSFRA and ANSTIL have valid dates *
ERROR_DATES = 0
ANS_TYPE = 2
unpack ANSFRA and ANSTIL

STEP 4. ANSFRA and ANSTIL have been recoded to NUM_ANS. The new NUM_ANS variables are used to assign the most likely value to *ANS_TYPE*. The variables *JOB_START_DATO*, *JOB_SLUT_DATO* and *ANS_DAGE* are computed. The new NUM_ANS variables are checked for illegal values.

The IF - THEN .... structure used to compute *ANS_TYPE* :

IF

ANS_TYPE =1

```
THEN
  IF NOT (NUM_ANSFRA = 0101 AND NUM_ANSTIL = 1231)

  THEN
    *trust the dates, change ANS_TYPE *
    ERROR_DATES = 6
    ANS_TYPE = 2
    HELARKOD = 0

  ELSE IF ANS_TYPE= 2
    IF(NUM_ANSFRA = 0101 AND NUM_ANSTIL = 1231)
    THEN
      *trust the dates, change ANS_TYPE *
      ERROR_DATES = 5
      ANS_TYPE = 1
      HELARKOD = 0
  * type 2 check dates *
  IF NUM_ANSFRAMM = 0
  THEN
    ERROR_DATES = 7
    NUM_ANSFRAMM = 1
  IF NUM_ANSFRADD = 0
  THEN
    ERROR_DATES = 7
    NUM_ANSFRADD = 1
  IF NUM_ANSTILMM = 0
  THEN
    ERROR_DATES = 7
    NUM_ANSTILMM = 12
  IF NUM_ANSTILDD = 0
  THEN
    ERROR_DATES = 7
    NUM_ANSTILDD = 31
  IF 12 < NUM_ANSFRAMM < = 1
  AND 1 ≤ NUM_ANSFRADD ≤ 12
  THEN
    ERROR_DATES = 8
    switch DD and MM
    * similar for NUM_ANSTIL *
  * type 2 check dates *
  IF
    FRAMM is 00 set it to 01
    TILDD is 00 set it to 31
    TILMM is 00 set it to 12
    ERROR_DATES = 7
```

IF

MM is in range [13,31] and DD is in range [1,12] for either FRA or TIL dates the MM and DD are interchanged.

ERROR_DATES = 8

IF

MM $>$ 80 and DD is in range [1,12] for either FRA or TIL dates MM is assumed to be year and DD to be month then DD is set to 1 for FRA and DD is set to 31 for TIL.

ERROR_DATES = 9

IF

MM $>$ 12

for FRA then MM is set to 1 and DD to 0.

for TIL MM is set to 12 and DD to 31.

ERROR_DATES = 10

* after these changes based on MM, D may still have an 'illegal value' for *ANS_TYPE* = 2 *

IF *ANS_TYPE*= 2

THEN

IF DD is not in range [1,31]

for FRA DD is set to 1 and MM to 1

for TIL DD is set to 31 and MM to 12

ERROR_DATES = 12

IF MM is not in range [1,12]

for FRA MM is set to 1

for TIL MM is set to 12

ERROR_DATES = 12

IF FRA $>$ TIL

FRA and TIL are interchanged

ERROR_DATES = 13

The table 7 describes how the dates are changes.

Table 7 Code values for ERROR_DATES

Value	Label
1,2	HELARKOD=1, either ANSFRA or ANSTIL has a valid date. *ANS_TYPE* is set to 2. Missing values for NUM_ANSFRA are set to 0101 and *ERROR_DATES* = 1, a missing value for NUM_ANSTIL is set to 1231 and *ERROR_DATES* = 2.
3,4	HELARKOD=0, either ANSFRA or ANSTIL has a missing value. Missing value for NUM_ANSFRA is set to 0101, NUM_ANSTIL is set to 1231, and *ERROR_DATES* is set to 4.
5	IF HELARKOD=0 and *ANS_TYPE* = 2 but ANSFRA = 0101 and ANSTIL = 1231 then *ANS_TYPE* is set to 1
6	*ANS_TYPE* = 1, ANSFRA and ANSTIL are unedited and have date values different from 0101, 1231. *ANS_TYPE* is set to 2.
7	HELARKOD=0, *ANS_TYPE* = 2, ANSFRA and ANSTIL are not updated, but one has a missing value. If ANSFRA is missing it is set to 0101, and if ANSTIL is missing it is set to 1231.
8	If $1 \leq$ NUM_ANSFRAMM ≤ 31 and $1 \leq$ NUM_ANSFRADD ≤ 1 then DD and MM are switched. The same for NUM_ANSTIL.
9	The first year in data is 01-01-1988. So if MM ≥ 80 we assume it is the year. If NUM_ANSFRAMM ≥ 80 and $1 \leq$NUM_ANSFRADD ≤ 12 then NUM_ANSFRAMM is set to NUM_ANSFRADD, and NUM_ANSFRADD is set to 1. The same for ANSTIL. If DD 12 then NUM_FRA is set to 0101 and NUM_TIL is set to 1231
10	If $12 \leq$ MM ≤ 80 NUM_ANSFRA is set to 0101. The same for NUM_ANSTIL which is set to 1231.
11	If NUM_ANSFRADD ≥ 31, then NUM_ANSFRAMM is set to 1, and NUM_ANSFRADD = 1. If NUM_ANSTILDD ≥ 1, then NUM_ANSTILDD is set to 31, and NUM_ANSTILMM = 1.
12	NUM_ANS has an illegal value: NUM_ANSFRADD is set to 1 NUM_ANSFRAMM is set to 1 NUM_ANSTILDD is set to 31 NUM_ANSTILMM is set to 12 The same for NUM_ANSTIL.
13	If NUM_ANSFRA $\leq$ NUM_ANSTIL they are switched.

Table 8 shows the distribution of error dates.

After correction of obvious errors approximately 90 percent of the observations have no error.

All other error types other than 5 do not occure after 1989.

Error types 5,7,13 are not serious errors; 8-10 percent of the observations have these errors. These changes are well justified.

The error types 3,4,6,7,8,9,10,11,12,13 are not well justified changes. Observations with these changes are less than 2 percent.

For error types 1,2,3,4,7,10,11 one of the dates has a valid value, the other value is missing. The missing value is set to one of the arbitrary dates 01-01 , 31-12. For error types 3,4 these changes have no effect if the specified date is 01-01 or 31-12.

There are no observations with error type 6.

Observations with *ANS_TYPE* equal to 3 have no date values; they are set to the arbitrary dates 01-01 and 31-12. In the first years 12-14 percent of the observations have this *ANS_TYPE*; in the lates years the percent is 5-8.

In total about 85 % of the observations have correct start and end dates.

Table 8 errorcodes

YEAR	DATE ERROR CODES													
Frequency Row Pct	0	1	2	3	4	6	7	8	9	10	11	12	13	TOTAL
1984	3694145	1161	103	89	45	314886	170	179	13	34	44	8	2651	4013528
	92.04	0.03	0.00	0.00	0.00	7.85	0.00	0.00	0.00	0.00	0.00	0.00	0.07	
1985	3869525	126	44	43	76	327955	411	1727	9	25	166	4	7144	4207255
	91.97	0.00	0.00	0.00	0.00	7.79	0.01	0.04	0.00	0.00	0.00	0.00	0.17	
1986	4023597	110	0	34	81	352697	198	1968	14	149	17080	0	8452	4404380
	91.35	0.00	0.00	0.00	0.00	8.01	0.00	0.04	0.00	0.00	0.39	0.00	0.19	
1987	3977546	3	13	18	196	357164	184	93	16	10	14509	1	7398	4357151
	91.29	0.00	0.00	0.00	0.00	8.20	0.00	0.00	0.00	0.00	0.33	0.00	0.17	
1988	3928456	0	0	344	125	357839	391	155	17	8	588	2	9285	4297210
	91.42	0.00	0.00	0.01	0.00	8.33	0.01	0.00	0.00	0.00	0.01	0.00	0.22	
1989	3809333	0	7	433	299	400075	9	766	14	6	446	1	8692	4220081
	90.27	0.00	0.00	0.01	0.01	9.48	0.00	0.02	0.00	0.00	0.01	0.00	0.21	
1990	3765725	0	0	0	0	399165	0	0	0	0	0	0	717	4165607
	90.40	0.00	0.00	0.00	0.00	9.58	0.00	0.00	0.00	0.00	0.00	0.00	0.02	
1991	3686119	0	0	0	0	405427	0	0	0	0	0	0	7	4091553
	90.09	0.00	0.00	0.00	0.00	9.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1992	3515221	0	0	0	0	511091	0	0	0	0	0	0	0	4026312
	87.31	0.00	0.00	0.00	0.00	12.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1993	3468692	0	0	0	0	510526	0	0	0	0	0	0	0	3979218
	87.17	0.00	0.00	0.00	0.00	12.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1994	3629902	0	0	0	0	512178	0	0	0	0	0	0	0	4142080
	87.63	0.00	0.00	0.00	0.00	12.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1995	3686343	0	0	0	0	511874	0	0	0	0	0	0	0	4198217
	87.81	0.00	0.00	0.00	0.00	12.19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1996	3658224	0	0	0	0	525084	0	0	0	0	0	0	0	4183308
	87.45	0.00	0.00	0.00	0.00	12.55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1997	3669556	0	0	0	0	540547	0	0	0	0	0	0	0	4210103
	87.16	0.00	0.00	0.00	0.00	12.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1998	3782429	0	0	0	0	532278	0	0	0	0	0	0	0	4314707
	87.66	0.00	0.00	0.00	0.00	12.34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1999	3822383	0	0	0	0	537194	0	0	0	0	0	0	0	4359577
	87.68	0.00	0.00	0.00	0.00	12.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
2000	3900143	0	0	0	0	515056	0	0	0	0	0	0	0	4415199
	88.33	0.00	0.00	0.00	0.00	11.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
2001	3909186	0	0	0	0	520861	0	0	0	0	0	0	0	4430047
	88.24	0.00	0.00	0.00	0.00	11.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
2002	3804816	0	0	0	0	556369	0	0	0	0	0	0	0	4361185
	87.24	0.00	0.00	0.00	0.00	12.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
2003	3764331	0	0	464	240	549659	0	0	0	0	0	0	1092	4315786
	87.22	0.00	0.00	0.01	0.01	12.74	0.00	0.00	0.00	0.00	0.00	0.00	0.03	
2004	3680076	0	0	1442	234	582551	0	0	0	0	0	0	993	4265296
	86.28	0.00	0.00	0.03	0.01	13.66	0.00	0.00	0.00	0.00	0.00	0.00	0.02	
2005	3782794	0	0	80	1064	612029	0	1242	0	0	0	0	681	4397890
	86.01	0.00	0.00	0.00	0.02	13.92	0.00	0.03	0.00	0.00	0.00	0.00	0.02	
2006	3979721	0	0	0	0	628931	0	22	0	0	0	0	0	4608674
	86.35	0.00	0.00	0.00	0.00	13.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
2007	4111747	0	0	0	0	504357	0	33	0	0	0	0	0	4616137
	89.07	0.00	0.00	0.00	0.00	10.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Total	90,920,010	1,400	167	2,947	2,360	11,565,793	1,363	6,185	83	232	32,833	16	47,112	102,580,501

The table 9 shows the distribution of jobs on spell length. About 50 % of the jobs are a full year employment, about 40% of the jobs are one part year spell, and the remaining 10 % are 2 or more spells within the year. The start and end date of these jobs are unknown.

Table 9 anstype

YEAR	ANS_TYPE CODES			
Frequency Row Pct	one full year spell	one part year spell	2 or more spells	total
1984	1985679 49.47	1470988 36.65	556861 13.87	4013528
1985	1992686 47.36	1609790 38.26	604779 14.37	4207255
1986	2047434 46.49	1739244 39.49	617702 14.02	4404380
1987	2101041 48.22	1697301 38.95	558809 12.83	4357151
1988	2121325 49.37	1657034 38.56	518851 12.07	4297210
1989	2128976 50.45	1526738 36.18	564367 13.37	4220081
1990	2143629 51.46	1541579 37.01	480399 11.53	4165607
1991	2138469 52.27	1464786 35.80	488298 11.93	4091553
1992	2084858 51.78	1453456 36.10	487998 12.12	4026312
1993	2044124 51.37	1479604 37.18	455490 11.45	3979218
1994	2098087 50.65	1624939 39.23	419054 10.12	4142080
1995	2125312 50.62	1677378 39.95	395527 9.42	4198217
1996	2177970 52.06	1603154 38.32	402184 9.61	4183308
1997	2212354 52.55	1625656 38.61	372093 8.84	4210103
1998	2297033 53.24	1689328 39.15	328346 7.61	4314707
1999	2363380 54.21	1671992 38.35	324205 7.44	4359577
2000	2392021 54.18	1689776 38.27	333402 7.55	4415199
2001	2390779 53.97	1724362 38.92	314906 7.11	4430047
2002	2449189 56.16	1633994 37.47	278002 6.37	4361185
2003	2447731 56.72	1577600 36.55	290455 6.73	4315786
2004	2472223 57.96	1584803 37.16	208270 4.88	4265296
2005	2388404 54.31	1794317 40.80	215169 4.89	4397890
2006	2437276 52.88	1877687 40.74	293711 6.37	4608674
2007	2435420 52.76	1999645 43.32	181072 3.92	4616137
Total	53475400	39415151	9689950	102580501

The table 10 shows that more than 99% of the observations have a positive wage, for the years 1984-1989 0.04% of the observations have a negative value, for all years 0.2 % of the observations have a zero value.

Table 10 jobloen

YEAR	WAGE VALUE CODES			
Frequency Row Pct	zero wages	negative wages	positive wages	total
1984	8909	1592	4003027	4013528
	0.22	0.04	99.74	
1985	10582	1632	4195041	4207255
	0.25	0.04	99.71	
1986	20182	2226	4381972	4404380
	0.46	0.05	99.49	
1987	9843	2105	4345203	4357151
	0.23	0.05	99.73	
1988	8344	1685	4287181	4297210
	0.19	0.04	99.77	
1989	7416	1866	4210799	4220081
	0.18	0.04	99.78	
1990	4268	0	4161339	4165607
	0.10	0.00	99.90	
1991	5240	0	4086313	4091553
	0.13	0.00	99.87	
1992	4289	0	4022023	4026312
	0.11	0.00	99.89	
1993	4124	0	3975094	3979218
	0.10	0.00	99.90	
1994	17005	0	4125075	4142080
	0.41	0.00	99.59	
1995	12140	0	4186077	4198217
	0.29	0.00	99.71	
1996	4960	0	4178348	4183308
	0.12	0.00	99.88	
1997	6667	0	4203436	4210103
	0.16	0.00	99.84	
1998	9155	0	4305552	4314707
	0.21	0.00	99.79	
1999	9448	0	4350129	4359577
	0.22	0.00	99.78	
2000	8604	88	4406507	4415199
	0.19	0.00	99.80	
2001	10617	0	4419430	4430047
	0.24	0.00	99.76	
2002	12932	0	4348253	4361185
	0.30	0.00	99.70	
2003	11577	3	4304206	4315786
	0.27	0.00	99.73	
2004	12390	0	4252906	4265296
	0.29	0.00	99.71	
2005	10416	0	4387474	4397890
	0.24	0.00	99.76	
2006	94078	3	4514593	4608674
	2.04	0.00	97.96	
2007	13736	0	4602401	4616137
	0.30	0.00	99.70	
Total	316922	11200	102252379	102580501

2.1.3 02 joining ESR_CVRNR before 2008_2021. v.1.

If *ESR_SE_CVR* has been updated the jobs in subdirectory *spell\firm_id\programs\before2008*.

The new *CONESR* has *SENR* for the entire period. For the period 2000-2007 this *SENR* is used to match with *SPELL_FIRM_ID* and add *CVRNR*. From 2008 *SENR* and *CVRNR* are selected from *BFL*.

2.2 _02_correction_dates_using_MIAPNRM_2021_v1

. Corrections are made for 1998 - 2008. A MIAPNRM observation is identified by (AAR, MONTH, PNR, SENR, DSKOD.)

STEP 1. A temporary dataset *miapnrnr* is created where the MIA variable ANGPER is imputed in AAR and MONTH.

Then the temporary dataset *temp.miapnrm* is created using miapnrm_t. The programme will make a match between *raw_employ_cons_ras* and *temp.miapnrm_t* if the MIA observation exists in month t-1, t, t+1.

A temporary dataset *temp.miaprnm* is created using dataset *mia.prnm_t*. This dataset contains the original MIA observations for time t, and a copy of the observations for t-1 and t+1, where AAR and MONTH are set to t-1 and +1.

The original observations have ORG = 1, the added observations have ORG = 0.

STEP 2. A temporary dataset *cons_monthly_obs_2* is created using *rawbf08.raw_employment_consesr_aar*. Fr the period 1998-2007 the yearly observataions are converted to monthly observations. It contains the variables START_DATO, SLUT_DATO, MONTH_START, PNR, SENR, DSKOD.

STEP 3. The MIA data, *temp.miapnrnr*, and the monthly RAWEMPLOY data are merged.

For *ANS_TYPE* = 1,2 we can use the ORG = 0,1 observations for the match.

For *ANS_TYPE* = 3 it is only possible to use ORG = 1.

It creates the dataset *employ_consesr_monthly.*

For years 1998-2007 we have additional employment information from dataset MIAPNRM. The dataset contains information on a monthly level about whether an individual was working or not and at which firm the employment took place. We use this information to verify the start and end date collected from raw employment records, i.e. from CONS/RAS.

April 2007 is a month with many erroneous records in MIAPNRM. The stock of individuals drop from approximately 2.8 million to 1.8 million. We correct this erroneous records by 'adding' a fictitious record for April 2007 if the individual is present in March 2007 and May 2007.

We define a boolean variable 'bool_potential_mia_validation' for each of the now monthly raw employment records which is equal to 1 if a given raw employment record (pnr, establishment, firm, year, month) matches MIAPNRM on (pnr, firm, year) and 0 otherwise.

Looking at the subset where the boolean is equal to one we look for a match on (pnr, firm, year, month), i.e. we look for a month-specific match between the two datasets. If there is a monthly match, we say the monthly raw employment record is confirmed otherwise we say it is not confirmed.

For raw employment records of 'ans_type' 2 we allow the first and last month to be confirmed by either a match on month or a match on the next/previous month. We now keep records for which either, year ¡ 1998, 'bool_potential_mia_validation' = 0, 'bool_potential_mia_validation' = 1 and record is confirmed, or 2008 ¡ year. Note that by keeping monthly records for which 'bool_potential_mia_validation' = 0, the stock of individuals in a given year should not change as a result of using MIAPNRM information.

Further, the records for years 2008-2013 do not need to split into monthly records as they are already monthly due to the structure of BFL. We now split salary and estimated hours worked onto the monthly employment records for years prior to 2008. We split the two measures by the amount of days worked in a given month compared to total days worked for the yearly raw employment record. The resulting dataset of this programme is

_11_monthly_mia_confirm_obs. The PDF of the programme shows that 'bool_potential_mia_validation' = 1 for around 95% of raw employment records for years 1998-2007.

For these records around 80% of the monthly raw employment records are confirmed.

2.2.1 03 -estimating timetal ATP before 2008_2021.v1

The purpose of this programme is to estimate number of hours worked using the Lund-Vejlin algorithm.[1]

The resulting dataset of this programme is *05_raw_employment_spells*

programme name:

Purpose:

The purpose of this SAS- programme is to estimate average salary per hour worked for jobs done from 1980 - 2007. We know the total salary earned per job but to estimate average salary per hour worked we need to estimate how many hours the person in question have worked. We use information about paid ATP to estimate hours worked. The levels of full ATP payments varies and they are as follows;

- 1980 - 1981: Only one level of ATP which is 432 kr.
- 1982 - 1987: Still only one level of ATP which is 1166 kr.
- 1988 - 1989: Two levels of ATP, high-level is 1749 kr. and low-level is 1166 kr.
- 1990 - 1995: Still two levels of ATP, high-level is 2332 kr and low-level is still 1166 kr.

[1] see Documenting and improving the Hourly Wage Measure in the Danish IDA database, Nationaløkonomisk Tidsskrift, Issue 1 No 1, Vol. 2016

- 1996 - 2013: Now three levels of ATP, 'sats A' is 2682 kr., 'sats B' is 1166 kr. and 'sats C' is 1516 kr.

The ATP amount paid varies due to amount of hours worked per week. For the period 1980 - 1992 the rules were as follows;

- 0 - 9 hours worked: No ATP amount paid.
- 10 - 19 hours worked: 1/3 of full ATP amount.
- 20 - 29 hours worked: 2/3 of full ATP amount.
- 30 hours worked or more: Full ATP amount.

For the period 1993 and onwards the rules are as follows;

- 0 - 8 hours worked: No ATP amount paid.
- 9 - 17 hours worked: 1/3 of full ATP amount.
- 18 - 26 hours worked: 2/3 of full ATP amount.
- 27 hours worked or more: Full ATP amount.

We calculate variable 'ATP.broek' as the relation between ATP payments and ATP level:

$$ATP.broek = \frac{ATP.payments}{ATP.sats}$$

We now estimate minimum and maximum hours worked that will result in the observed 'ATP.broek'. We first look at ATP.broek = 0. We do (example where year is 1993);

$$ATP.broek = \leq \frac{1}{3}$$

$$minimum.hours.worked = \left(52 weeks * \frac{ATP.broek}{\frac{1}{3}} \right) * 9.hours$$

$$maximum.hours.worked = \left(52 weeks * \frac{ATP.broek}{\frac{2}{3}} \right) * 17.hours$$

We now look at

$$ATP.broek = \leq \frac{2}{3}$$

$$minimum.hours.worked = \left(52 weeks * \frac{ATP.broek}{\frac{2}{3}} \right) * 18.hours$$

$$maximum.hours.worked = \left(52weeks * \frac{ATP.broek}{\frac{2}{3}}\right) * 26.hours$$

Finally, we look at

$$ATP.broek = \leq 1$$

$$minimum.hours.worked = (52weeks * ATP.broek) * 27.hours$$

$$maximum.hours.worked = (52weeks * ATP.broek) * 37.hours$$

2.2.2 04 joining idas_lbnr before 2008_2021. v.1

The purpose of this programme is to merge the establishment identifier from *IDAS* (*LBNR*) onto the resulting dataset of programme 02 in order to define spell establishment identifier. We use dataset *IDAS* to merge *LBNR* onto the raw employment records. *IDAS* contains information on non-fictive establishments and this dataset is available for the entire spell period, but the *IDAS* dataset is available much later than *BFL*. We merge on (*ARBGNR*, *ARBSTK*, *YEAR*).

The resulting dataset of this programme is *_03_raw_employment_spells.*

The PDF file of the programme analyzing this dataset shows that we are able to map *LBNR* from IDAS for 90% - 95% for 1985-2006 and for 2008-2013. For year 2007 the level of matches drops quite substantially. [2] The resulting dataset of this programme is *_06_raw_employment_spells.* This is the result of the process to create the *raw_employment_spells_dataset*

2.2.3 05 joining fida_cvr before 2008_2021. v1

This programme has to be updated when FSE delivers a correct FIDA dataset.

The purpose of this programme is to merge the firm identifier (*CVRNR*) used in *FIRM* onto the resulting dataset of programme 01 in order to use this variable as our *SPELL_FIRM_ID* whenever possible. Usage of this firm identifier enables us to merge employment spells directly with the datasets of the firm side *FIRE*, f.ex.

We merge the resulting dataset of programme *fida_analysis_v1 named fida_3* onto the resulting dataset of the previous programme 01. Dataset *fida_3* contains the year-specific mapping between the establishment identifiers of CONS/RAS/ and the firm identifier of *FIRM* constructed using *FIDA*. Analysis of *FIDA* has shown that the mapping is independent of *PNR*. i.e. the mapping is n-1 (if the mapping depended

[2] Note that we do not expect a 100% match since *IDAS* only contains information on non-fictive establishments. For raw employment records with no match, we set *LBNR* equal to '0000000000' consistent with DST's coding of fictive establishments.

on *PNR* then we would expect the mapping to be n-m), [3] Only 5% ans_type = 3 when looking at non-erroneous raw employment records. The low fraction of no-error is likely due to the fact that all blank raw employment records are assigned to ANS_TYPE = 3.

Given that the mapping is independent of *PNR* we merge on the condition that year and establishment identifiers match between the two datasets. This is important as it enables us to use the firm identifier of *FIRM* for individuals who are not present in *FIDA*. The first year of *FIDA* is 1995, however we are only able to use *FIDA* for years 2003-2007 since the establishment identifiers of *FIDA* differ from the establishment identifiers of CONS for years prior to 2003.

The resulting dataset of this programme is *_02_raw_employment_spells.*

The PDF of the programme shows that we are able to match 90% - 97% of raw employment records with FIDA.

2.2.4 old-04-joining_mapped_cvrnr_before_2008_2021_v4

The purpose of this programme is to merge firm identifier 'mapped_cvrnr' onto the resulting dataset of the previous programme in order to define spell firm identifier for records for which we are not able to merge information on the firm identifier from in *FIRM*, i.e. records where no merge was possible in programme 02.

The merging of the firm identifier 'mapped_cvrnr' is conducted using dataset *_cvr_se2014_5* constructed in programme *se_cvrnr_92* (located in folder arbsted). The firm identifier 'mapped_cvrnr' is a variable provided by DST that maps a given 'senr' into a 'cvrnr'. There exist few records in the mapping dataset for which a given value of 'senr' maps to some 'cvrnr' and then in later years map to another 'cvrnr'. In this case, we use the most recent mapping in all years. We now have all the needed firm identifiers merged onto our raw employment records needed in order to define the spell firm identifier called 'spell_firm_id'.

From 2008 *CVRNR* of *BFL* is used as spell firm identifier.

The resulting dataset of this programme is *_04_raw_employment_spells.*

The PDF of the programme shows the level and percentage of raw employment records for which we are able to match a mapped_cvrnr. The level of matches increase by year as we get closer and closer to the launch of firm identifier 'cvrnr'.

2.3 FROM 2008

The *BFL* is used as input to the jobs. The following jobs are used to create the employment spells.

[3] http://www.dst.dk/da/Statistik/dokumentation/Times/moduldata-for-arbejdsmarked/atpblb.

2.3.1 00_driver from 2008_2021_v1

The driver runs the programmes needed for generating the dataset *raw_employment*. The driver job can be edited using the following % let values:

- *INOBS*, controls how many records are read: max or an integer number.
- *BOOL_DELETE*, deletes temporary files after each step.
- *BOOL_PDF_ON*, controls generation of PDF files for description and debug.
- lib_in_SPELL, SAS library for in data.
- LIB_OUT, library for storing permanent output data. Test and temporary data are stored in
  temp. The files must be deleted manually when they are not needed anymore,
- START_YEAR and END_YEAR determine the period for whih the spells are generated

2.3.2 01 extracting bfl from 2008_2021. v1

The relevant variables are extracted from *BFL*. They contain a few more variables than raweb08 raw_employment_conesr_ras.

If the value of *JOB_LOEN_BELOEB_SMAL* is missing then it is set to 0. If the variable *JOB_LOEN_BELOEB_SMAL* is 0 the BOOL_LONBLB is set to 1, else to 0.

A BOOL variable is set for missing values for *JOB_ATP_BELOEB* and *JOB_ATP_BIDRAG_SATS_KODE*. All ATP contributions are converted to SATSB and assigned to variable *JOB_ATP_BIDRAG_SATS_KODE* using *JOB_ATP_BELOEB*.

The files raweb08 raw_employment_conesr_ras and rawef08 raw_employment_bls are merged to rawempl raw_employment. The resulting dataset is checked fr duplicaes using key (aar, pnr). Duplicate records are written to temp_employment in DDP. The created dataset is described in several PDF files in directory ?????

- Number of missing values for SENR
- Number of missing vales for *JOB_ATP_BIDRAG_SATS_KODE*
- Number of missing values for *JOB_ATP_BELOEB*
- Showing frequencies for *ANS_TYPE*
- Histogram of AGE

2.3.3 02 joining ESR_CVRNR from 2008_2021. v1

2.3.4 04_joining_idas_lbnr_arbnr_from_2008_2021_v1

The purpose of this programme is to merge the firm identifier (*CVRNR*) used in *FIRM* onto the resulting dataset of programme 01 in order to use this variable as our *SPELL_FIRM_ID* whenever possible. Usage of this firm identifier enables us to merge employment spells directly with the datasets of the firm side *FIRE*, f.ex.

2.3.5 03_joining_IDAS_LBNR_from_2008_2021_v1

The purpose of this programme is to merge the establishment identifier from *IDAS* (*LBNR*) onto the resulting dataset of programme 02 in order to define spell establishment identifier. We use dataset *IDAS* to merge *LBNR* onto the raw employment records. *IDAS* contains information on non-fictive establishments and this dataset is available for the entire spell period, but the *IDAS* dataset is available much later than *BFL*. We merge on (*CVRNR*, *ARBNR*).

(*ARBNR*, *YEAR*) should uniquely identify an establishment but for a small amount of *IDAS* records (*ARBNR*, *YEAR*) are not unique on (*CVRNR*, *ARBNR*, *YEAR*). It has been checked that for a given match on (*ARBNR*, *YEAR*) we also have match on (*CVRNR*, *ARBNR*, *YEAR*). The problem has not yet been solved buy you could match on (*CVRNR*, *ARBNR*, *YEAR*).

The resulting dataset of this programme is *_03_raw_employment_spells.*

The PDF file of the programme analyzing this dataset shows that we are able to map *LBNR* from IDAS for 90% - 95% for 1985-2006 and for 2008-2013.

2.3.6 old-05-estimating_timetal_ATP_from_2008_2021_v1

For now this programme is not used. Later on it may be used to be compared with variables in *BFL*.

The purpose of this programme is to estimate number of hours worked using the Lund-Vejlin algorithm.[4] In order to have a consistent hours worked measure, we estimate hours worked for all hours worked for years from 2008. An analysis shows that the observed amount of hours worked measure in BFL is not without problems. We keep both the estimated and the observed amount of hours worked. We directly use the SAS code of Lund-Vejlin. The estimation of hours is computed for each (pnr, arbnr, year). Raw employment records are monthly thus in order to use the Lund-Vejlin algorithm we aggregate the monthly records to yearly records for each establishment (pnr, arbnr, year). It is only a temporary aggregation, i.e. we do not discard the monthly information for records in 2008 and later. After having estimated yearly hours worked for (pnr, arbnr, year), we linearly assign total hours to each monthly record. Note that in order for this SAS- programme to run it is needed that SAS- programmes 01merge- Pooled, 02BasicQuantitiesChap4Pooled, and 03CalcProbsPooled have all been run. These programmes are located in *Raw employ obs TIMELON.*

The resulting dataset of this programme is *_05_raw_employment_spells.*

[4] see Documenting and improving the Hourly Wage Measure in the Danish IDA database, Nationaløkonomisk Tidsskrift, Issue 1 No 1, Vol. 2016

2.3.7 06-final_preparation_of_raw_employment_spells_after_2008_2021_v1

The purpose of this programme is to make final adjustments to the resulting dataset of the previous programme. The following adjustments are made,

1. The spell establishment identifier is defined as 'lbnr' obtained from *IDAS* in programme 03. The variable is named SPELL_LBNR.
2. We conduct a minimum of sample selection. We discard records if the age of the individual is below age 15, and/or if 'wage' is equal to zero, and/or if spell firm identifier is missing.
3. We discard temporary variables used in computation throughout all the previous programmes.
4. For all non-discarded raw employment records SPELL_RAW_EMPLOY_ID is added. In the computation of employment spells using these raw employment records a unique sequential identifier keeps track of which raw employment records have been used for each final employment spell. In this way, it is possible to 'go back one level' and e.g. see establishment level changes within an employment spell.

The resulting dataset of this programme is *_06_raw_employment_spell*. This is the result of the process to create the *raw_employment_spells_dataset*

We keep a dataset containing a larger list of variables mainly for debug purposes. This dataset is named *_06_raw_employment_spells_alvar.*

2.4 Checking_raw_employment_spells_v5

The jobs are in drectory *spell\programs \raw_employment textbackslash check*. This programme carries out checks on the constructed raw employment records including a count of the employment stock and an analysis of estimated timelon (hourly wage). We first check for missing values. We find that we do not have missing values in the raw employment records. Next, we compute the stock of workers in raw employment records at every second Monday of all month all years. We also compute the stock of workers on a yearly level.

The drop in the monthly stock of workers between December 2007 and January 2008 is not present when we look at the yearly stock of workers between 2007 and 2008, thus the drop in the stock of workers is mainly explained by the better start and end dates in BFL. I.e. for raw employment records without information on start and end date employment is set to have taken place between January first and December 31. also some employees probably report full year employment even when employment only took place for a sub period of the year. 11p. 1-14 of the PDF. 12p. 19-31 of the PDF.

We conduct a series of distributional analysis of the hours worked and hourly wage measures in the raw employment records. When conducting this analysis, we aggregate 2008 and later records to (pnr, arbnr, year). Recall that this is the level we estimate hours for in programme 05. We compare 'timelon' in IDAN for type H employments with a measure denoted FATH (full algorithm type H) and with estimated hourly wage in spells. The comparison between 'timelon' and FATH timelon shows the difference between DST and Lund-Vejlin hourly wage measure, especially for the early dates of the spell. The Lund-Vejlin hourly wage measure improves DST 'timelon' thus a difference between these two measures is expected. For spells, we are not able to use the full Lund- Vejlin algorithm since this algorithm uses IDA information which is only available for type H employment. The hourly spell wage measure is computed using a restricted version of FATH where the information only available to type H employment is left out. We compare FATH and spell hourly wage measure in order to quantify the loss in performance of using the restricted algorithm. Pages 276-302 of the PDF show the distribution of the difference between 'timelon' and FATH and the difference between FATH and hourly wage measure of spells. We see that the difference between 'timelon' and FATH is larger than the difference between FATH and hourly wage measure of spells for all years, thus the difference between 'timelon' and hourly wage measure in spells is mainly driven by the Lund-Vejlin improvements to the hourly wage measure.

2.4.1 old-04-joining_mapped_cvrnr_from_2008_2021_v4

The merging of the firm identifier 'mapped_cvrnr' is conducted using dataset *_cvr_se2014_5* constructed in programme *se_cvrnr_92* (located in folder arbsted). The firm identifier 'mapped_cvrnr' is a variable provided by DST that maps a given 'senr' into a 'cvrnr'. There exist few records in the mapping dataset for which a given value of 'senr' maps to some 'cvrnr' and then in later years map to another 'cvrnr'. In this case, we use the most recent mapping in all years. We now have all the needed firm identifiers merged onto our raw employment records needed in order to define the spell firm identifier called 'spell_firm_id'.

From 2008 *CVRNR* of *BFL* is used as spell firm identifier. The spell_firm_id is defined as the first non- missing value of the following variables (in decreasing priority): the firm identifier from FIDA, 'mapped_cvrnr', and 'senr'/'cvrnr'.

The resulting dataset of this programme is *_04_raw_employment_spells.*

The PDF of the programme shows the level and percentage of raw employment records for which we are able to match a mapped_cvrnr. The level of matches increase by year as we get closer and closer to the launch of firm identifier 'cvrnr'.

Chapter 3
Firm Identifiers

3 Introduction

The firm and workplace identifiers to the spell data are created by a sequence of jobs in directory *firm_id*.

The two major firm identifiers are *SENR, A firm account number for tax and custom payments* and *CVRNR, Encrypted legal unit identifier. Data break 1994*.

SENR was introduced in 1985 as an identifier for fiscal payments from firms to the customs and tax authorities.

CVRNR was introduced in 1999 as a unique identifier of legal units. Firms may also choose to have several *CVRNR* and to change a *CVRNR*. Therefore, *CVRNR* is not a unique identifier of an *economic unit*. *OKNR* is an idenfitier of an economic unit created by *ESR*. One *OKNR* may be owner of one or more *CVRNR*. One *CVRNR* is marked as carrying data about an economic unit.

A *CVRNR* may own one or more *SENR*. If the legal unit only has one *SENR* then the *CVRNR* is equal to the *SENR*.

The DS-DRDS project "Firm Warehouse" is making an *OK_OT_ID*, an over-time identifier for the economic unit from 2000.

The spell project has constructed a similar over-time identifier using worker-flows among *IDAS* workplaces. These identifiers can be constructed from 1980. They are compared to *OK_OT_ID*.

Until 2018, several datasets did not include *SENR* before 1999, but only *ARBGNR* or *JURNR*.

JURNR is an *ESR* unique identifier of legal units and *ARBGNR* is an IDA unique identifier of owners of workplaces. *JURNR* and *ARBGNR* are similar. For privately owned legal units they are equal to *SENR*. For non-private units different identifiers are used, see the description in *ARBGNR*. For privately owned legal units *ARBGNR* is a *SENR*, for others it is similar to *JURNR*.

From 2018 all relevant *FSE* datasets contain *SENR* hence *ARBGNR* and *JURNR* are no longer needed. Before 2018 *ARBGNR* was equal to *ARBGNR9* i.e. *ARBGNR8* with a one digit prefix identifying the system reporting taxable income. From 2018

ARBGNR is equal to *ARBGNR8* in FSE datasets, but all *ECONAU* datasets may not yet have been updated. If the dataset has been updated *ARBGNR* is renamed to *SENR*.

The *SENR* can be used to map *CVRNR* on these datasets. The mapping is done using dataset *ESR_SE_CVR*. *ESR_SE_CVR* is also used to verify *SENR* and *CVRNR* in all new data. All *raw spell dataset* use this mapped *CVRNR*.

3.1 Workplace identifier

Until 2008 a physical workplace is identified by (*AAR*, *ARBGNR*, *ARB_STK*) in *CONESR* and *RAS* or (*AAR*, *ARBGNR*, *DSKOD*) in IDA. *ARB_STK* and *DSKOD* are similaar but not identical. A fictitious workplace is identified by specific *ARB_STK* code values.

From 2008 a workplace is identified by *ARB_NR* and *ARB_SUBTYPE* and an indicator *FIKTIV_ARB_KODE* defines the workplace to be a physical (workplace) or a fictitious workplace.

IDA has constructed an over-time physial workplace identifier *LBNR*. *LBNR* is 00.00.00.00 for fictitious workplaces, hence they are not uniquely identified by *LBNR*, but must be identified by *ARBGNR*. If a *ARBGNR* has only one workplace then *ARB_STK* has a missing value, but (*AAR*, *ARBGNR*, *ARB_STK*) is still a unique key. An establishment is a physical workplace.

IDAS only has physical workplaces but *CONESR*, *BFL*, *IDAN*, and *RAS* have both physical and fictitious workplaces. *CONESR* and *BFL* only have jobs occupied by employed workers.

All the workplace keys are collected in spell dataset SPELL_ARB.

We keep a dataset containing a larger list of variables mainly for debug purposes. This dataset is named *_06_raw_employment_spells_alvar*.

4 Spell_firm_id

Dataset SPELL_FIRM_ID The job _01_esr_firm_id_b08_2021_v1 creates the dataset *SPELL_FIRM_ID* in libray *firm_id*. The dataset contains all firm and workplace identifers used in spell data. The job inputs *ESR_SE_CVR* and these observations are the core of *SENR* and *CVRNR*. The *SOURCE* and *SOURCE_CVR* are set to 1, with fomat label ”ESR”.

4.1 Match a CVRNR on dataset *RAW_EMPLOYMENT*

The wage reporting unit in *RAW_EMPLOYMENT_CONESR_RAS* has key *SENR*. The identifier *CVRNR* is mapped on using *ESR_SE_CVR*.

First it is atttempted to match on (*AAR*,vnJOBulSENR). If there is not a match it is attempted to match on (*JOB_SENR*) only. If there is a match *EDITS*=91 and spell dates are set to [1984-2007]. In both cases the observations are flagged, *SOURCE*=1 and *SOURCE_CVR*=1.

If there still is not a match it is attempted to match to *ESR_SE_CVR_EVENTS* using (*JOB_SENR*). These *ESR* observations do not have a *CVRNR*, therefore *JOB_CVRNR*is assigned *JOB_SENR*. *EDITS*=92 and spell dates are set to [1984-2007]. The observations are flagged, *SOURCE*=1 and *SOURCE_CVR*=2.

If there still is not a match these rerporting units in *RAW_EMPLOYMENT_CONESR_RAS* are added to*SPELL_FIRM_ID*. These observations do not have a *CVRNR*, therefore *JOB_CVRNR*is assigned *JOB_SENR*. EDITS=93 and spell dates are set to [1984-2007]. The observations are flagged, *SOURCE*=2 and *SOURCE_CVR*=2.

These tasks are done by the following jobs:

_02_match_esr_conesr_b08_2021_v1 For the period 1984-2007 *CVRNR* is matched on *RAW_EMPLOYMENT_CONESR_RAS* from *ESR_SE_CVR* using key (*AAR*, *SENR*.)

The key is unique for *ESR_SE_CVR*.

In-data for the marge are:

- *RAW_EMPLOYMENT_CONESR_RAS* has 102,580,501 obsevations.
- *ESR_SE_CVR* has 9,865,379 observations. The resulting datasets are stored in subdirectory extittemp.
- *RAW_EMPLOYMENT_CONESR_RAS* has 87,443,453 observations.
- *RAW_EMP_IN_EMP_NOT_ESR* has 14,137,048 observations without a match.

_01_match_esr_conesr_only_senr_b08_2021_v1 The purpose of this job is to examine how to match on a *CVRNR* on the observations in *RAW_EMP_IN_EMP_NOT_ESR*.

The dataset *SPELL_FIRM_ID* is a yearly data set. If a new *SENR* has to be added or the Rel spell dates are changes it is necessary first to create a spell_firm_id_events dataset and then expand it to yearly observations to be added to *SPELL_FIRM_ID*. If the observations have to be updated the old observations must first be deleted.

_02_match_esr_conesr_only_senr_b08_2021_v1 The missing match in previous jobs can be caused by

- the *SENR* is in *ESR_SE_CVR* but the spell dates are wrong.
- the *SENR* exists in *ESR_SE_CVR_EVENTS* but *CVRNR* is missing.
- the key vnSENR does not exist in *ESR*, The unit from *RAW_EMP_IN_EMP_NOT_ESR* is added to *SPELL_FIRM_ID*.
- *ESR_SE_CVR_EVENTS* contains *JOB_SENR* units with or without *CVRNR*.

STEP 1 Match *RAW_EMP_IN_EMP_NOT_ESR* and *ESR_SE_CVR_EVENTS* unsig only *JOB_SENR*.

The key (*SENR*) in *RAW_EMP_IN_EMP_NOT_ESR* is made unique. The resulting dataset has 241,424 observations.

The in datasets in the merged subdirectory *temp* are:

- *RAW_EMP_IN_EMP_NOT_ESR* has 241,424 obsevations.
- *ESR_SE_CVR_EVENTS* has 2,268,657 observations.

The out datasets are:

- *RAW_EMP_IN_EMP_NOT_ESR* has 122,076 observations with a match, some with, some without a *CVRNR*.
- *RAW_EMP_IN_EMP_NOT_ESR* has 119,348 observations in *RAW_EMP_IN_EMP_NOT_ESR_2* without a match.

two resulting dataset is 249,483 compared to 241,424 observations in indata.

The dataset *RAW_EMP_IN_EMP_NOT_ESR_2* has to be added to the dataset *SPELL_FIRM_ID* by job _02_change_rel_dates_conesr. As a first step it is added to *SPELL_FIRM_ID_EVENTS*.

It is checked that *RAW_EMP_IN_EMP_NOT_ESR_1* has no duplicaes on key (*SENR*.) The resulting dataset is *RAW_EMP_IN_EMP_NOT_ESR_5*. It has to be appended to *SPELL_FIRM_ID* by job *_02_change_rel_dates_conesr_b08_2022_v1*.

_1_02_change_rel_dates_conesr_b08_2021_v1 The job updates *SPELL_FIRM_ID* such that all observations in *RAW_EMPLOYMENT_CONESR_RAS* can be matched on *JOB_SENR*.

First the dataset *temp.SPELL_FIRM_ID_EVENTS* is created. It consists of *RAW_EMP_IN_EMP_NOT_ESR_2* with 119,348 obsevations and

- *SOURCE* = 2
- *SOURCE_CVR* = 2
- *EDITS* = 92
- *REL_START_AAR* = 1984
- *REL_SLUT_AAR* = 2007

and of *RAW_EMP_IN_EMP_NOT_ESR_5* with 238,696 obsevations and

- *SOURCE* = 1
- *SOURCE_CVR* = 1 if *CVRNR* is not missing, else *JOB_CVRNR* = *JOB_SENR* and *SOURCE_CVR* = 2
- *EDITS* = 91
- *REL_START_AAR* = 1984
- *REL_SLUT_AAR* = 2007

The two datasets are appended to *temp.SPELL_FIRM_ID_EVENTS*.

The dataset is merged with *SPELL_FIRM_ID* to delete all observ¡tions with key (*JOB_SENR*) to be replaced.

No observations were deleted!

The *temp.SPELL_FIRM_ID_EVENTS* dataset is expanded to early data which is appended to *SPELL_FIRM_ID* to form textitSPELL_FIRM_ID_1.

The job *_02_match_after_esr_conesr_b08_2022_v1* tests that all observations in *RAW_EMPLOYMENT* can be matched with *temp.SPELL_FIRM_ID_EVENTS*.

temp.SPELL_FIRM_ID_EVENTS is renamed to *SPELL_FIRM_ID*. It has 27,797,658 observations.

The job *freq_edits* in subdirectory *analysis* *before_2008* produces some descriptive pdf files.

4.2 OK units and workplace variables

The dataset *ESR_OK* contains the core variables of the *economic units*.

The dataset *ESR_OK_CVR* contains a yearly dataset of relations between *economic units* and *legal units*.

_03_esr_OK_CVR_b08_2021_v1 . This job matches *OKNR* and *CVR_DATABAERENDE* on *SPELL_FIRM_ID* using key (*AAR*, *JOB_CVRNR*) for the period [2004,2007]. *ESR_OK_CVR* has 5 duplicates on key (*AAR*, *JOB_CVRNR*).

- *SPELL_FIRM_ID* has 2,620,169 observations
- *ESR_OK_CVR* has 2,340,989 observations
- 2,452,912 observations have a match
- 107,256 observations in dataset *temp.IN_FIRM_NOT_ESR* were in *firm* not in *esr*
- 19,900 observations in dataset *test.NON_FIRM_IN_ESR* were in *esr* but not in *firm*

Chapter 4
Spell dataset SPELL_E

_00_driver_employment xxxx

_02_obs_before_2008_2021_v1 The purpose of this SAS programme is to split yearly raw employment records for the period 1985-2007 into monthly raw employment records.

Raw employment records for years prior to 2008 are reported at (pnr, establishment, firm, year) level.

We now split each of the yearly records into monthly records such that all records are reported as (pnr, establishment, firm, year, month). Within these prior to 2008 raw employment records we distinguish between 1985-1997 and 1998-2007 records.

12-Compress_overlaying_firm_id_obs_v4 The purpose of this programme is to identify monthly raw employment records for a given individual on the same 'spell_firm_id' in the same month which either overlap or have no day gap between end date and start date. Once identified, these multiple raw employment records are compressed into one with salary and amount of hours worked equal to the aggregated salary and amount of hours worked for the two raw employment records.

Having compressed overlaying or directly connected raw employment records on same 'spell_firm_id', we create a new spell identifier named 'spell_id_12' for all monthly raw employment records. The new spell identifier is unique on 'pnr' and 'year'. We create a mapping file between 'spell_firm_12' and the raw employment records identified by 'spell__employ_id'. We create a boolean variable called 'bool_overlap_or_zero' indicating whether the monthly raw employment record orig- inates from two or more compressed or directly connected raw employment records. Using the map ping file, it is possible to know which establishment(s) a given monthly raw employment record maps to.

The resulting datasets of this programme are named

_12_overlap_obs, _12_mapping_file The PDF of the programme shows that around 1% of the final monthly raw employment records of the programme originate from compressed monthly records.

The maximum amount of monthly raw employment records compressed into one is seventeen. Given that the final monthly employment record originates from compressed monthly records, 95% of the compressed monthly records have different values of 'lbnr'.

Finally, we count the number of initial monthly raw employment records going into this programme and the number of monthly raw employment records resulting of this programme.

14-Determine_overlay_status_v2 The purpose of this programme is to identify months for a given individual for which he/she is employed at two or more different firms. This information is used when we create primary employment spells in the following two SAS programmes.

The resulting dataset of this programme is named

_14_firm_id_pot_overlay_pnr.

15-Create_primary_secondary_employment_pot_overlay_v2 The purpose of this programme is to identify each monthly raw employment record as either primary or secondary employment. For a given combination of (pnr, month, 'spell_firm_id'), we compute the amount of hours worked and salary earned at the given firm in the current and two following months and name these aggregated hours worked and salary earned 'wide_lontimer', 'wide_lonblb' respec- tively.

For each month, we define raw employment records at the 'spell_firm_id' with the largest 'wide_lontimer' as primary employment. If two or more values of 'spell_firm_id' have the same amount of 'wide_lontimer' then we use 'wide_lonblb' to define primary raw employment records. We now look for raw employment records not chosen to be primary employment in a given month for which 'spell_firm_id' matches the primary employment value of 'spell_firm_id' of the previous month. If such raw employment records fit in a gap before the start date of the primary raw employment record of the month then we mark these raw employment records as primary employment. We mark raw employment records chosen to be primary employment in this second step by a boolean variable called 'bool_beg_obs'.

An example is an individual working at 'spell_firm_id' A from 1st of January until 10th of March and at 'spell_firm_id' B from 11th of March to 31st of August. Raw employment records at 'spell_firm_id' A is primary employment in January and February and raw employment records at 'spell_firm_id' B is primary employment in March-August. The raw employment record at 'spell_firm_id' A from 1st March to 10th of March is not chosen to be primary employment in the first place, but since it matches the previous month' primary employment 'spell_firm_id' and fits before the start date of March' primary 'spell_firm_id' the raw employment record is chosen to be primary employment.

If a primary raw employment record with 'bool_beg_obs' = 1 has end date greater or equal to the start date of the primary raw employment record with 'spell_firm_id' equal to the 'spell_firm_id' with largest 'wide_lontimer' then we do not alter the end date and hence the two primary raw employment records will overlap. We leave it to the researcher to choose whether to rely on either the end date information of the 'bool_beg_obs' = 1 record or the start date information in order to deal with the overlap. All raw employment records not chosen to be primary employment are defined as secondary employment. The resulting datasets of this programme are named

_15_prim_spell_firm_id, _15_secondary_jobs_all. The variables in *SPELL_E* are listed in Table 11

Table 11 List of Variables - SPELL_E

Variable name	Title
PNR	*PNR, Person identifier, encrypted CPRNR number*
SPELL_FIRM_ID	*SPELL_FIRM_ID, SPELL, Firm ID*
JOB_START_DATO	*JOB_START_DATO, In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*
JOB_SLUT_DATO	*JOB_SLUT_DATO, In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*
DSKOD	*DSKOD, Establishment code*
ARBGNR	*ARBGNR, IDA identifer for unit reporting wages for taxation, without sector prefix-IDA identifier unit reporting wages for taxation, without sector prefix.*

16-Compress_primary_and_secondary_jobs_v4 The purpose of this programme is to compress the monthly raw employment records into employment spells. We set salary and hours worked equal to the aggregated salary and hours worked across the monthly raw employment records constituting each employment spell. First, we compress raw employment records for individuals found in programme 14 not to have any month with overlapping raw employment records into primary employment spells. These spells are of course defined as primary employment spells since the individual does not have overlapping employment. Secondly, we compress the primary raw employment records found in programme 15 into primary employment spells. Last, we compress the secondary raw employment records into secondary employment spells.

All employment spells are indexed by 'spell_id_16' and we create a mapping file between 'spell_id_16' and 'spell_id_12', thus it is possible to disentangle a given employment spell into the monthly raw employment records constituting the given spell. Further, using the mapping file of programme 12, it is possible to map this 'spell_id_12' into the raw employment records to get information about e.g. establishment. We create subspells containing yearly and/or monthly salary and hours worked for each primary employment spell. For years prior to 2008, we report yearly salary and yearly estimated hours worked. For years 2008-2013, we report monthly salary and monthly observed hours worked. The resulting datasets of this programme are named

- spells_prim_employ (_16_prim_jobs_compress)
- mapping_file_prim (_16_mapping_file_prim)
- subspells_sec_employ (_16_secondary_jobs_compress)
- mapping_file_sec (_16_mapping_file_sec)

- subspells_lon (_16_subspell_lon).

The dataset spells_prim_employ contains variables

- ’pnr’, person identifier.
- ’lonblb’, salary earned.
- ’startdato’, start date of employment spell.
- ’slutdato’, end date of employment spell.
- ’lontimer_estimated’, hours worked estimated in programme 05.
- ’lontimer_mix’ equal to lontimer_estimated for years prior to 2008 plus lontimer_bfl for years 2008-2013.
- ’pers_foed_dag’, date of birth from dataset PERSONER.
- ’spell_firm_id’, spell firm identifier defined in programme 04.
- bool_multiple_lbnr, indicating whether the employment spell consists of employment at two or more different ’spell_lbnr’.
- bool_overlap_or_zero, indicating whether the employment spell consists of raw employ- ment records collapsed in programme 12.
- bool_beg_obs, boolean defined in programme 15.
- ’spell_id_16’, identifier of an employment spell, unique by pnr.

22-Checking_constructed_employment_spells_v4 The purpose of this SAS programme is to check the constructed employment spells. P. 2 of the PDF shows the monthly count of workers in the primary employment spell data.

P. 3 shows the monthly count of firms (spell_firm_id).

P. 4 shows a primitive count of job-to-job transitions defined as an E-N-E transition from one firm to another with a maximum of 40 days between the two employment spells. The spike in 2007 is believe to be due to the big municipality reform of 2007. Further, note there is a spike in 2003, this is believed to be due to the fact that we from 2003 use ’cvrnr’ of FIDA as spell_firm_id.11

P.5 shows a primitive count of recalls, where a recall is defined as E-N-E again with a maxi- mum of 40 days between the two employment spells involving separation and hiring from/to the same spell_firm_id.

P.6 shows a count of recalls where we condition on the recall happening within the same year, i.e. the end date and start date need share the same year. This is primarily a check conducted to see whether such recalls occur before 1998.

P. 34-35 shows average and median daily salary within each year of the primary employment spells. This is a check of the salary variable. The salary contain small data breaks, however note that the median salary does not seem to have any significant break.

Name of dataset: all_prim_employment_spell

General information

The dataset consists of all primary employment spells for the spelldata population. We have not limited the spelldata population at the moment, but we will later limit it by only considering persons in dataset ’Personer’ aged between 15 and 75.

Variables in dataset

- Personnummer, person id variable.

- Firm id, firm id variable. We use 'senr' from 1985-2004 and 2006-2007 and we use 'cvrnr' for period 2005 and 2008-2011 which is due to the fact that these variables are the ones that DST uses in Cons, Ras and BFL for the given periods. We will later make an time consistent firm identification variable probably using information from KOB. We observe a substantial amount of job changes in 2005 and 2008 which we think is due to the change in firm id variable. We expect the time-consistent firm id variable to solve this issue.
- Start date for spell, contains start date for spell.
- End date for spell, contains end date for spell.
- Dskod, contains 'dskod' for the subperiod of 1985-2007 which the spell covers. We observe persons which on same firm id in a continuous period have multiple values of 'dskod'. In this case we choose the value of 'dskod' which corresponds to the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked). See variable 'bool indicating multiple dskod subspell' and dataset 'all_mult_dskod_subspell'.
- Arbgnr8, contains 'arbgnr8' for the subperiod of 1985-2007 which the spell covers. We do not make any subspells for persons having more than one value of 'arbgnr' for the same spell as we do for multiple values of 'dskod', this is due to the fact that there is no logic explanation for this phenomenon. We will look at these records later. The number of records having this abnormality is less than 1%.
- bfl_dskod, contains 'ajo_prod_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers. 'ajo_prod_nr_fra_prod_job' is believed to be the equivalent of 'dskod' in DST dataset BFL (e-indkomst). As for 'dskod' we do observe persons which on same firm id in a continuous period have multiple values of 'ajo_prod_nr_fra_prod_job'. In this case we choose the value of 'ajo_prod_nr_fra_prod_job' which corresponds to

Documentation Spell dataset - all_prim_employment_spell the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked). See variable 'bool indicating multiple dskod subspell' and dataset 'all_mult_dskod_subspell'.

- bfl_arbgnr8 contains 'ajo_se_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers. 'ajo_se_nr_fra_prod_job' is believed to be the equivalent of 'arbgnr8' in DST dataset BFL. As for 'arbgnr' we do not make any subspells for persons having more than one value of 'ajo_se_nr_fra_prod_job' for the same spell as we do for multiple values of 'bfl_dskod', this is due to the fact that there is no logic explanation for this phenomenon. We will look at this records later. The number of records having this abnormality is less than 1%.
- Total hours worked, contains total number of hours worked for the whole duration of the spell. We have estimated number of hours worked using ATP-data for the period 1985-2007. We have directly used the variable containing number of hours worked for the period 2008- 2011 which we get from dataset BFL.We keep

yearly number of hours worked for every year in-between the start date and end date of the spell in subspell dataset 'all_lon_subspell'.

- Total salary, contains total salary for the whole duration of the spell. We keep yearly salary for every year in-between the start date and end date of the spell in subspell dataset 'all_lon_subspell'.
- bool indicating secondary employment subspell, boolean variable which indicates whether or not there exists a secondary employment spell mapping to the primary employment spell. Please see dataset 'all_sec_employ_subspell' for more information about this 'mapping'.
- bool indicating multiple dskod subspell, as argued the primary employment spell might have multiple values of 'dskod' and/or multiple values of 'bfl_dskod'. We save the monthly records for any such spell in dataset 'all_mult_dskod_subspell'. 'bool indicating multiple dskod subspell' is a boolean variable which indicate whether or not there exists such a multiple dskod subspell.
- Person spell id, is an identification key mapping from the primary employment spell to matching subspell records. Combinations of pnr and person spell id are unique.

General information

The dataset consists of all primary employment spells for the spelldata population. We have not limited the spelldata population at the moment, but we will later limit it by only considering persons in dataset 'Personer' aged between 15 and 75.

Variables in dataset

- Personnummer, person id variable.
- Firm id, firm id variable. We use 'senr' from 1985-2004 and 2006-2007 and we use 'cvrnr' for period 2005 and 2008-2011 which is due to the fact that these variables are the ones that DST uses in Cons, Ras and BFL for the given periods. We will later make an time consistent firm identification variable probably using information from KOB. We observe a substantial amount of job changes in 2005 and 2008 which we think is due to the change in firm id variable. We expect the time-consistent firm id variable to solve this issue.
- Start date for spell, contains start date for spell.
- End date for spell, contains end date for spell.
- Dskod, contains 'dskod' for the subperiod of 1985-2007 which the spell covers. We observe persons which on same firm id in a continuous period have multiple values of 'dskod'. In this case we choose the value of 'dskod' which corresponds to the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked). See variable 'bool indicating multiple dskod subspell' and dataset 'all_mult_dskod_subspell'.
- Arbgnr8, contains 'arbgnr8' for the subperiod of 1985-2007 which the spell covers. We do not make any subspells for persons having more than one value of 'arbgnr' for the same spell as we do for multiple values of 'dskod', this is due to the fact that there is no logic explanation for this phenomenon. We will look

at these records later. The number of records having this abnormality is less than 1%.
- bfl_dskod, contains 'ajo_prod_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers. 'ajo_prod_nr_fra_prod_job' is believed to be the equivalent of 'dskod' in DST dataset BFL (e-indkomst). As for 'dskod' we do observe persons which on same firm id in a continuous period have multiple values of 'ajo_prod_nr_fra_prod_job'. In this case we choose the value of 'ajo_prod_nr_fra_prod_job' which corresponds to 1

Documentation Spell dataset - all_prim_employ_spell

the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked). See variable 'bool indicating multiple dskod subspell' and dataset 'all_mult_dskod_subspell'.

- bfl_arbgnr8 contains 'ajo_se_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers. 'ajo_se_nr_fra_prod_job' is believed to be the equivalent of 'arbgnr8' in DST dataset BFL. As for 'arbgnr' we do not make any subspells for persons having more than one value of 'ajo_se_nr_fra_prod_job' for the same spell as we do for multiple values of 'bfl_dskod', this is due to the fact that there is no logic explanation for this phenomenon. We will look at this records later. The number of records having this abnormality is less than 1%.
- Total hours worked, contains total number of hours worked for the whole duration of the spell. We have estimated number of hours worked using ATP-data for the period 1985-2007. We have directly used the variable containing number of hours worked for the period 2008- 2011 which we get from dataset BFL.We keep yearly number of hours worked for every year in-between the start date and end date of the spell in subspell dataset 'all_lon_subspell'.
- Total salary, contains total salary for the whole duration of the spell. We keep yearly salary for every year in-between the start date and end date of the spell in subspell dataset 'all_lon_subspell'.
- bool indicating secondary employment subspell, boolean variable which indicates whether or not there exists a secondary employment spell mapping to the primary employment spell. Please see dataset 'all_sec_employ_subspell' for more information about this 'mapping'.
- bool indicating multiple dskod subspell, as argued the primary employment spell might have multiple values of 'dskod' and/or multiple values of 'bfl_dskod'. We save the monthly records for any such spell in dataset 'all_mult_dskod_subspell'. 'bool indicating multiple dskod subspell' is a boolean variable which indicate whether or not there exists such a multiple dskod subspell.
- Person spell id, is an identification key mapping from the primary employment spell to matching subspell records. Combinations of pnr and person spell id are unique.

The dataset consists of all primary employment spells existing in the income tax reports *CON* 1985-2004, *RAS* 2005-2007, and *BFL* from 2008. We have not limited the spell data population, but the *IDAN* employment data only includes persons with permanent address end of the year.

5 Secondary employment subspell dataset *SUBSPELL_E_SEC*

General information

The dataset contains all secondary employment subspells. We create secondary employment subspells whenever a given person is employed at more than one firm at the same period. Simply speaking, we choose between primary and secondary firms of employment by looking at number of hours worked at the different firms. For each firm we sum hours worked at the firm in the current month plus the two following months. In every month we choose the firm with the highest calculated sum of hours worked as the primary employment firm. Please check separate documentation for more information about how we construct primary and secondary employment spells/subspells.

We do not create any subspells for our secondary employment subspells meaning that we do not keep monthly records of data with multiple values for dskod/bfl_dskod nor do we keep yearly number of hours worked and salary. We can construct this data if it should be of any interest.

Variables in dataset

- Personnummer, person id variable.
- Firm id, firm id variable. We use 'senr' from 1985-2004 and 2006-2007 and we use 'cvrnr' for period 2005 and 2008-2011 which is due to the fact that these variables are the ones that DST uses in Cons, Ras and BFL for the given periods. We will later make an time consistent firm identification variable probably using information from KOB. We observe a substantial amount of job changes in 2005 and 2008 which we think is due to the change in firm id variable. We expect the time-consistent firm id variable to solve this issue.
- Start date for spell, contains start date for spell.
- End date for spell, contains end date for spell.
- Dskod, contains 'dskod' for the subperiod of 1985-2007 which the spell covers. We observe persons which on same firm id in a continuous period have multiple values of 'dskod'. In this case we choose the value of 'dskod' which corresponds to the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked).
- Arbgnr8, contains 'arbgnr8' for the subperiod of 1985-2007 which the spell covers.

Documentation Spell dataset - all_sec_employ_subspell

- bfl_dskod, contains 'ajo_prod_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers. As for 'dskod' we do observe persons which on same firm id in a continuous period have multiple values of 'ajo_prod_nr_fra_prod_job'. In this

case we choose the value of 'ajo_prod_nr_fra_prod_job' which corresponds to the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod' have the same amount of hours worked).
- bfl_arbgnr8 contains 'ajo_se_nr_fra_prod_job' for the subperiod 2008-2011 which the spell covers.
- Total hours worked, contains total number of hours worked for the whole duration of the spell. We have estimated number of hours worked using ATP-data for the period 1985- 2007. We have directly used the variable containing number of hours worked for the period 2008-2011 which we get from dataset BFL.
- Total salary, contains total salary for the whole duration of the spell.
- Person spell id, contains an identification key which maps the secondary employment spell to a single primary employment spell. In principle any secondary employment subspell could coincide with more than one primary employment spell. We only map the secondary employment subspell to the first (sorted by start date) primary employment spell which the secondary employment subspell coincide with.

6 *SPELL_E_U_N*

Source: Short description of spell dataset: *SPELL_E_U_N*. October 24, 2016.

We merge our employment and unemployment spells. For overlapping employment and unemployment spells, we choose the spell with highest average daily hours spent in the spell. We create residual spells for all periods where an individual is not either employed or unemployed. We set state = 'N' for these spells.

The E_U_N spell dataset contains the following variables

- pnr, person identifier.
- startdato, start date of the spell.
- slutdato, end date of the spell.
- avg_timetal_daily, average daily hours spent in the spell.
- avg_bel_daily, average daily amount of salary(unemployment benefit).
- state, can take value 'U', 'E', or 'N', i.e. unemployment, employment or residual spell state 'N'.
- spell_id_U, identifier of the unemployment spell. The combination (pnr, spell_id_U) uniquely identifies an unemployment spell.
- spell_id_E, identifier of the employment spell. The combination (pnr, spell_id_E) uniquely identifies an employment spell.

Abstract

I describe how we merge the constructed employment and unemployment spells in order to create a spell dataset in which an individual is either in state 'E', 'U' or 'N' (employment, unemployment or residual state).

Important folders

The main folder used data is called Et samlet spell dataset. It is located on path G:
Data
Workdata
702728
FAWC2728
test. Within the folder, datasets are kept in folder Dataset, logs are kept in Logs, PDF output of the programmes are kept in PDF, and SAS programmes are kept in programme.

Introduction We merge the constructed employment and unemployment spells for period 1985-2013. We construct an algorithm which chooses between the constructed employment and unemployment spells whenever to such spells overlap. Note that at this moment we do not smooth/overwrite any of the resulting 'E', 'U', and 'N' spells. E.g., short periods of 'N' spells between two 'E' spells at same employer could be overwritten such that the three spells, 'E'-'N'-'E', would result in one 'E' spell spanning the whole period.

Description of the individual SAS programmes

01-join_E_U_v3 The purpose of this SAS programme is to find which employment and unemployment spells over- lap. We create boolean variables for both employment and unemployment spells which indicate whether the employment(unemployment) spell overlaps an unemployment(employment) spell.

The resulting datasets of this programme are

- _01_E_spells_bool_overlay,
- _01_U_spells_bool_overlay,
- _01_all_U_joined_records.

The analysis of this programme shows that 18% of all employment spells overlap one or more unemployment spells.

02-join_E_U_v3 The purpose of this programme is to define primary state for employment and unemployment spells which overlap.

We first check whether the start date(end date) of the unemployment spell is within five weekdays of the start date(end date) of the employment spell. If so, then we set state equal to 'U' for the overlapping period since the start date and end date of the unemployment spell are more credible than the start date and end date of the employment spell. E.g., consider a primary employment spell with start date 1. of January and end date equal to 31. of January and an unemployment spell with start date equal to 28. of January and end date equal to 15. of March. In this case the overlap between the two spells is within five weekdays of the end date of the employment spell, thus we set primary state equal to 'U' for the period 28. of January to 15. of March and primary state equal to 'E' for the period 1. of January to 27. of January. In the case where the overlap is not within five weekdays then we choose the primary state based on the average number of hours employed/unemployed for the two spells. E.g., consider again an employment spell with start date 1. of January and end date equal to 31. of January and an unemployment spell with start date equal to 14. of January and end date equal to 15. of March. The overlapping period (14. of January to 28. of January) is not within five weekdays of the employment spell, thus we choose the primary state of this period based on which of the spells have the largest amount of average hours in the spell. Assuming that the employment spell has the largest amount of average hours then the primary state of the period 1. of January to 31. of January is 'E' and the primary state of the period 1. of February to 15. of March is 'U'.

The resulting dataset of this programme is

_02_temp_E_U_merged_spells. 03-join_E_U_v3 The purpose of this SAS programme is to add employment and unemployment spells which in programme 01 was found not be overlapping to the spells resulting from programme 02. Further, we create residual spells for periods where an individual is not in either state 'E' or 'U'. The residual state is named 'N'.

The resulting dataset of this programme is

_03_E_U_N_merged_spells. 04-Analyse_v1 The purpose of the SAS programme is to conduct an analysis of the constructed E-U-N spell data. We count the stock of individuals in employment, unemployment, and in the residual state at the first day of each week between 1985-2013. Further, we compute average duration, average daily salary/benefit, and average hours spent of spells within the three states. 1p. 1 of the PDF.

2p. 24 of the PDF.

7 Yearly and Yearly Smoothed spells

7.1 SPELL_E_N_YEARLY_SMOOTH

Folder and programme

The folder containing programmes, data, output, and log files is located at path
G:
Data
Workdata
702728
FAWC2728
test
September2017
EmploymentSpells. The SAS- programme used to construct the dataset is called SpellsENYearlySmoothv2. The resulting dataset is called SPELL_EN_YEARLY_SMOOTH.

Constructing the dataset

We construct yearly smoothed E-N spell data using spell datasets SPELL_PRIM_EMPLOY and SUBSPELL_LOEN. First, we merge the two datasets which result in a yearly employment spell dataset. We alter end dates of spells in order to ensure that employment spells do not overlap each other. The employment spell data intentionally include overlapping employment spells within a month. Whenever two employment spells overlap within a month we alter the end date of the first spell such that it ends one day prior to the second employment spell.

Next, we smooth employment spells. We identify subsequent employment spells for which an individual made a job-to-job transition or a recall. We alter the end date of the first employment spell to the day prior to the start date of the second employment spell for job-to-job transitions with a maximum gap of 14 days between two employment spells. For recall employment with a maximum gap of 93 days between two employment spells, we combine the two employment 1There exists few cases in which one of the overlapping spells starts later than the other and ends prior to the other. E.g. consider spell A covering 1st of March to 31 of March and spell B starting 5th of March and ending 20th of March. In this case, spell A is split into two spells, i.e. one spell from 1st of March to 4th of March and one spell from 21st of Match to 31 of March, and spell B is left unchanged.

Two subsequent employment spells at two employers is referred to as a job-to-job transition. Two subsequent employment spells at the same employer is referred to as a recall.

spells into one employment spell covering the entire period between the start date of the first employment spell to the end date of the second employment spell.

We do not alter hours worked or earnings for smoothed recall employment spells or smoothed job-to-job transition employment spells for which the job-to-job transition occur within the same year. For smoothed job-to-job transitions spells covering two years, we do not alter earnings and hours in the first year. In the second year, we set earnings (hours worked) equal to average earnings (hours worked) per day in the first year of the employment spell tines the duration in days of the employment spell in the second year.

We create yearly non-employment spells in all gaps between employment spells. We create a non-employment spell id named spell_id_N.

Last, we rename and collapse variables compared to spell data SPELL_PRIM_EMPLOY and SUBSPELL_LOEN. We rename yr_lontimer_estimated to timer, yr_lonblb to beloeb, startdato to start_dato, and slutdato to slut_dato. We collapse spell_id_16 and spell_id_N to one variable called spell_id.

List of variables

- pnr, person identifier,
- start_dato, start date of spell
- slut_dato, end date of spell
- spell_firm_id, spell firm identifier
- state, state of spell, employment spell (E) non-employment spell (N)
- spell_id identifying different spells within combinations of (pnr and state)
- aar, year of spell
- beloeb, earnings
- timer, hours worked estimated using Lund/Vejlin algorithm

31.6% of employment spells are smoothed due to job-to-job transitions and 8.62% percent are smoothed due to recall employment.

As start and end dates of spells are the more inaccurate information compared to yearly earnings information.

E.g. consider an employment spell ending 20th of December 1991 and a subsequent employment spell starting 4th of January 1992 at two different employers. We smooth the first employment spell such that it ends 3rd of January. We set earnings and hours worked equal to three times the average earnings and hours worked of the employment spell in 1991. For the part of the smoothed employment spell in 1991 we do not alter earnings or hours worked.

8 *SUBSPELL_LOEN*

Name of dataset: subspell_lon

General information

The dataset consists of subspells containing yearly salary and yearly number of hours worked for a given year in the duration of the primary employment. We create such subspells for all years in the duration of the primary employment spell. The combination of pnr and person spell id tells which primary employment spell that each record maps to.

Variables in dataset

- Personnummer, person id variable.
- Start date for spell, contains start date of the subspell. It might be the case that the duration of the subspell is less than an year which occurs when the duration of the primary employment spell is less than an year.
- End date for spell, contains end date of subspell.
- Total yearly hours worked, contains yearly number of hours worked for the given year of the primary employment spell. We have estimated number of hours worked using ATPdata for the period 1985-2007. We have directly used the variable containing number of hours worked for the period 2008-2011 which we get from dataset BFL.
- Total yearly salary, contains yearly salary for the given year of the primary employment spell.
- Year, contains the year of the subspell.
- Person spell id, contains an identification key which maps the subspell to the primary employment spell from which the subspell was created.

9 Constructing spell firm identifiers

9.1 Purpose

A *spell-firm-id* and *spell-arb-id* are constructed from existing datasets. Two versions of firm identifiers are constructed. The first version constructs a *CVRNR* for the entire period mapping *SENR* to *CVRNR* using *ESR_SE_CVR*. It is important that the spell-firm-identifier can match the firm identifier in *IDAN* and identifies an economic unit, i.e. being matched with the *CVRNR* in *FIRM, Firm statistics*. The second version constructs an over-time *CVRNR* or *OKNR* which attempts to identify the same *economic unit* over time by taking into account changes in *CVRNR*, merges and splits or off-springs. In the first version of firm identifiers the main source is the reporting of taxable income from jobs paying taxable income. Before 2008 the source datasets are *CONESR* and *RAS_CONESR*. From 2008 the source dataset is *BFL*.

The spell workplace identifier identifies a physical or fictitious workplace. A physical workplace is named an establishment. It is a well-defined geographical unit which produces a few products with a well-defined group of workers.

In the first version of the workplace identifier the main sources are the same as for the firm identifier. In the period 1980 to 1994 the variable used is *ARB_STK*, from 1995 *ARB_NR* is used.

The second version of the spell workplace identifier uses the *IDA* variable *LBNR* which defines an over-time workplace identifier based on the concept of a permanent workplace.

9.2 Spell firm identifier

A job spell is terminated either by being associated a new firm identifier or by a gap of more than 14 days without being associated a new firm identifier.

An economic unit my have several fiscal or legal identifiers. If a job is being reported using several of these identifiers it creates a fictitious job change/new spell. An *OKNR* may be associated one or more *CVRNR*. And a *CVRNR* may be associated one or more *SENR*. In order to avoid fictitious job changes *OKNR* is used in the first version of spell firm identifier, or a *CVRNR* chosen by Statistics Denmark may also be used to carry the same information as *OKNR*.

An ongoing project in Statistics Denmark, DRDS Firm Warehouse, will produce an over-time *OKNR* using more data than job-flows. Their identifiers will only be constructed from 2010 and onwards.

9.3 Firm-id before 2008

The source for firm-id for employment spells before 2008 is *CONESR*. This dataset contains *annual reports* for *earnings and withheld taxes* for all persons working in Denmark.

The dataset has 4 firm identifiers:

- *SENR, A firm account number for tax and custom payments*, all years
- *ARBGNR, IDA identifer for unit reporting wages for taxation, without sector prefixIDA identifier unit reporting wages for taxation, without sector prefix.*, all years. It is equal to *SENR*
- *ARBGNR9,* , all years
- *ARBGNR8, An old IDA identifier for unit reporting wages for taxation without sector prefix*, 1980-1984

All *SENR* can be matched to a *SENR* in dataset *ESR_SE_CVR*.

SENR is only associated one *CVRNR* at any date, but a *CVRNR* can be associated many *SENR* at any date.

For each (year, SENR) the dataset *ESR_SE_CVR* is used to find the *CVRNR* it is associated at the end of the year. *SENR* is associated in the middle of a year.

This *CVRNR* is used as the first spell firm-id for the years 1980-2007.

10 ANALYSIS OF IDAN_FIDA_MIA

Chapter 5
Raw Unemployment spells

The jobs construct raw unemployment spells *RAW_UNEMPLOYMENT*.

The dataset contains the following variables:

Table 12 List of Variables - SPELL_U

Variable name	Title
PNR	*Person identifier, encrypted CPRNR number*
JOB_START_DATO	*In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*
JOB_SLUT_DATO	*In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*
BELOEB	*Earnings or benefits in the spell*
BOOL_SUBSPELL_CHANGE_OF_SOURCE	*Boolean variable indicating whether the unemployment spell is constructed using more than one DST register.*
SPELL_ID	*SPELL, An identifier of a spell*
STATE	*State for a spell: U: All observations have state U.*
TIMER	*Hours worked estimated using Lund/Vejlin algorithm*

The SAS programmes used to construct unemployment spells are documented below. The jobs use registers LC and SHSS for the period 1984-2007, and ILME and OF from 2008.

The unemployment registers change in 2007 from LC and SHSS to ILME and OF which among other things improve the quality of from- and to-date of the spells. It is possible to compare the two data sources in 2007. The constructed raw unemployment spells do not contain a break in 2007; thus there are no potential problems arising from the change in data sources.

The unemployment spells have been constructed such that they contain individuals who receive benefits for which it can be assumed that they are actively searching for a job. Thus, the unemployment spells constructed primarily contain individuals who receive cash benefits (kontanthjælp) or unemployment insurance benefit (dagpenge). Therefore, the unemployment spells do not contain individuals who participate in active labour market programmes or individuals who are on leave as these individuals cannot be assumed to be actively searching for a job. It is planned to add separate active labour market programmes and leave spells in a later version.

Description of the individual SAS programmes:

00_Driver_before_2008.v1 The driver programme runs the SAS jobs needed for generating the unemployment spells. The driver job can be edited using the following SAS macro variables:

- *INOBS*, controls how many records are read: max or an integer number.
- *BOOL_DELETE*, deletes temporary files after each step.
- *BOOL_PDF_ON*, controls generation of PDF files for description and debug.
- LIB_IN and NAME_IN_DATA define SAS library and file name for in data.
- LIB_OUT, library for storing permanent output data. Test and temporary data are stored in \temp. The files must be deleted manually when they are not needed in the job anymore.
- END_YEAR_SHSS_BEL, last year for dataset _01_SHSS_BEL
- END_YEAR_SHSS_SPELL, last year for dataset _02_SHSS_SPELL
- START_YEAR and END_YEAR, determine the period for which the spells are generated
- START_YEAR_SUB_SPELLS and END_YEAR_SUB_SPELLS, determine the period for which sub-spells and other spells than UNEMPLOYMENT are generated

00 CRAM_kalender CRAM week number 1 starts in week 50-52 in the previous year.

Job *cram_kalender* reads an excel file with information about the relation of CRAM weeks and calendar weeks. It creates the yearly file *cram_kalender*. Both files are stored in \data\workdata\702728\spell\datasets\unemployment_spells\cram_kalender.

The job creates the variables:

- week_1_cram_in_real_week for weeks 1,2,3, mapping from cram_week_number for real week_number.
- max_week_cram, the number of cam weeks in this year
- difference, the difference in number of weeks for calendar week and cram week for weeks 4 ....52/53

_01_SHSS_before_2008_2021.v1 *SHSS* is used for two tasks:

- It is used to add amounts to the unempoyment spells. This requires that monthly data is generated.
- It is used to compute both spell dates and amounts to all other spells than unemployment spells and for sub-spells.

It generates two datasets:

The dataset unemploy SHSS_BEL covers the years 1985-2007. From 2008 the dataset *ILME* is used.

The dataset unemploy SHSS covers the years 1985-2006. From 2007 the dataset *OF* is used.

SHSS is a yearly dataset where yearly amounts of benefits are stored in BELxx and the monthly duration are stored in different variables VARMMxxmm, where xx is a code identifying the type of benefit and mm identifies the month of the duration, e.g., VARMMV10 with value 30 refers to an individual who received pension benefits for 30 days during the 10th month (October). The value of the variables is an integer in the interval [0, 30]. The mapping between VARMMxx and BELxx variables is inferred as it is not documented by DST. The mapping can be found in the following APPENDIX.

The job has two SAS macros:

SAS Macro TRANSPOSE_VAR The macro has four arguments:

1. *SHSS* variable name VARMMxxmm containing the duration, assigned to variable VARIGHED in new dataset
2. Name of new dataset (*TRANSPOSE_SHSS*)
3. Month identified by mm in the *SHSS* variable name, assigned to variable MONTH in new dataset
4. Name for duration this month for benefit xx, VARMMxx, assigned to variable KODE in new dataset

For each record with a positive duration a monthly record is output to dataset *transpose_shss* with three more variables: MONTH which contains the month, KODE which contains the duration variable name without month, i.e. VARMMxx, and VARIGHED which is the value of the argument VARMMxxmm.

STEP 1 This step extracts the relevant variables from dataset *SHSS*. SHSS only contains data about durations and received benefit, therefore, start-date and end-date must be imputed.

A new temporary dataset named *transpose_shss* is created from *SHSS*using macro transpose.

The macro is called with the arguments *VARMMxxmm, TRANSPOSE_SHSS, mm, VARMMxx*

for mm = 1,2,...,12

for xx = all unemployment codes, all leave codes, all labour market codes, and all pension codes, see APPENDIX.

STEP 2 The amounts are added to *transpose_shss*, such that a new dataset *transpose_sort_bel_2* where each record will contain the yearly benefits, BELXX, MONTH, name of benefit, VARMMxx, and monthly duration , VARIGHED.

Number of monthly hours associated with the benefit is calculated as timer = 160x(varighed/30)

STEP 3 Temporary dataset transpose_shss_sort_bel2 containing duration aggregated to yearly duration in order to match the yearly benefits BELLXX.

The yearly benefits for unemployment are calculated as bel_unemp = BEL_ARB + BEL_AF + BEL_AK. For now amounts for other benefits are not calculated.

Next a new dataset transpose_shss_sort_bel3 is created containing monthly data on duration and benefits. The yearly benefits are allocated according to the fration of duration in this month.

Now durations are grouped into

- U, unemployment, as a sum of VARMMA, VARMMAFA, VARMMAFB, VARMMAFT,
- N, others, is a sum of VARMMF, VARMMK, VARMML, VARMMLYD, VARMMO, VARMMSF, VARMMSS, VARMMU, VARMMV

Durations and amounts for records with 'KODE_AGG' = 'UNEMPLOYMENT' come from the following variables:

- VARMMA: Unemployment benefits excl. recipients of cash benefits. Amount received: 'BEL_ARB'.
- VARMMAFA: Cash benefit. Amount received from: 'BEL_KON1', 'BEL_KON2', 'BEL_KON3',
- VARMMAFB: Cash benefit. Amount received from: 'BEL_KON1', 'BEL_KON2', 'BEL_KON3', 'BEL_KON194', 'BEL_KON294', 'BEK_KON394', 'BEL_KON494', 'BEL_KON598', 'BEL_KON699'.
- VARMMAFF: Unemployment benefits (for years 2001-2005). Amount received: 'BEL_LYD'.
- VARMML: Unemployment benefits (for years 2006-2007). Amount received: 'BEL_LYD'.

inds't 5)

Now records with 'KODE_AGG' = unemployment are aggregated. Multiple records exist whenever an individual changes benefit within 'KODE_AGG' = UNEMPLOYMENT, e.g. an individual changing from unemployment insurance benefit (dagpenge) to cash benefit (kontanthjælp). Yearly amounts of received benefits for records with 'KODE_AGG' = UNEMPLOYMENT are computed.

JOB_START_DATO and *JOB_SLUT_DATO* will be defined using the following algorithm.

Consecutive months with 'VARMXXYY' >0 are regarded as a spell. *JOB_START_DATO* of the spell is set equal to the first day in the month if 'VARMMXXXX' = 30. If 'VARM' <30 then we look at the next months' value of 'VARM' for the same individual on the same code. If this next months' 'VARMM' <30 then start date is set equal to first day of the month, otherwise if it is equal to 30 then start date is set equal to the last day of the month minus the value of 'VARM'. E.g. 'VARMMXX-01' = 15 and 'VARMMXX-02' = 20. In this case start date is set equal to first of January as the unemployment spell is considered to be a part-time unemployment spell. E.g. 'VARMXX-01 = 15 and 'VARMXX-02' = 30. In this case, we believe that the unemployment spell is a full-time unemployment spell and we set start date equal to 15th of January.

JOB_SLUT_DATO is defined in the same way, i.e. end date is set equal to the last day of the month if 'VARMM' = 30. Otherwise, if 'VARMM' <30 and the previous month' 'VARMM' <30 then we set end date equal to the last day of the month. If the previous month' 'VARMM' = 30 then we set the end date equal to the first day of the month plus 'VARMM'.

In this version, only unemployment spells are created.

The analysis of the programme shows that around 1-4 percent of all records have yearly amount of received benefit ('BEL_YEARLY') equal to zero.

DST documents that around 2 percent of records in SHSS have amount of received benefit equal to zero due to faulty data.

STEP 4 In this the year all benfits are imputed for each KODE value VARMMxx. For some different xx different BELyy are used over the years.

item - DETTE SKAL JEG LIGE HAVE CHECKET

_02-LC_before_2008_2021_v1 . The *SHSS* is not used for determining the start/end dates for unemployment spells; instead *LC* is used. [5]

The resulting dataset is unemploy._02_LC.

A debugging dataset temp._lc_temp_3 can be generated.

LC contains weekly unemployment data for years 1985-2007. LC is used to construct unemployment spells for years 1985-2006. LC does not contain information about amount of received benefit. The information extracted from SHSS in programme *_01_SHSS_before_2008_2001_v1* is used to get information on amount of received benefits for each unemployment spell.

This SAS programme extracts data from LC and prepare the data in order to make unemployment spells. The dataset contains variables LGRADWW and LARSAGXX. WW where 'WW' is CRAM week number. Variable LGRADWW has values in [0, 1000] and LGRADWW = 1000 corresponds to five days of unemployment, LGRAD = 800 corresponds to four days of unemployment etc.

The weeks and years in LC are 'CRAM-weeks' meaning that they follow the 'CRAM-calendar'. 'CRAM-weeks' are usually, but not always, lagged by two weeks compared to the calendar week. 'CRAM-weeks' are converted to calendar weeks using a mapping file which maps 'CRAM-weeks' to calendar weeks.[6].

Variable 'LARSAG' can take seven different values.[7] 76 percent of the weekly records are unemployment insurance benefit (dagpenge) coded as 'LARSAG' equal to missing. Another 16 percent of the weekly records are cash benefits (kontanthjælp) coded as 'LARSAG' = 8. $'LARSAG' \in \{2,4,5,6\}$ are holiday benefits, and $'LARSAG' \in \{1,2,3\}$ are various special unemployment benefits agreed on with specific unions. These special agreements only account for 1 percent of the weekly records.

The weeks are compressed to spells identified by *JOB_START_DATO* and *JOB_SLUT_DATO*.

Variable 'KODE_AGG' has the following values 'LEAVE' if $'LARSAG' \in \{4,5,6\}$, otherwise 'KODE_AGG' takes value 'UNEMPLOYMENT'.

The dataset also contains the following variables from *LC* [8]

---

[5] In DS documentation the *LC* dataset is named CRAM.

[6] See *CRAM_kalender*

[7] See `https:/lmdg.econ.au.dk/datasets_doc/lc_vde.pdf`

[8] See `https:/lmdg.econ.au.dk/datasets_doc/lc_vde.pdf`

- *AKASSE, FSE variable, see AKASSE_KODE*
- *EFMARK, Indicator for postemployment payment, 0: does not receive postemployment payment, 1: does receive postemployment payment*
- *FORSIKRINGS_KATEGORI_KODE, Code for category of insurance*
- *FORSIKREDE_TIMER, Number of hours per week covered by unemployment insurance.*

The variables LGRADWW and LARSAGWW are converted to a new variable *LGRAD_LC* and *LARSAG_LC* of weekly data identified by *YEAR_LC* and *WEEK_LC*. Weeks 1-3 are hand coded weeks, weeks 4-9 are converted using %macrooutput AND weeks 10-53 using %macrooutput10.

_03_LC_before_2008_2021_v1 Unemployment spells are created using unemploy.lc. Consecutive weeks with lgrad $>$ 0 and 'kode_agg' = 'unemployment' are regarded as one spell.

Start date and end date are defined using the following algorithm for full-time and part-time:

Full-time unemployment, LGRAD = 1000 Consecutive weeks with LGRAD = 1000 and KODE_AGG = 'UNEMPLOYMENT' is considered a spell. The first and last week may have LGRAD ¡ 1000 and the start and end date are modified with the days defined by LGRAD for these weeks.

E.g. consider records where 'lgrad' = 600 and next month's 'lgrad' = 1000. In this case start date is set equal to Monday of the week incremented by $5-3=2$ days resulting in the start date of the spell equal to Wednesday of the week. End date is defined in the same way.

Part-time unemployment, LGRAD ¡ 1000 Consecutive weeks with LGRAD = 1000 and KODE_AGG = 'UNEMPLOYMENT' is considered a spell. LGRAD for the spell is the average over these weeks.

For each weekly observation, 'number of hours' unemployed is computed. These number of hours are only going to be used when merging employment and unemployment spells together. If an overlap between these is found the number of hours unemployed and number of hours worked are compared in order to define which state, employment or unemployment, is chosen to be the primary state. Number of hours unemployed is computed as 37 ' 'LGRAD' 1000. For the compute spells NUMBER_OF_HOURS are summed over the days in the spell.

A boolean variable 'bool_multiple_larsag' is created which indicates whether the unemployment spell has more than one value of 'larsag', e.g. an unemployment spell where the individual changes from unemployment insurance benefit to social assistance sometime during the unemployment spell. We construct subspells for these cases, i.e. we save the weekly records in a subspell dataset for all unemployment spells where 'bool_multiple_larsag' = 1. We index each LC unemployment spell by 'lc_spell_id' and we index each subspell by 'lc_subspell_id'.

04-LC_before_2008_2021_v1 Dataset SHSS contains information on the yearly amount of received unemployment benefits. We merge this information from programme 01 onto each computed LC unemployment spell from programme 03. For each unemployment spells and for each year the unemployment spell span, we compute the duration in days of the unemployment spell and we compare this to the total duration in days of all unemployment spells for the given individual in the given year. Let 'day_dur' equal the duration in days of the unemployment spell and 'total_day_dur' equal the total duration in days of all unemployment spells. We now assign the amount equal to day_dur total_day_dur

05-ILME_from_2008_2021_v1 The purpose of this SAS programme is to extract information on the amount of benefit received for a given unemployed individual for years 2008 to 2013 using DST register ILME. The variables of interest are 'vmo_startdato', 'vmo_slutdato', 'vmo_indkomst_type_kode', and all the variables which describe the amount of benefits received. The first two variables contain information about start date and end date, and 'vmo_indkomst_type_kode' contain information about which kind of benefit the individual received. The reason why we do not use ILME as the source for constructing unemployment spells is that the variable 'vmo_indkomst_type_kode' does not deliver as detailed information as we can find in dataset OF. We find that variables containing information about the amount of received benefits can be negative. However, if we aggregate the amounts for a consecutive period of received benefits the aggregated amounts are almost always non-negative. We conclude that the negative amounts are corrections. Therefore, we aggregate the amounts (negative and positive) when constructing ILME spells. An ILME spell is constructed as a consecutive period of received benefits on the same 'vmo_indkomst_type_kode'. ILME spells with 'vmo_indkomst_type_kode' equal to '04', '06', or '24' are regarded as potential unemployment spells, thus these spells will be merged onto unemployment spells constructed using OF for 2008 to 2013. We use the term potential unemployment spells since 'vmo_indkomst_type_kode' doe not uniquely determine whether the ILME spell originates from a spell of unemployment, pension, leave, etc. E.g. 'vmo_indkomst_type_kode' = 04 is used for unemployment insurance payments, pension payments, flex job payments, etc. We truncate the created ILME spells at 31. of December 2013. We reduce all amount variables by the duration in days in the period 2008-2013 compared to the total duration of the ILME spell. The resulting dataset of this programme is

06-OF__from_2008_2021_v1 The purpose of the OF SAS programmes is to make unemployment spells using dataset OF. OF consists of almost all benefits for individuals aged between 16-64. We extract information for the period 2007-2013. The purpose of this SAS programme is to extract data from OF and make unemployment spells. In the next programme, we add amount of received benefit to the constructed unemployment spells. The variables of interest are 'pti_vfra', 'pti_vtil', 'pti_timer_per_uge', and 'pti_tilstand_kode'. The variables contain information about start date and end date, the amount of hours per week, and the kind of benefit received. 8DST documents that around 2data.

We create variable 'kode_agg' which aggregates the different values of 'pti_tilstand_kode' into unemployment, active labor market programmes, pension, and leave.9 A full list of the assignment of the values of 'pti_tilstand_kode' to 'kode_agg' can be found in the appendix. We truncate the OF records if end date exceeds 31. of December 2013. We compute the total number of hours for each record as 'days' ' 'pti_timer_per_uge' 7 , where 'days' is the truncated duration measured in days between start date and end date. We create OF unemployment spells by compressing consecutive records with 'kode_agg' = 'unemployment'. The existence of consecutive records on 'kode_agg' is due to either a change of 'pti_tilstand_kode' (within 'kode_agg' = 'unemployment') or a change of 'pti_timer_per_uge' (within same value of 'pti_tilstand_kode'). We set number of hours for the compressed record equal to the sum of hours for the consecutive records. We create OF subspells on 'pti_tilstand_kode' and we create a boolean variable 'bool_of_subspell' which indicates whether the OF unemployment spell has multiple values of 'pti_tilstand_kode'.10 We index OF unemployment spells by 'of_spell_id' and subspells by 'of_subspell_id'. We create a dataset containing the mapping between 'of_spell_id' and 'of_subspell_id'. The resulting datasets of this programme are

- _06_OF_spells_without_bel,
- _06_OF_subspells,
- _06_OF_subspells_mapping.

The analysis of this programme shows the distribution of average weekly hours 'spent' in unemployment.

_07-OF_from_2008_2021_v1 The purpose of the SAS programme is to merge information about the amount of benefits received onto each constructed OF unemployment spell. OF unemployment spells span the period 2007- 2013. ILME span the period 2008-2013, thus we also merge the unemployment spells with SHSS from programme 01 in order to get amount of benefits for 2007.

We merge variables 'vmo_a_indk_am_bidrag_fri' and 'vmo_b_indk_am_bidrag_fri' from ILME dataset constructed in programme 05. We compute the number of days the ILME spell overlaps the OF unemployment spell and we assign the fraction of benefits which corresponds to the fraction between the number of overlaying days compared to the total duration of the ILME spell. E.g., consider an OF spell spanning the period of 1. of January to 25. of January and an ILME spell spanning the period 10. of January to the 30. of January with benefits equal to 8.000 DKK. The number of overlaying days is sixteen and the total duration of the ILME spell is 21. In this case we assign 16 21 ' 8.000 DKK to the OF unemployment spell.

We merge yearly amount of received benefits from SHSS for year 2007 onto the OF unemployment spells. As for LC unemployment spells, we split the yearly amount of received benefits from SHSS onto the OF unemployment spells by the fraction between duration in days in 2007 of each OF unemployment spell and the total 2007 duration of all OF unemployment spells for a given individual.

The resulting dataset of this programme is We plan to add active labor market programmes, pension, and leave spells in a later version. 10We keep all 'pti_tilstand_kode' spells as subspells, i.e. for OF unemployment spell for which 'pti_tilstand_kode' changes and for OF unemployment spells for which it does not change. In this way, it is possible to merge 'pti_tilstand_kode' back onto the OF unemployment spells and e.g. to distinguish between individuals receiving unemployment insurance benefit and individuals who receive cash benefit.

08-Spells_full_period_2021_v1 The purpose of the SAS programme is to combine the LC and OF unemployment spells in order to construct unified unemployment spells for the spell period 1985-2013. We input LC unemployment spells covering period 1985-2006 from programme 04 and we input OF unemployment spells covering period 2007-2013 from programme 07. We compress LC and OF unemployment spells for which LC end date is equal to 31. of December 2006 and OF start date is equal to 1. of January 2007. We index the resulting unemployment spells by 'spell_id'. We create subspells for resulting unemployment spells which originates from both a LC and an OF unemployment spell, and boolean variable 'bool_subspell_change_of_source' indicates this event. We keep a mapping dataset which maps ('pnr', 'spell_id') into ('pnr', 'lc_spell_id', 'of_spell_id') such that it is possible to find corresponding LC and OF unemployment spells for all resulting unemployment spells. The resulting datasets of this programme are

- _08_U_spells,
- _08_U_subspells,
- _08_U_mapping.

The analysis of this programme consist of a time series plot and a count of the stock of individuals in the constructed unemployment spells at the first day of every month between 1985-2013. 11p. 6 and 8 of the PDF. 12p. 18 of the PDF. 7

11 Appendix

List of all values of variable 'pti_tilstand_kode' with corresponding value of kode_agg
Unemployment KODE_AGG = UNEMPLOYMENT consists of the following types:

- 5010: Ledige, detaljer uoplyst
- 5020: Ledige dagpengemodtagere
- 5030: G dage
- 5035: Arbejdsmarkedsydelse
- 5080: Ledige kontanthjælpsmodtagere
- 5090: Ledighedsydelse, detaljer uoplyst
- 5100: Ledighedsydelse mellem fleksjob
- 5110: Ledighedsydelse i visitionsperioden før første fleksjob
- 5140: Ledighedsydelse efter ustøttet beskæftigelse
- 7075: Ledighedsydelse SS
- 7090: Introduktionsydelse

Pension KODE_AGG = PENSION consists of the following types:

- 6010: Efterløn
- 6020: Overgangsydelse
- 6030: Fleksydelse
- 6040: Delefterløn

- 7020: Førtidspension SS
- 7025: Førtidspension

Leave KODE_AGG = LEAVE consists of the following types:

- 3110: Orlov til uddannelse
- 3120: Orlov til sabbat
- 3130: Orlov til børnepasning
- 4030: Sygedagpenge, detaljer uoplyst
- 5040: Feriedagpenge fra ledighed
- 5050: Feriedagpenge fra beskæftigelse
- 5060: Feriedagpenge o. a. ledhedsårsag 8
- 5070: Feriedagpenge, uoplyst
- 5120: Ledighedsydelse under ferie
- 5130: Ledighedsydelse under sygdom og barsel
- 5800: Syge på dagpenge eller arbejdsmarkedsydelse
- 6050: Nedsatte dagpenge (par. 32)
- 7030: Sygedagpenge SS
- 7035: Sygedagpenge, beskæftigede
- 7036: Sygedagpenge, ej beskæftigede
- 7037: Sygedagpenge, detaljer uoplyst
- 7040: Barselsdagpenge, SS
- 7045: Barselsdagpenge, beskæftigede
- 7046: Barselsdagpenge, ej beskæftigede
- 7047: Barselsdagpenge, detaljer uoplyst

Labour market training KODE_AGG = AKTIVERING consists of the following types:

- 1010: Jobtræning
- 1020: Individual jobtræning
- 1030: Ansættelse med løntilskud
- 1039: Jobrotation
- 1040: Virksomhedspraktik
- 1049: Nytteindsats
- 1050: Fleksjob
- 1055: Fleksjob, KMD-aktiv
- 1060: Fleksjob, selvstændige
- 1070: Skånejob
- 1075: Skånejob, KMD-aktiv
- 1080: Servicejob
- 1090: Arbejdspraktik
- 1100: Puljejob
- 1110: Rotationsansættelse
- 1120: Voksenlærlinge
- 1510: Opkvalificering iflg. Integrationsloven
- 1520: Ordinær uddannelse

- 1540: Voksen- og efteruddannelse
- 1550: Korte vejlednings- og afklaringsforløb
- 1560: Særligt tilrettelagte projekter og uddannelsesforløb
- 1562: Særligt tilrettelagte uddannelsesforløb
- 1564: Særligt tilrettelagte projekter
- 1570: Vejledning og introduktion
- 1580: Særligt aktiverende forløb
- 1590: Intensiv jobsøgning
- 1591: Vejl opkval ordinær udd.
- 1592: Vejl opkval øvrige forløb under 4 uger
- 1593: Vejl opkval øvrige forløb over 4 uger
- 1594: Vejl opkval øvrige forløb
- 1595: Mentorstøtte
- 1599: Vejledning og opkvalificering
- 1600: Særlig formidling
- 1800: Selvvalgt uddannelse i 6 uger
- 2010: Kursus i samfundsforståelse
- 2020: Danskundervisning
- 2030: Særligt tilrettel. danskunderv.
- 2510: Aktivering, detaljer uoplyst
- 2512: Aktiveringsgodtgørelse
- 2514: Forsørgelse under vejledning, opkvalificering og virksomhedspraktik
- 2516: Løntilskud til personer i tilbud efter kapitel 12
- 2518: Kontanthjælp og starthjælp under forrevalidering
- 2520: Forsøg
- 2530: Frivillige ulønnede aktiviteter
- 2540: Uddannelse m. voksenudd.støtte
- 3010: Etableringsydelse
- 3020: Igangsætningsydelse
- 4010: Revalidering, detaljer uoplyst
- 4012: Revalideringsydelse med virksomhedspraktik
- 4014: Løntilskud ved revalidenters ansættese med løntilskud
- 4020: Forrevalidering, detaljer uoplyst
- 7005: Jobafklaringsforløb
- 7010: Socialtilstand detaljer uoplyst
- 7015: Førtidspension, gammel ordning
- 7024: Ressourceforløb
- 7050: Kontanthjælp (passiv)
- 7055: Kontanthjælp SS
- 7060: Revalidering SS
- 7070: Forrevalidering SS
- 7080: Støtte til handicappede

Education KODE_AGG = EDUCATION consists of the following types:

Chapter 6
Dataset SPELL_EUN

We merge *SPELL_E* and *RAW_UNEMPLOYMENT*. For overlapping employment and unemployment spells we choose the spell with highest average daily hours spent in the spell. We create residual spells for all periods where an individual is not either employed or unemployed. We set state ='N' for these spells.

The *SPELL_E_U_N* contains the following variables:

Table 13 List of Variables - SPELL_EUN

Variable name	Title
AVG_TIMETAL_DAILY	*AVG_TIMETAL_DAILY, Average daily hours worked or receiving benefits in the spell.*
AVG_BEL_DAILY	*AVG_BEL_DAILY, Average daily amount of salary (E) or unemployment benefit (U) in the spell.*
PNR	*PNR, Person identifier, encrypted CPRNR number*
SPELL_ID_U	*SPELL_ID_U, SPELL, An indentifier of the unemployment spell.*
SPELL_ID_E	*SPELL_ID_E, SPELL, An identifier of the employment spell.*
JOB_START_DATO	*JOB_START_DATO, In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*
JOB_SLUT_DATO	*JOB_SLUT_DATO, In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*
STATE	*STATE, State for a spell: U: All observations have state U.*

Chapter 7
Dataset SPELL_EN_YEARLY_SMOOTH

This section describes how we create a yearly spell dataset containing employment (E) and non-employment (N) spells. We alter end dates of employment spells for job-to-job transitions and recall employment. We refer to this process as smoothing of employment spells.

12 Folder and programmes

The folder containing programmes, data, output, and log files is located at path

X:/Data/Workdata/702728/Spell/ programme/EmploymentSpells

The SAS- programme used to construct the dataset is called SpellsENYearlySmoothv2. The resulting dataset is *SPELL_E_N_YEARLY_SMOOTH*.

12.1 Description

SPELL_E_N_YEARLY_SMOOTH contains smoothed employment and non-employment spells allocated to a year and is constructed by merging *SUBSPELL_LOEN* from the spell on *SPELL_E*.

In the first step we allocate spells to a year. If a raw spell spans the beginning or end of a year the start date and end date are modified. The income and hours are allocated proportionally to the new spell.

In the second step we smooth the spells such that spells are not overlapping.

12.2 Constructing the Datasets

SPELL_E_N_YEARLY_SMOOTH contains smoothed employment and non-employment spells allocated to a year and is constructed by merging *SUBSPELL_LOEN* from the spell on *SPELL_E*.

In the first step we allocate spells to a year. If a raw spell spans the beginning or end of a year the start date and end date are modified. The income and hours are allocated proportionally to the new spell.

In the second step we smooth the spells such that spells are not overlapping.

We remove overlaps by altering end dates of spells in order to ensure that employment spells do not overlap each other. The employment spell data includes overlapping employment spells within a month. Whenever two employment spells overlap within a month we alter the end date of the first spell such that it ends one day prior to the second employment spell. [9]

Secondly, we smooth employment spells. We identify subsequent employment spells for which an individual made a job-to-job transition or a recall. [10] For a job-to-transition with a maximum gap of 14 days between the spells we alter the end date of the first employment spell to the day prior to the start date of the second employment. For recall employment with a maximum gap of 93 days between two employment spells, we combine the two employment spells into one employment spell covering the entire period between the start date of the first employment spell to the end date of the second employment spell. [11]

We do not alter hours worked or earnings for smoothed employment spells which occur within the same year. [12] For smoothed job-to-transition spells covering two years we do not alter earnings and hours in the first year. In the second year, we set earnings (hours worked) equal to average earnings (hours worked) per day in the first year of the employment spell times the duration in days of the employment spell in the second year. [13]

We create yearly non-employment spells in all gaps between employment spells. We create a non-employment spell id named *SPELL_ID_N*. We do not allocate earn-

---

[9] There exist few cases in which one of the overlapping spells starts later than the other and ends prior to the other. E.g. consider spell A covering 1st of March to 31st of March and spell B starting 5th of March and ending 20 th of March. In this case, spell A is split into two spells, i.e. one spell from 1st of March to 4th of March and one spell from 21st of March to 31st of March, and spell B is left unchanged.

[10] Two subsequent employment spells at two employers are referred to as a job-to-job transition. Two subsequent employment spells at the same employer are referred to as a recall.

[11] 1.62% of employment spells are smoothed due to job-to-job transitions and 8.62% are smoothed due to recall employment.

[12] As start and end dates of spells is the more inaccurate information compared to yearly earnings information.

[13] e.g. consider an employment spell ending 20th of December 1991 and a subsequent employment spell starting 4th of January 1992 at two different employers. We smooth the first employment spell such that it ends 3rd of January. We set earnings and hours worked equal to three times the average earrings and hours worked of the employment spell in 1991. For the part of the smoothed employment spell in 1991 we do not alter earnings or hours worked.

ings to these non-employment spells even if it might be a U spell. These earnings are only available in EUN spell data.

Last, we rename and collapse variables compared to spell data *SPELL_E* and *SUBSPELL_LOEN*. We rename *YR_LONTIMER_ESTIMATED* to *TIMER*, *YR_LONBLB* to *BELOEB*, *START_DATO* til *JOB_START_DATO*, and *SLUT_DATO* to *JOB_SLUT_DATO*. We collapse SPELL_ID_en17 and SPELL_ID_N to one variable *SPELL_ID*.

Table 14 List of Variables - SPELL_U

Variable name	Title
PNR	*PNR, Person identifier, encrypted CPRNR number*
JOB_START_DATO	*JOB_START_DATO, In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*
JOB_SLUT_DATO	*JOB_SLUT_DATO, In SPELLS, the last day of an employment. ANS_TYPE equal to 1 or 2.*
SPELL_FIRM_ID	*SPELL_FIRM_ID, SPELL, Firm ID*
STATE	*STATE, State for a spell: U: All observations have state U.*
SPELL_ID	*SPELL_ID, SPELL, An identifier of a spell*
AAR	*AAR, Year for the reference period of the observation*
BELOEB	*BELOEB, Earnings or benefits in the spell*
TIMER	*TIMER, Hours worked estimated using Lund/Vejlin algorithm*

13 SPELL_EUN_YEARLY

Documentation will be added.

14 SPELL_EUN_YEARLY_SMOOTHED

We construct yearly EUN spell data using the following datasets

Table 15 List of Variables - SPELL_EUN_YEARLY_SMOOTHED

Dataset name	Title
SPELL_E_U_N	*SPELL_E_U_N, Spells, E-U-N spells with overlaps.*
SPELL_E	*SPELL_E, Spells, Primary jobs with overlaps.*
SUBSPELL_LOEN	*SUBSPELL_LOEN, Subspells with wage and salary for SPELL_PRIM_EMPLOY.*
RAW_UNEMPLOYMENT	*RAW_UNEMPLOYMENT, Spells, Unemployment with overlaps.*

We censor unemployment spells at 1st of January 1985.[14] We then delete unemployment spells with end date prior to start date. [15]

We merge *SPELL_E_U_N* and *SPELL_E* in order to have *SPELL_FIRM_ID* for the merged employment spells.

15 Dataset SUBSPELL_E_SEC_EMPLOY

Documentation will be added.

16 Dataset SPELL_U

The jobs contruct ˘spells.

For the period 2004-2007. For the period 2004-2007 the datasets *LS* and *SHSS* are used for constructing ˘spells.

From 2007 the datasets *ILME* and *OF* are used for constructing ˘spells.

Thus unemployment spells spanning 1st of January 2007 are constructed using both datasets. In this case, there exist two unemployment subspells, one for the *LC* period and one for the*OF* period.

The dataset contains the following variables:

Table 16 List of Variables - SUBSPELL_U

Variable name	Title
PNR	*PNR, Person identifier, encrypted CPRNR number*
JOB_START_DATO	*JOB_START_DATO, In SPELLS, the first day of an employment. ANS_TYPE equal to 1 or 2.*
BEL	BEL
SPELL_ID	*SPELL_ID, SPELL, An identifier of a spell*
STATE	*STATE, State for a spell: U: All observations have state U.*
TIMER	*TIMER, Hours worked estimated using Lund/Vejlin algorithm*

Abstract

I describe how we construct unemployment spells using registers LC, SHSS, ILME and OF for the period 1985-2013.

Important folders

The main folder of the unemployment spell data is called Unemployment spells. It is located on path G:

Data

[14] Unemployment spells in *SPELL_E_U_N* are constructed from weekly registers. Therefore, whenever the week contains 1st of January 1985 it also contains days in 1984.

[15] 0.25 % of spells are deleted. The erroneous spells occur whenever the week spans 1st of January 1985 but the end date is in 1984.

Workdata
702728
FAWC2728
test. Within the folder, datasets are kept in folder Dataset, logs are kept in Logs, and PDF output of the programmes are kept in PDF. Further, the SAS programmes are kept on path programme-version fida cvr 2016.

Introduction

We construct unemployment spells and we use income registers such that each unemployment spell contains an approximate of the amount of unemployment benefit received during the spell. The unemployment registers change in 2007 from LC and SHSS to ILME and OF which among other things improve the quality of start- and end date of the spells. The constructed unemployment spells do not contain a break in 2007 thus we are confident that we have circumvent the potential problems arising from a change in data source.

We have constructed the unemployment spells such that they contain individuals who receive benefits for which it can be assumed that the individuals are actively searching for a job, thus the unemployment spells constructed primarily contain individuals who receive cash benefit (kontanthjælp) or unemployment insurance benefit (dagpenge). On the contrary, the unemployment spells do not contain individuals who participate in active labour market programmes or individuals who are on leave as these individuals can not be assumed to be actively searching for a job. We plan to add separate active labour market programme and leave spells in a later version. The rest of the document contains individual description of the different SAS programmes used to construct unemployment spells.

Description of the individual SAS programmes

01-SHSS_v91 The purpose of this SAS programme is to extract information on the amount of benefit received for a given unemployed individual for years 1985 to 2007 using DST register SHSS. The variables of interest are all variables named 'VARM-XXXX-YY' where VARM stands for 'monthly duration', XXXX is a code identifying which type of benefit the variable contains information about, and YY identifies which month the variable contains information about. E.g., VARMMV10 = 30 refers to an individual who received pension benefits for thirty days of the month. The value of the variables is an integer in the interval [0, 30]. Variables with 'bel' as prefix contains information about yearly amount of received benefit. We have inferred the mapping between 'VARM' and 'bel' variables as we have not been able to find this in the documentation provided by DST. The mapping can be found in the following list.

Since we only use SHSS to get information about amount of received benefit, we do not need to define start and end date.

We assign the code of each 'VARM' variable, i.e. XXXX, and create a new variable 'kode_agg'. The new variable can take values 'unemployment', 'leave', 'ALMP', 'pension', and 'education'.2

The following four values of code get assigned to 'kode_agg' = 'unemployment'.

- VARM-MAN: Arbejdsløshedsdagpenge EKSL. kontanthjælpsmodtagere. Amount received: 'bel_arb'.
- VARM-MF: Underhold. Amount received from: 'bel_kon1', 'bel_kon2', 'bel_kon3', 'bel_kon194', 'bel_kon294', 'bel_kon394', 'bel_kon494', 'bel_kon598', 'bel_kon699'.
- VARM-MLYD: Ledighedsydelse (for years 2001-2005). Amount received: 'bel_lyd'.
- VARM-ML: Ledighedsydelse (for years 2006-2007). Amount received: 'bel_lyd'.

We now aggregate records with 'kode_agg' = unemployment. Multiple records exist whenever an individual changes benefit within 'kode_agg' = unemployment, e.g. an individual changing from unemployment insurance benefit (dagpenge) to cash benefit (kontanthjælp). We compute yearly amount of received benefits for a given individual with 'kode_agg' = 'unemployment'.

The resulting dataset of this programme is

_01_shss_bel. Start and end date will if necessary be defined using the following algorithm. We regard consecutive months with 'VARM-XXXX-YY' ¿ 0 as a spell. We set start date of the spell equal to the first day in the month if 'VARM' = 30. If 'VARM' ¡ 30 then we look at the next months' value of 'VARM' for the same individual on the same code. If this next months' 'VARM' ¡ 30 than we set start date equal to first day of the month, otherwise if it is equal to 30 then we set start date equal to the last day of the month minus the value of 'VARM'. E.g. 'VARM-XX-01' = 15 and 'VARM-XX-02' = 20. In this case we set start date equal to first of January as we believe that the unemployment spell is a part-time unemployment spell. E.g. 'VARM-XX-01 = 15 and 'VARM- XX-02' = 30. In this case, we believe that the unemployment spell is a full-time unemployment spell and we set start date equal to 15th of January.

We define end date in the same way, i.e. we set end date equal to the last day of the month if 'VARM' = 30. Otherwise, if 'VARM' ¡ 30 and the previous month' 'VARM' ¡ 30 then we set end date equal to the last day of the month. If the previous month' 'VARM' = 30 then we set the end date equal to the first day of the month plus 'VARM'.

In this version, we only create unemployment spells.

The analysis of the programme shows that around 1amount of received benefit ('bel_yearly') equal to zero.3 DST documents that around 2records in SHSS have amount of received benefit equal to zero due to faulty data.

02-LC_v7 Register LC contains weekly unemployment data for years 1985-2007. We use LC to construct unemployment spells for years 1985-2006. LC does not contain information about amount of received benefit. We use the information extracted from SHSS in programme 01 to get information on amount of received benefit onto the unemployment spells created using LC.

The purpose of this SAS programme is to extract data from LC and prepare the data in order to make unemployment spells.

The dataset contains to types of variables, i.e. lgradXX and larsagXX. E.g., lgrad01 is the degree of unemployment in week 1 and larsag01 is the type of unemployment in week 1. Variable lgradXX 2 [0, 1000] and lgrad = 1000 corresponds to five days of unemployment, lgrad = 800 corresponds to four days of unemployment etc. The weeks and years in LC are 'CRAM-weeks' meaning that they follow the 'CRAM-calendar'. 'CRAM-weeks' are usually, but not always, lagged by two weeks compared to the European calendar. We convert 'CRAM-weeks' to European calendar weeks using a mapping file which maps 'CRAM-weeks' to European calendar weeks.4 Variable 'larsag' can take seven different values. 76insurance benefit (dagpenge) coded as 'larsag' equal to missing.56 Another 16weekly records are cash benefit (kontanthjælp) coded as 'larsag' = 8. 'larsag' 2 4, 5, 6 are holiday benefits, and 'larsag' 2 1, 2, 3 are various special unemployment benefits agreed on with specific unions. These special agreements only account for 1.5We create variable 'kode_agg' which takes value' leave' if 'larsag' 2 4, 5, 6, otherwise 'kode_agg' takes value 'unemployment'.

The resulting dataset of this programme is

- _02_LC.
- 03-LC_v7 We now create unemployment spells using the prepared data from programme 02. We regard consecutive weeks with lgrad ¿ 0 and 'kode_agg' = 'unemployment' as a spell. We define start date and end date using the following algorithm. For each record, we define a temporary variable 'days' as 'days' = ceil('lgrad' 200). If 'lgrad' = 1000 then we set start date of the spell equal to the first day of the week. If 'lgrad' ¡ 1000 then we look at the next weeks' value of 'lgrad' for the same individual. If the next months' 'lgrad' ¡ 1000 than we set start date equal to first day of the week, otherwise if it is equal to 1000 then we set start date equal to the first day of the week incremented by 5− 'days'. E.g. consider records where 'lgrad' = 600 and next month's 'lgrad' = 1000. In this case we set start date equal to Monday of the week incremented by 5 - 3 = 2 days resulting in the start date of the spell equal to Wednesday of the week. We define end date in the same way, i.e. if 'lgrad' = 1000 then we set end date equal the last day of the week. Otherwise, if 'lgrad' ¡ 1000 and the previous week's 'lgrad' ¡ 1000 then we set
- p. 6-7 of the PDF.
- The mapping file is called CRAM_kalender_1985_2007 and it is located in the programme folder.
- p. 3 of the PDF.

Missing refers to character value ', however in 2007 missing is coded as '.'. end date equal to the last day of the week. If the previous week's 'lgrad' = 1000 then we set end date of the spell equal to the first day of the week incremented by days - 1.7 For each weekly record, we compute 'number of hours' unemployed. These number of hours are only going to be used when we merge employment and unemployment spells together and we find an overlap between these. In that case, we will compare number of hours unemployed and number of hours worked in order to define which state, employment or unemployment, that is chosen to be the primary state. We compute number of hours unemployed as 37 ' 'lgrad' 1000 .

Having computed start and end date, we compress consecutive weeks of unemployment into spells. We set start date and end date of the spell equal to the computed dates. We aggregate 'number of hours' unemployed. We create a boolean variable 'bool_multiple_larsag' which indicates whether the unemployment spell has more than one value of 'larsag', e.g. an unemployment spell where the individual changes from unemployment insurance benefit to social assistance sometime during the unemployment spell. We construct subspells for these cases, i.e. we save the weekly records in a subspell dataset for all unemployment spells where 'bool_multiple_larsag' = 1. We index each LC unemployment spell by 'lc_spell_id' and we index each subspell by 'lc_subspell_id'.

The resulting datasets of this programme are

- _03_LC_spell_without_bel,
- _03_LC_subspells.

The analysis of the programme shows time series plots of the stock of individuals in each week in each different 'larsag'.

04-LC_v5

Dataset SHSS contains information on the yearly amount of received unemployment benefits. We merge this information from programme 01 onto each computed LC unemployment spell from programme 03. For each unemployment spells and for each year the unemployment spell span, we compute the duration in days of the unemployment spell and we compare this to the total duration in days of all unemployment spells for the given individual in the given year. Let 'day_dur' equal the duration in days of the unemployment spell and 'total_day_dur' equal the total duration in days of all unemployment spells. We now assign the amount equal to day_dur total_day_dur ' bel_yearly to each unemployment spell.

The resulting dataset of this programme is

_04_LC_spells_with_bel. The analysis of the programme shows that around 1records have amount of received benefit equal to missing. These missing values are due to no match between the constructed LC unemployment spells and the yearly amount of received

As a special case only relevant for end date computation, consider an record with 'lgrad' 2 [900, 1000) implying that 'days' = 5, thus if the previous week's 'lgrad' = 1000 then end date of the spell will equal Monday of the week incremented by 5 - 1 days, i.e. the end date will be Friday of the week. In order not to have gaps in the spell data and in order to be consistent with 'lgrad' = 1000, we set end date equal to the last day of the week instead of Friday of the week.

benefits from SHSS.8 Further, the analysis shows distributions of amount of received benefits weighted by either number of weeks in the unemployment spell or number of weeks and average 'lgrad'.

05-ILME_v9 The purpose of this SAS programme is to extract information on the amount of benefit received for a given unemployed individual for years 2008 to 2013 using DST register ILME. The variables of interest are 'vmo_startdato', 'vmo_slutdato', 'vmo_indkomst_type_kode', and all the variables which describe the amount of benefits received. The first two variables contain information about start date and end date, and 'vmo_indkomst_type_kode' contain information about which kind of benefit the individual received. The reason why we do not use ILME as the source for constructing unemployment spells is that the variable 'vmo_indkomst_type_kode' does not deliver as detailed information as we can find in dataset OF.

We find that variables containing information about the amount of received benefits can be negative. However, if we aggregate the amounts for a consecutive period of received benefits the aggregated amounts are almost always non-negative. We conclude that the negative amounts are corrections. Therefore, we aggregate the amounts (negative and positive) when constructing ILME spells. An ILME spell is constructed as a consecutive period of received benefits on the same 'vmo_indkomst_type_kode'.

ILME spells with 'vmo_indkomst_type_kode' equal to '04', '06', or '24' are regarded as potential unemployment spells, thus these spells will be merged onto unemployment spells constructed using OF for 2008 to 2013. We use the term potential unemployment spells since 'vmo_indkomst_type_kode' doe not uniquely determine whether the ILME spell originates from a spell of unemployment, pension, leave, etc. E.g. 'vmo_indkomst_type_kode' = 04 is used for unemployment insurance payments, pension payments, flex job payments, etc.

We truncate the created ILME spells at 31. of December 2013. We reduce all amount variables by the duration in days in the period 2008-2013 compared to the total duration of the ILME spell.

The resulting dataset of this programme is

_05_ILME_spells. The analysis of the programme shows the duration of ILME spells across the different values of 'vmo_indkomst_type_kode'. Further, the analysis delivers a count of spells for each value of 'vmo_indkomst_type_kode'.

06-OF_v92 The purpose of the OF SAS programmes is to make unemployment spells using dataset OF. OF consists of almost all benefits for individuals aged between 16-64. We extract information for the period 2007-2013.

The purpose of this SAS programme is to extract data from OF and make unemployment spells. In the next programme, we add amount of received benefit to the constructed unemployment spells. The variables of interest are 'pti_vfra', 'pti_vtil', 'pti_timer_per_uge', and 'pti_tilstand_kode'. The variables contain information about start date and end date, the amount of hours per week, and the kind of benefit received. 8DST documents that around 2data.

We create variable 'kode_agg' which aggregates the different values of 'pti_tilstand_kode' into unemployment, active labor market programmes, pension, and leave.9 A full list of the assignment of the values of 'pti_tilstand_kode' to 'kode_agg' can be found in the appendix. We truncate the OF records if end date exceeds 31. of December 2013. We compute the total number of hours for each record as 'days' ' 'pti_timer_per_uge' where 'days' is the truncated duration measured in days between start date and end date. We create OF unemployment spells by compressing consecutive records with 'kode_agg' = 'unemployment'. The existence of consecutive records on 'kode_agg' is due to either a change of 'pti_tilstand_kode' (within 'kode_agg' = 'unemployment') or a change of 'pti_timer_per_uge' (within same value of 'pti_tilstand_kode').

We set number of hours for the compressed record equal to the sum of hours for the consecutive records.

We create OF subspells on 'pti_tilstand_kode' and we create a boolean variable 'bool_of_subspell' which indicates whether the OF unemployment spell has multiple values of 'pti_tilstand_kode'.

We index OF unemployment spells by 'of_spell_id' and subspells by 'of_subspell_id'. We create a dataset containing the mapping between 'of_spell_id' and 'of_subspell_id'.

The resulting datasets of this programme are

- _06_OF_spells_without_bel,
- _06_OF_subspells,
- _06_OF_subspells_mapping.

The analysis of this programme shows the distribution of average weekly hours 'spent' in unemployment. 07-OF_before_2008_2021_v.1

The purpose of the SAS programme is to merge information about the amount of benefits received onto each constructed OF unemployment spell. OF unemployment spells span the period 2007- 2013. ILME span the period 2008-2013, thus we also merge the unemployment spells with SHSS from programme 01 in order to get amount of benefits for 2007.

We merge variables 'vmo_a_indk_am_bidrag_fri' and 'vmo_b_indk_am_bidrag_fri' from ILME dataset constructed in programme 05. We compute the number of days the ILME spell overlaps the OF unemployment spell and we assign the fraction of benefits which corresponds to the fraction between the number of overlaying days compared to the total duration of the ILME spell. E.g., consider an OF spell spanning the period of 1. of January to 25. of January and an ILME spell spanning the period 10. of January to the 30. of January with benefits equal to 8.000 DKK. The number of overlaying days is sixteen and the total duration of the ILME spell is 21. In this case we assign 16 21 ' 8.000 DKK to the OF unemployment spell.

We merge yearly amount of received benefits from SHSS for year 2007 onto the OF unemployment spells. As for LC unemployment spells, we split the yearly amount of received benefits from SHSS onto the OF unemployment spells by the fraction between duration in days in 2007 of each OF unemployment spell and the total 2007 duration of all OF unemployment spells for a given individual.

The resulting dataset of this programme is

We plan to add active labor market programmes, pension, and leave spells in a later version. 10We keep all 'pti_tilstand_kode' spells as subspells, i.e. for OF unemployment spell for which 'pti_tilstand_kode' changes and for OF unemployment spells for which it does not change. In this way, it is possible to merge 'pti_tilstand_kode' back onto the OF unemployment spells and e.g. to distinguish between individuals receiving unemployment insurance benefit and individuals who receive cash benefit.

_07_OF_spells_with_bel. The analysis of this programme shows the distribution of variable 'overlap_grad_of' which measures the percentage of days of the OF unemployment spell that we have been able to overlay with ILME spells. We find that 90ILME spells.11 Further, boolean variable 'bool_shss_merge' indicates whether we have been able to merge an SHSS spell onto OF unemployment spells in 2007. We see that for 99OF unemployment spells in 2007 we are able to match the spell with a SHSS spell.12 Note that when merging OF and SHSS spells, we only merge on ('pnr', 'year') which is a weaker condition than when we merge OF and ILME spells where we merge on ('pnr', 'date'), thus the higher overlap degree was expected.

08-Spells_full_period_v1 The purpose of the SAS programme is to combine the LC and OF unemployment spells in order to construct unified unemployment spells for the spell period 1985-2013. We input LC unemployment spells covering period 1985-2006 from programme 04 and we input OF unemployment spells covering period 2007-2013 from programme 07. We compress LC and OF unemployment spells for which LC end date is equal to 31. of December 2006 and OF start date is equal to 1. of January 2007. We index the resulting unemployment spells by 'spell_id'. We create subspells for resulting unemployment spells which originates from both a LC and an OF unemployment spell, and boolean variable 'bool_subspell_change_of_source' indicates this event.

We keep a mapping dataset which maps ('pnr', 'spell_id') into ('pnr', 'lc_spell_id', 'of_spell_id') such that it is possible to find corresponding LC and OF unemployment spells for all resulting unemployment spells.

The resulting datasets of this programme are

- _08_U_spells,
- _08_U_subspells,
- _08_U_mapping.

The analysis of this programme consist of a time series plot and a count of the stock of individuals in the constructed unemployment spells at the first day of every month between 1985-2013. 11p. 6 and 8 of the PDF. 12p. 18 of the PDF.

Appendix

List of all values of variable 'pti_tilstand_kode' with corresponding value of kode_agg

Unemployment

- 5010: Ledige, detaljer uoplyst
- 5020: Ledige dagpengemodtagere
- 5030: G dage
- 5035: Arbejdsmarkedsydelse
- 5080: Ledige kontanthjælpsmodtagere
- 5090: Ledighedsydelse, detaljer uoplyst
- 5100: Ledighedsydelse mellem fleksjob
- 5110: Ledighedsydelse i visitionsperioden før første fleksjob
- 5140: Ledighedsydelse efter ustøttet beskæftigelse
- 7075: Ledighedsydelse SS
- 7090: Introduktionsydelse

Pension

- 6010: Efterløn

- 6020: Overgangsydelse
- 6030: Fleksydelse
- 6040: Delefterløn
- 7020: Førtidspension SS
- 7025: Førtidspension

Leave

- 3110: Orlov til uddannelse
- 3120: Orlov til sabbat
- 3130: Orlov til børnepasning
- 4030: Sygedagpenge, detaljer uoplyst
- 5040: Feriedagpenge fra ledighed
- 5050: Feriedagpenge fra beskæftigelse
- 5060: Feriedagpenge o. a. ledhedsårsag
- 5070: Feriedagpenge, uoplyst
- 5120: Ledighedsydelse under ferie
- 5130: Ledighedsydelse under sygdom og barsel
- 5800: Syge på dagpenge eller arbejdsmarkedsydelse
- 6050: Nedsatte dagpenge (par. 32)
- 7030: Sygedagpenge SS
- 7035: Sygedagpenge, beskæftigede
- 7036: Sygedagpenge, ej beskæftigede
- 7037: Sygedagpenge, detaljer uoplyst
- 7040: Barselsdagpenge, SS
- 7045: Barselsdagpenge, beskæftigede
- 7046: Barselsdagpenge, ej beskæftigede
- 7047: Barselsdagpenge, detaljer uoplyst

Aktivering

- 1010: Jobtræning
- 1020: Individual jobtræning
- 1030: Ansættelse med løntilskud
- 1039: Jobrotation
- 1040: Virksomhedspraktik
- 1049: Nytteindsats
- 1050: Fleksjob
- 1055: Fleksjob, KMD-aktiv
- 1060: Fleksjob, selvstændige
- 1070: Skånejob
- 1075: Skånejob, KMD-aktiv
- 1080: Servicejob
- 1090: Arbejdspraktik
- 1100: Puljejob
- 1110: Rotationsansættelse
- 1120: Voksenlærlinge

- 1510: Opkvalificering iflg. Integrationsloven
- 1520: Ordinær uddannelse
- 1540: Voksen- og efteruddannelse
- 1550: Korte vejlednings- og afklaringsforløb
- 1560: S̊ligt tilrettelagte projekter og uddannelsesforløb
- 1562: Særligt tilrettelagte uddannelsesforløb
- 1564: Særligt tilrettelagte projekter
- 1570: Vejledning og introduktion
- 1580: Særligt aktiverende forløb
- 1590: Intensiv jobsøgning
- 1591: Vejl opkval ordinær udd.
- 1592: Vejl opkval øvrige forløb under 4 uger
- 1593: Vejl opkval øvrige forløb over 4 uger
- 1594: Vejl opkval øvrige forløb
- 1595: Mentorstøtte
- 1599: Vejledning og opkvalificering
- 1600: Særlig formidling
- 1800: Selvvalgt uddannelse i 6 uger
- 2010: Kursus i samfundsforståelse
- 2020: Danskundervisning
- 2030: Særligt tilrettel. danskunderv.
- 2510: Aktivering, detaljer uoplyst
- 2512: Aktiveringsgodtgørelse
- 2514: Forsørgelse under vejledning, opkvalificering og virksomhedspraktik
- 2516: Løntilskud til personer i tilbud efter kapitel 12
- 2518: Kontanthjælp og starthjælp under forrevalidering
- 2520: Forsøg
- 2530: Frivillige ulønnede aktiviteter
- 2540: Uddannelse m. voksenudd.støtte
- 3010: Etableringsydelse
- 3020: Igangsætningsydelse
- 4010: Revalidering, detaljer uoplyst
- 4012: Revalideringsydelse med virksomhedspraktik
- 4014: Løntilskud ved revalidenters ansættese med løntilskud
- 4020: Forrevalidering, detaljer uoplyst
- 7005: Jobafklaringsforløb
- 7010: Socialtilstand detaljer uoplyst
- 7015: Førtidspension, gammel ordning
- 7024: Ressourceforløb
- 7050: Kontanthjælp (passiv)
- 7055: Kontanthjælp SS
- 7060: Revalidering SS
- 7070: Forrevalidering SS
- 7080: Støtte til handicappede

We construct unemployment spells using DST registers SHSS, LC, ILME, and OF. We use income registers (SHSS, ILME) to approximate the amount of unemployment benefit received during the unemployment spell.

The unemployment spell dataset contains the following variables

- pnr, person identifier
- startdato, start date of the unemployment spell
- slutdato, end date of the unemployment spell
- bel, amount of received unemployment benefit during the unemployment spell.
- timer, amount of hours spent in the unemployment spell.
- state, takes value 'U', 'E', or 'N', i.e. unemployment, employment or residual spell state 'N'. In dataset U_spells, state = 'U' for all records.
- spell_id, identifier of the unemployment spell. The combination (pnr, spell_id) uniquely identifies an unemployment spell.
- bool_subspell_change_of_source, boolean variable which indicates whether the unemployment spell is constructed using more than one DST registers.

17 SUBSPELL_LOEN

Name of dataset: subspell_loen

General information

The dataset consists of subspells containing yearly salary and yearly number of hours worked for a given year in the duration of the primary employment. We create such subspells for all years in the duration of the primary employment spell. The combination of pnr and person spell id tells which primary employment spell that each record maps to.

Variables in dataset

- Personnummer, person id variable.
- Start date for spell, contains start date of the subspell. It might be the case that the duration of the subspell is less than an year which occurs when the duration of the primary employment spell is less than an year.
- End date for spell, contains end date of subspell.
- Total yearly hours worked, contains yearly number of hours worked for the given year of the primary employment spell. We have estimated number of hours worked using ATPdata for the period 1985-2007. We have directly used the variable containing number of hours worked for the period 2008-2011 which we get from dataset BFL.
- Total yearly salary, contains yearly salary for the given year of the primary employment spell.
- Year, contains the year of the subspell.
- Person spell id, contains an identification key which maps the subspell to the primary employment spell from which the subspell was created.

18 SUBSPELL_MULT_DSKOD

Name of dataset: subspell_mult_dskod

General information

The dataset contains all 'multiple dskod' subspells. We create these subspells since we observe that some persons are employed at the same firm id in a continuous period while variable 'dskod'/'bfl_dskod' takes on different values. In this case we choose a single value of 'dskod'/'bfl_dskod' for the whole primary employment spell which corresponds to the 'dskod'/' bfl_dskod' of the period of the spell in which the person worked the largest amount of hours (or the where the largest salary was obtained if two different values of 'dskod'/'bfl_dskod' have the same amount of hours worked).

We save all the monthly records of the primary employment spell as single 'multiple dskod' subspells so that it is easily seen where the values of 'dskod'/'bfl_dskod' changes within the primary employment spell.

Variables in dataset

- Personnummer, person id variable.
- Firm id, firm id variable. We use 'senr' from 1985-2004 and 2006-2007 and we use 'cvrnr' for period 2005 and 2008-2011 which is due to the fact that these variables are the ones that DST uses in Cons, Ras and BFL for the given periods. We will later make an time consistent firm identification variable probably using information from KOB. We observe a substantial amount of job changes in 2005 and 2008 which we think is due to the change in firm id variable. We expect the time-consistent firm id variable to solve this issue.
- Start date for spell, contains start date for the subspell.
- End date for spell, contains end date for the subspell.
- Dskod, contains actual 'dskod'. Actual in the sense that we in these subspells list the observed 'dskod', whereas we have chosen a single value of 'dskod' for primary employment spells.
- Arbgnr8, contains actual 'arbgnr8'.
- bfl_dskod, contains actual 'ajo_prod_nr_fra_prod_job'.
- bfl_arbgnr8 contains actual 'ajo_se_nr_fra_prod_job'.
- Hours worked, contains number of hours worked. We have estimated number of hours worked using ATP-data for the period 1985-2007. We have directly used the variable containing number of hours worked for the period 2008-2011 which we get from dataset BFL.

Documentation Spell dataset - all_mult_dskod_subspell

- Salary, contains salary.
- Year, contains the year of the spell.
- Person spell id, identification key mapping the subspell to the matching primary employment spell.