ADVANCED ANALYSIS

OF

LINKED HEALTH DATA

Principles and

Hands-On Applications

A Short-Course with
Professor David B. Preen

SYNTAX SOLUTIONS

TO R TRAINING EXERCISES

R Solutions by Dr Pete Arnold and Dr Joanne Demmler

Version 3.1 March 2021

TRAINING SESSION 1: BASE-R SYNTAX SOLUTIONS

```
################################################################################
# AALHD DAY 1
# R Syntax solutions
#
# Dr Pete Arnold based on code by Dr Joanne Demmler
# Updated March 2021
################################################################################
# Hints about what has been used in this solution.
#
# Step 3 uses data.table to create the sequence variables. A data.table is a
#        high performance extension of a data frame. The tidyverse is very
#        good at this kind of thing as well.
#
################################################################################

# If you do not have the packages below installed, you should install using the
# install.packages command: e.g. install.packages('data.table') or clicking the
# link sometimes provided at the top of this code in R-Studio.
library(data.table)
# The chron library helps with the dates etc. at step 20.
library(chron)
# Survival library used at step 21.
library(survival)

################################## Exercise 1 ##################################

# Create a project with the code in a directory e.g. 'R' or 'source' and data in
# a directory, 'data'. Copy all the data files to 'data'.
# You should now be able to work with the project working directory.

# Preparation.
# We will be combining a series of tables by appending the rows from each table
# to the first. Since this depends on each table having exactly the same columns
# in the same order, we will define that once, here, to try to minimise the risk
# of errors.
cutdown_columns <- c('rootlpno', 'date', 'sex', 'age', 'sequence', 'type')

# Step 1: Open the dataset.
load("data/MBSdata.RData")

# Order the data as described, if necessary.
# MBSdata <- MBSdata[order(MBSdata$rootlpno, MBSdata$servdate),]

# OPTION:
# you can also navigate to the file in the files pane in R Studio and double
# click to open.

# Check the data.
head(MBSdata, 10)

# Some over-the-top checking.
MBSdata_rows <- nrow(MBSdata)
```

```
if (MBSdata_rows == 1963465) {
    cat('Number of rows', MBSdata_rows, 'is as expected.\n')
} else {
    cat('WARNING: unexpected number of rows in MBSdata (', MBSdata_rows, ')!\n')
}


# Step 2: Tag diabetes related items.
# Create a new column and initialise all to 0.
MBSdata$glyhb <- 0

# Recode `glyhb` to 1 for any entry matching a value in the table on page TS1-3.
diabetes_mbs_items <- c(66551, 66554, 66557, 73815, 73840, 66319, 66322)
MBSdata$glyhb[MBSdata$mbsitem %in% diabetes_mbs_items] <- 1


# Step 3: Create the sequence variables.
# fileseq is a count from 1 to nrows for all rows in the correct order.
# morbseq is a count from 1 to nrows for all rows for a patient.
# servseq is a count from 1 to nrows for all the diabetes flagged rows for a patient.
MBSdata$fileseq <- seq(1, nrow(MBSdata))

# Convert to a data.table to assign values according to a key and the ordering
# already set-up.
temp_DT <- data.table(MBSdata)

# Set `rootlpno` as a primary key.
# setkeyv(temp_DT, "rootlpno")

# Create a new column, morbseq, with a value of 1 to .N as the count for each
# rootlpno.
# The square brackets have [<matching condition (none means everything)>,
#                           <order by / what to do if there is a match>,
#                           <update by grouping column(s)>].
# .N is the number of rows in a group in a data.table.
temp_DT <- temp_DT[, morbseq := 1:.N, by=rootlpno]

# Where `glyhb` is not 0 then count the number of rows for matching rootlpno's.
temp_DT <- temp_DT[glyhb != 0, servseq := 1:.N, by=rootlpno]
MBSdata2 <- as.data.frame(temp_DT)
rm(temp_DT)

# View the data as shown in the workbook.
View(MBSdata2[704:729, ])
```

R_Solutions - RStudio Source Editor

MBSdata2[704:729,]

Filter

	rootlpno	mbsitem	servdate	sex	age	postcode	glyhb	fileseq	morbseq	servseq
704	4	116	1999-12-01	1	76	6026	0	704	174	NA
705	5	66209	1990-02-18	1	67	6156	0	705	1	NA
706	5	69207	1990-03-30	1	67	6156	0	706	2	NA
707	5	66211	1990-05-13	1	67	6156	0	707	3	NA
708	5	57712	1991-10-19	1	68	6156	0	708	4	NA
709	5	57712	1991-10-19	1	68	6156	0	709	5	NA
710	5	136	1992-01-15	1	69	6156	0	710	6	NA
711	5	66203	1992-01-15	1	69	6156	0	711	7	NA
712	5	66291	1992-01-15	1	69	6156	0	712	8	NA
713	5	66303	1992-01-15	1	69	6156	0	713	9	NA
714	5	65007	1992-04-17	1	69	6156	0	714	10	NA
715	5	66211	1992-04-17	1	69	6156	0	715	11	NA
716	5	66551	1992-04-17	1	69	6156	1	716	12	1
717	5	73907	1992-04-17	1	69	6156	0	717	13	NA
718	5	66596	1992-04-25	1	69	6156	0	718	14	NA
719	5	66602	1992-04-25	1	69	6156	0	719	15	NA
720	5	73907	1992-04-25	1	69	6156	0	720	16	NA
721	5	66211	1992-07-30	1	69	6156	0	721	17	NA
722	5	66291	1992-07-30	1	69	6156	0	722	18	NA
723	5	73917	1992-07-30	1	69	6156	0	723	19	NA
724	5	66211	1992-12-03	1	69	6156	0	724	20	NA
725	5	66551	1992-12-03	1	69	6156	1	725	21	2
726	5	73917	1992-12-03	1	69	6156	0	726	22	NA
727	5	65007	1993-02-26	1	70	6156	0	727	23	NA
728	5	66211	1993-02-26	1	70	6156	0	728	24	NA
729	5	66551	1993-02-26	1	70	6156	1	729	25	3

Showing 1 to 26 of 26 entries, 10 total columns

```
# Step 4: Save the data.
save(MBSdata2, file='data/MBSdata2.RData')

# Rename the columns.
# Note that it is possible to combine the following lines into a more compact
# form if you wanted to e.g. by cutting down and adding the new column in the
# same line of code.
names(MBSdata2)[c(3, 10)] <- c('date', 'sequence')

# Add a type column, set to 1 for all rows.
MBSdata2 <- data.frame(MBSdata2, type = 1)

# Cutdown the file; the first part defines the rows (i.e. those with a sequence)
# and the second defines the columns.
```

```
MBScutdown <- data.frame(MBSdata2[!is.na(MBSdata2$sequence), cutdown_columns])

# Check that the numbers match the expected values (29,571 records with 7,239
# with sequence set to 1).
nrow(MBScutdown)
nrow(MBScutdown[MBScutdown$sequence == 1, ])

# Save the data.
save(MBScutdown, file='data/MBScutdown.RData')


# Step 5: Open the PBS data.
load('data/PBSdata.RData')

# Check the data.
head(PBSdata, 10)

# There should be 1546335 rows.
nrow(PBSdata)

# Order the data as described, if necessary.
# PBSdata <- PBSdata[order(PBSdata$rootlpno, PBSdata$dispdate),]


# Step 6: Find and mark diabetes codes.
diabetes_pbs_items_oral <- c(1202, 1801, 2178, 2430, 2440, 2449, 2607, 2720,
                             2939, 2940, 8188, 8189, 8391, 8392, 8450, 8451,
                             8452, 8533, 8535, 8607, 8687, 8688, 8689, 8690,
                             8691, 8692, 8693, 8694, 8695, 8696, 8810, 8811)
diabetes_pbs_items_insn <- c(1425, 1426, 1429, 1430, 1431, 1461, 1462, 1531,
                             1532, 1533, 1534, 1535, 1537, 1591, 1592, 1710,
                             1711, 1713, 1715, 1716, 1718, 1721, 1722, 1761,
                             1762, 1763, 2061, 2062, 8006, 8084, 8085, 8212,
                             8390, 8435, 8571, 8609)
# Create and initialise the column to zero (no medication).
PBSdata$diabmed <- 0
# Code diabmed to 1 if pbsitem matches anything in the oral list.
PBSdata$diabmed[PBSdata$pbsitem %in% diabetes_pbs_items_oral] <- 1
# code diabmed to 2 if pbsitem matches anything in the insulin list.
PBSdata$diabmed[PBSdata$pbsitem %in% diabetes_pbs_items_insn] <- 2

# Step 7: Create various sequence variables.
# Fileseq: just count the rows from 1 to the total number.
PBSdata$fileseq <- seq(1, nrow(PBSdata))

# Morbseq: the same as fileseq but just for each rootlpno. Again, we'll use a
# data.table (see above for details).
temp_DT <- data.table(PBSdata)
# setkeyv(temp_DT, "rootlpno")
temp_DT <- temp_DT[, morbseq := 1:.N, by=rootlpno]
# Where diabmed is not 0, mark each row with a count within each rootlpno.
temp_DT <- temp_DT[diabmed != 0, dispseq := 1:.N, by=rootlpno]

PBSdata2 <- as.data.frame(temp_DT)
rm(temp_DT)
```

```
# View the data as shown in the workbook.
View(PBSdata2[51:78, ])
```

R_Solutions - RStudio Source Editor

PBSdata2[51:78,]

Filter

	rootlpno	pbsitem	scripts	dispdate	sex	age	postcode	diabmed	fileseq	morbseq	dispseq
51	1	2417	1	1999-12-20	1	86	6010	0	51	51	NA
52	2	2367	1	1998-02-24	2	61	6108	0	52	1	NA
53	2	2449	1	1998-02-28	2	61	6108	1	53	2	1
54	2	1279	1	1998-06-23	2	61	6108	0	54	3	NA
55	2	2367	1	1998-06-23	2	61	6108	0	55	4	NA
56	2	2449	1	1998-06-23	2	61	6108	1	56	5	2
57	2	1279	1	1998-08-07	2	61	6108	0	57	6	NA
58	2	2367	1	1998-08-25	2	61	6108	0	58	7	NA
59	2	1279	1	1998-09-10	2	61	6108	0	59	8	NA
60	2	2367	1	1998-09-27	2	61	6108	0	60	9	NA
61	2	2367	1	1998-10-27	2	61	6108	0	61	10	NA
62	2	1279	1	1998-11-16	2	61	6108	0	62	11	NA
63	2	2367	1	1998-11-16	2	61	6108	0	63	12	NA
64	2	1279	1	1998-12-16	2	61	6108	0	64	13	NA
65	2	2367	1	1998-12-16	2	61	6108	0	65	14	NA
66	2	2449	1	1998-12-16	2	61	6108	1	66	15	3
67	2	1279	1	1999-01-17	2	62	6107	0	67	16	NA
68	2	2367	1	1999-01-17	2	62	6107	0	68	17	NA
69	2	3050	1	1999-01-26	2	62	6107	0	69	18	NA
70	2	1279	1	1999-02-14	2	62	6107	0	70	19	NA
71	2	2367	1	1999-02-14	2	62	6107	0	71	20	NA
72	2	2449	1	1999-02-14	2	62	6107	1	72	21	4
73	2	3050	1	1999-02-14	2	62	6107	0	73	22	NA
74	2	8213	1	1999-02-14	2	62	6107	0	74	23	NA
75	2	1279	1	1999-03-05	2	62	6107	0	75	24	NA
76	2	2367	1	1999-03-05	2	62	6107	0	76	25	NA
77	2	8213	1	1999-03-05	2	62	6107	0	77	26	NA
78	2	2449	1	1999-03-16	2	62	6107	1	78	27	5

Showing 1 to 28 of 28 entries, 11 total columns

```
# Step 8: Save the data.
save(PBSdata2, file='data/PBSdata2.RData')

# Rename the variables.
names(PBSdata2)[c(4, 11)] <- c('date', 'sequence')

# Add a type column, set to 2 for all rows.
PBSdata2 <- data.frame(PBSdata2, type = 2)

# Cutdown the file with just the useful columns.
```

```
PBScutdown <- data.frame(PBSdata2[!is.na(PBSdata2$sequence), cutdown_columns])

# Check that the numbers match the expected values (174,131 records with 6,820
# with sequence set to 1).
nrow(PBScutdown)
nrow(PBScutdown[PBScutdown$sequence == 1, ])

# Save the data.
save(PBScutdown, file='data/PBScutdown.RData')


# Step 9: Open the HMDS data.
load('data/HMDSdata.RData')

# Check the data.
head(HMDSdata, 10)

# There should be 62,356 rows.
nrow(HMDSdata)


# Step 10: Find and tag the rows with diabetes codes (in multiple columns).
diabetes_hmds_items <- c(format(seq(250.00, 250.99, 0.01), nsmall=2),
                         'V77.1',
                         paste('E', format(seq(10.00, 14.99, 0.01), width=4), sep=''))

# This time, we need to look through multiple columns: diag1-diag21 (columns 9:29)
# and build up a list of all the rows that have a match.
HMDSdata$diabetes <- 0
rows_with_diabetes <- 0
for (i in 9:29){
    rows_with_diabetes <-
        append(rows_with_diabetes, which(HMDSdata[, i] %in% diabetes_hmds_items))
}

# For the matching rows set diabetes to 1.
HMDSdata$diabetes[rows_with_diabetes] <- 1


# Step 11: Create various sequence variables.
# Fileseq: just count the rows from 1 to the total number.
HMDSdata$fileseq <- seq(1, nrow(HMDSdata))

# Morbseq: the same as fileseq but just for each rootlpno. Again, we'll use a
# data.table (see above for details).
temp_DT <- data.table(HMDSdata)
# setkeyv(temp_DT, "rootlpno")
temp_DT <- temp_DT[, morbseq := 1:.N, by=rootlpno]

temp_DT <- temp_DT[diabetes != 0, condseq := 1:.N, by=rootlpno]
HMDSdata2 <- as.data.frame(temp_DT)
rm(temp_DT)

# View the data as shown in the workbook.
View(HMDSdata2[1:21,40:48])
```

R_Solutions - RStudio Source Editor

HMDSdata2[1:21, 40:48]

Filter

	proc11	ecode1	ecode3	ecode4	ecode5	diabetes	fileseq	morbseq	condseq
1	NA	NA	NA	NA	NA	0	1	1	NA
2	NA	NA	NA	NA	NA	1	2	2	1
3	NA	NA	NA	NA	NA	1	3	3	2
4	NA	NA	NA	NA	NA	1	4	4	3
5	NA	NA	NA	NA	NA	1	5	5	4
6	NA	NA	NA	NA	NA	1	6	6	5
7	NA	NA	NA	NA	NA	1	7	7	6
8	NA	NA	NA	NA	NA	0	8	1	NA
9	NA	NA	NA	NA	NA	1	9	2	1
10	NA	NA	NA	NA	NA	1	10	3	2
11	NA	NA	NA	NA	NA	1	11	4	3
12	NA	NA	NA	NA	NA	1	12	5	4
13	NA	NA	NA	NA	NA	1	13	6	5
14	NA	NA	NA	NA	NA	0	14	1	NA
15	NA	NA	NA	NA	NA	1	15	1	1
16	NA	NA	NA	NA	NA	1	16	2	2
17	NA	NA	NA	NA	NA	1	17	3	3
18	NA	NA	NA	NA	NA	1	18	4	4
19	NA	NA	NA	NA	NA	0	19	5	NA
20	NA	NA	NA	NA	NA	1	20	6	5
21	NA	NA	NA	NA	NA	1	21	7	6

Showing 1 to 21 of 21 entries, 9 total columns

```
# Step 12: Save the data.
save(HMDSdata2, file='data/HMDSdata2.RData')

# Rename variables.
names(HMDSdata2)[c(5, 48)] <- c('date', 'sequence')

# Add a type column, set to 3 for all rows.
HMDSdata2 <- data.frame(HMDSdata2, type = 3)

# Cutdown the file.
HMDScutdown <- data.frame(HMDSdata2[!is.na(HMDSdata2$sequence), cutdown_columns])

# Check that the numbers match the expected values (21,274 records with 5,328
# with sequence set to 1).
nrow(HMDScutdown)
nrow(HMDScutdown[HMDScutdown$sequence == 1, ])
```

```
# Save data.
save(HMDScutdown, file='data/HMDScutdown.RData')


# Step 13: Open the death data.
load('data/Dthdata.RData')

# Order the data as described, if necessary.
# Dthdata <- Dthdata[order(Dthdata$rootlpno),]

# Step 14: Tag diabetes related rows.
diabetes_dth_items <- c(seq(2500, 2509), 'V771', paste('E', seq(100, 149), sep=''))

Dthdata$diabetes <- 0
Dthdata$diabetes[Dthdata$codcode %in% diabetes_dth_items] <- 1

# View the data as shown in the workbook.
View(Dthdata[1:25,])
```

R_Solutions - RStudio Source Editor

Dthdata[1:25,]

Filter

	rootlpno	dthdate	codcode	sex	age	diabetes
1	1	1999-05-26	2500	1	86	1
2	3	1999-05-06	431	1	81	0
3	5	1996-04-09	2500	1	73	1
4	7	1995-08-17	2639	2	62	0
5	14	1991-09-11	4109	2	79	0
6	22	1997-11-03	5789	2	69	0
7	34	1996-01-18	2500	2	88	1
8	35	1995-10-11	4369	1	87	0
9	37	1998-04-11	2507	2	85	1
10	48	1995-09-14	4260	2	78	0
11	52	1994-09-05	4331	1	56	0
12	57	1995-02-14	2500	2	81	1
13	59	1999-01-15	2500	1	76	1
14	78	1996-10-21	4369	2	84	0
15	85	1997-02-15	2500	2	85	1
16	89	1992-08-06	2500	1	49	1
17	92	1995-05-03	1579	1	83	0
18	97	1996-01-17	4109	2	73	0
19	101	1993-08-01	1629	1	61	0
20	103	1990-04-11	2500	2	83	1
21	116	1999-02-05	2507	1	76	1
22	127	1995-03-01	2500	2	68	1
23	131	1997-12-07	1629	1	84	0
24	143	1994-06-14	2500	1	91	1
25	158	1996-06-02	2762	2	73	0

Showing 1 to 25 of 25 entries, 6 total columns

```
# Save the data (need to rename for compatibility with the workbook).
Dthdata2 <- Dthdata
rm(Dthdata)
save(Dthdata2, file='data/Dthdata2.RData')

# Step 16: rename columns, add a type, cutdown the table.

# Rename the variables.
names(Dthdata2)[2] <- 'date'

# Add the sequence and type variables.
Dthdata2$sequence <- 1
Dthdata2$type <- 4

# Cutdown to include the diabetes rows and the useful columns.
Dthcutdown <- data.frame(Dthdata2[Dthdata2$diabetes == 1, cutdown_columns])

# Check that the numbers match the expected values (965 records).
nrow(Dthcutdown)

# Save the data.
save(Dthcutdown, file='data/Dthcutdown.RData')


# Step 17: Antepenultimate platform.
diabantepenult <- rbindlist(list(MBScutdown, PBScutdown, HMDScutdown, Dthcutdown))

# Check that the number of rows match the expected value (225,941).
nrow(diabantepenult)

# Sort the data by rootlpno and date.
diabantepenult <- diabantepenult[order(diabantepenult$rootlpno, diabantepenult$date), ]

# Create a fileseq variable.
diabantepenult$fileseq <- seq(1, nrow(diabantepenult))

# Create a morbseq variable.
temp_DT <- data.table(diabantepenult)
# setkeyv(temp_DT, "rootlpno")
temp_DT <- temp_DT[, morbseq := 1:.N, by=rootlpno]
diabantepenult <- as.data.frame(temp_DT)

# View the data as shown in the workbook.
View(diabantepenult[1:30, ])
```

R_Solutions - RStudio Source Editor

diabantepenult[1:30,]

Filter

	rootlpno	date	sex	age	sequence	type	fileseq	morbseq
1	1	1998-06-21	1	85	1	1	1	1
2	1	1998-06-26	1	85	1	3	2	2
3	1	1999-01-02	1	86	1	2	3	3
4	1	1999-02-03	1	86	2	3	4	4
5	1	1999-03-15	1	86	3	3	5	5
6	1	1999-03-15	1	86	4	3	6	6
7	1	1999-04-15	1	86	5	3	7	7
8	1	1999-04-19	1	86	6	3	8	8
9	1	1999-05-26	1	86	1	4	9	9
10	2	1992-03-24	2	55	1	1	10	1
11	2	1992-10-25	2	55	2	1	11	2
12	2	1997-10-29	2	60	3	1	12	3
13	2	1998-02-28	2	61	1	2	13	4
14	2	1998-06-23	2	61	2	2	14	5
15	2	1998-12-16	2	61	3	2	15	6
16	2	1999-01-10	2	62	4	1	16	7
17	2	1999-02-14	2	62	4	2	17	8
18	2	1999-03-16	2	62	5	2	18	9
19	2	1999-04-02	2	62	6	2	19	10
20	2	1999-04-04	2	62	5	1	20	11
21	2	1999-05-03	2	62	7	2	21	12
22	2	1999-08-22	2	62	6	1	22	13
23	2	1999-09-20	2	62	8	2	23	14
24	2	1999-10-09	2	62	9	2	24	15
25	2	1999-10-31	2	62	10	2	25	16
26	2	1999-11-23	2	62	11	2	26	17
27	2	1999-11-28	2	62	12	2	27	18
28	2	1999-12-17	2	62	13	2	28	19
29	2	1999-12-18	2	62	14	2	29	20
30	3	1994-11-30	1	76	1	2	30	1

Showing 1 to 30 of 30 entries, 8 total columns

```
# Save the data.
save(diabantepenult, file='data/diabantepenult.RData')


# Step 18: Create penultimate platform.
# Select just the first in the sequences.
diabpenult <- diabantepenult[diabantepenult$sequence == 1, ]
# Discard the sequence variable.
diabpenult <- subset(diabpenult, select = -sequence)

# Create a fileseq variable.
diabpenult$fileseq <- seq(1, nrow(diabpenult))

temp_DT <- data.table(diabpenult)
# setkeyv(temp_DT, "rootlpno")
temp_DT <- temp_DT[, morbseq := 1:.N, by=rootlpno]
diabpenult <- as.data.frame(temp_DT)
rm(temp_DT)

# Check that the number of rows match the expected value (20,352).
nrow(diabpenult)

# Save the data.
save(diabpenult, file='data/diabpenult.RData')


# Step 19: Create the ultimate platform.
diabetes <- diabpenult[diabpenult$morbseq == 1, ]
# Discard the morbseq variable.
diabetes <- subset(diabetes, select = -morbseq)

# Create a fileseq variable.
diabetes$fileseq <- seq(1, nrow(diabetes))

# View the data as shown in the workbook.
View(diabetes[1:30, ])
```

R_Solutions - RStudio Source Editor

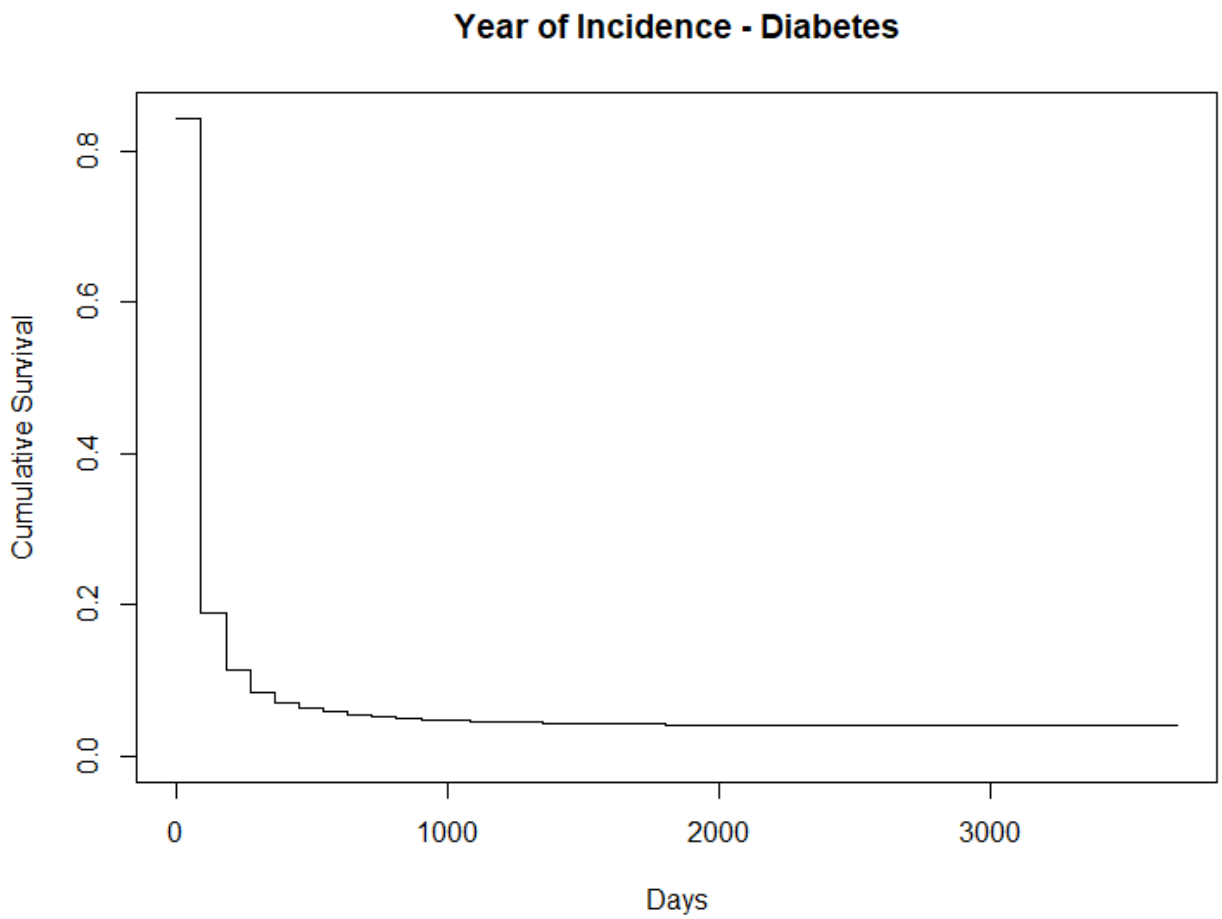
diabetes[1:30,]

Filter

	rootlpno	date	sex	age	type	fileseq
1	1	1998-06-21	1	85	1	1
5	2	1992-03-24	2	55	1	2
7	3	1994-11-30	1	76	2	3
9	4	1993-12-15	1	70	2	4
10	5	1990-05-08	1	67	3	5
14	6	1997-10-10	1	44	2	6
15	7	1995-07-08	2	62	3	7
17	8	1996-05-18	2	43	1	8
18	9	1996-11-15	2	42	2	9
19	10	1999-06-16	1	41	1	10
20	11	1995-08-25	1	90	1	11
21	12	1994-08-18	2	42	2	12
24	13	1990-05-18	1	53	1	13
27	14	1990-05-08	2	78	3	14
28	16	1991-01-16	2	49	1	15
30	17	1993-02-17	2	70	2	16
33	18	1990-07-10	2	63	3	17
36	19	1992-11-01	1	59	2	18
37	21	1993-11-22	2	30	1	19
38	22	1992-09-18	2	64	1	20
40	24	1994-04-22	1	61	2	21
43	25	1997-12-02	2	54	2	22
45	26	1993-03-29	2	71	2	23
47	27	1993-03-01	2	65	2	24
49	28	1990-08-01	2	62	3	25
50	29	1991-01-29	1	43	1	26
53	31	1995-08-12	1	33	1	27
56	33	1997-01-02	2	59	1	28
58	34	1996-01-18	2	88	4	29
59	35	1993-12-07	1	85	1	30

Showing 1 to 30 of 30 entries, 6 total columns

```
# Check that the number of rows match the expected value (10,675).
nrow(diabetes)

# Save the data.
save(diabetes, file='data/diabetes.RData')

# Check the frequencies.
table(diabetes$type)


# Step 20: Backcasting correction.
diabantepenult$yearinc <- years(diabantepenult$date)

# Add  year labels to the frequency table to make the data clear.
frequencies <- table(diabantepenult$yearinc[diabantepenult$morbseq == 1])
names(frequencies) <- seq(1990, 1999)
frequencies
rm(frequencies)


# Step 21:
# Create a flag for any row with a previous morbseq.
diabantepenult$previous <- 0
diabantepenult$previous[diabantepenult$morbseq > 1] <- 1

# Create a reverse survival time (since 31st December 1989) for the first rows.
diabantepenult$revsurti[diabantepenult$previous == 0] <-
        diabantepenult$date[diabantepenult$previous == 0] - as.Date('1989-12-31')

# Calculate difference in days since the previous record.
time_difference <- diff(diabantepenult$date, lag = 1)
# List of rows which have a previous record.
previous_records <- which(diabantepenult$previous == 1)
# Get the time difference for each record with a previous record from the
# previous record.
diabantepenult$revsurti[diabantepenult$previous == 1] <-
time_difference[previous_records - 1]

# Create a survival plot.
# Use a 90 day interval.
survival_step_21 <- survfit(Surv(ceiling(revsurti/90), previous) ~ 1,
                            conf.type = 'none', data = diabantepenult)

plot(survival_step_21, main = 'Year of Incidence - Diabetes',
     xlab = 'Days', xscale = 1/90, ylab = 'Cumulative Survival',
     mark.time=FALSE)
```




```
# Review the survival data.
summary(survival_step_21)


# Step 22: Apply the specified backcasting correction formula.
# Survival correction, Cx = 0.04 / (1.017 * revsurti ^ -0.438) if revsurti < 2160
# or 1, if revsurti > 2,160.
# Only need to calculate for the incident event (morbseq=1).
backcast_table <- diabantepenult[diabantepenult$morbseq == 1, ]

backcast_table$Cx <- 0
backcast_table$Cx[backcast_table$revsurti >= 2160] <- 1

# Make sure we avoid the possibility of a divide by zero error by excluding
# unexpected revsurti values of 0.
rows_to_correct <- which(backcast_table$revsurti < 2160 &
                         backcast_table$revsurti != 0)

# Calculate the correction.
backcast_table$Cx[rows_to_correct] <-
  0.04 / (1.017 * (backcast_table$revsurti[rows_to_correct] ^ -0.438))
```

```
# Constrain Cx to be between 0 and 1.
backcast_table$Cx[backcast_table$Cx < 0] <- 0
backcast_table$Cx[backcast_table$Cx > 1] <- 1

# Get the frequencies of the incident events (already selected).
table(backcast_table$yearinc)

# Get the corrected frequencies (tidied up).
corrected_table <- xtabs(Cx ~ yearinc, data=backcast_table)
names(corrected_table) <- seq(1990,1999)
round(corrected_table, digits=0)


# Step 23:Capture - recapture correction.
temp_DT <- data.table(diabpenult)
# setkeyv(temp_DT, "rootlpno")
temp_DT[, mbs := .(ifelse(any(type==1), 1L, 0L)), by=rootlpno]
temp_DT[, pbs := .(ifelse(any(type==2), 1L, 0L)), by=rootlpno]
temp_DT[, hmds := .(ifelse(any(type==3), 1L, 0L)), by=rootlpno]
temp_DT[, dth := .(ifelse(any(type==4), 1L, 0L)), by=rootlpno]
temp_DT[, cth := .(ifelse(mbs==1 | pbs==1, 1L, 0L)), by=rootlpno]
temp_DT[, state := .(ifelse(hmds==1 | dth==1, 1L, 0L)), by=rootlpno]
diabpenult <- as.data.frame(temp_DT)
rm(temp_DT)


# Step 24: Get the frequencies.
# Select only the index event.
diabpenult <- diabpenult[diabpenult$morbseq == 1,]
# Get the frequencies for all the combinations.
f1 <- ftable(xtabs(~ cth + state, data=diabpenult))
f2 <- ftable(xtabs(~ mbs + pbs + hmds + dth, data=diabpenult))

# Display the tables.
print(f1)
print(f2)

# Calculate the Chapman estimator and confidence intervals.
Nstate <- (f1[1,2] + f1[2,2] + 1)   # first = cth, second = state
Ncth   <- (f1[2,1] + f1[2,2] + 1)
Nboth  <- (f1[2,2] + 1)
N <- as.integer(((Nstate * Ncth) / Nboth) - 1)

CIupper <- as.integer(N + 1.96 *
        sqrt((Ncth * Nstate * (Ncth - Nboth) * (Nstate - Nboth)) /
                ((Nboth ^ 2) * (Nboth + 1))))
CIlower <- as.integer(N - 1.96 *
        sqrt((Ncth * Nstate * (Ncth - Nboth) * (Nstate - Nboth)) /
                ((Nboth ^ 2) * (Nboth + 1))))

cat('\nThe Chapman estimator is', N, 'with 95% confidence intervals', CIlower,
    'to', CIupper, '.\n', sep=' ')
```

TRAINING SESSION 1: TIDYVERSE-R SYNTAX SOLUTIONS

```
# Preparation
# Start a new R session by opening the project.

# Load the libraries used in the exercise.

library(dplyr)
library(magrittr)
library(lubridate)
library(survival)
library(survminer)

################################### Exercise 1 ###################################

#### Steps 1-4: Tag, sequence and cut down MBSdata file

#### Step 1: Open the MBSdata data file (MBSdata.RData)
# Alternatively, if you navigate to the data file in the Files window, you can
# click on the file name to load the data into the environment.
# If you click on the table symbol to the right of the MBSdata
# item in the Environment panel, it will execute `View(MBSdata)`.
load('data/MBSdata.RData')
head(MBSdata)


#### Step 2: Tag all records that mention a diabetes-related MBS item
diabetes_codes <- c(66551, 66554, 66557, 73815, 73840, 66319, 66322)
MBSdata2 <- MBSdata %>%
    mutate(glyhb=ifelse(mbsitem %in% diabetes_codes, 1, 0))


#### Step 3: Create sequence variables
# In general, we'll sort the variables/columns so that the items we wish to
# sequence are at the start of the list and we can then use row_number() to get
# the sequence number without needing to create a variable to do the job.
MBSdata2 <- MBSdata2 %>%
    # Sort the list as required then add a column containing the row-numbers.
    arrange(rootlpno, servdate, mbsitem) %>%
    mutate(fileseq=row_number())

MBSdata2 <- MBSdata2 %>%
    # Sort the list as required.
    arrange(rootlpno, servdate, mbsitem) %>%
    # Group by the grouping variable (all following actions are performed on the
    # separate groups).
    group_by(rootlpno) %>%
    # Add the row number for each item in the group to the morbseq column.
    mutate(morbseq=row_number()) %>%
    # Remove the grouping.
    ungroup()

MBSdata2 <- MBSdata2 %>%
    arrange(rootlpno, servdate, mbsitem) %>%
```

```
    group_by(rootlpno) %>%
    # Now we sort in reverse order of the tag (i.e. 1's first) so that they
    # are located in rows 1...
    arrange(-glyhb) %>%
    mutate(servseq = ifelse(glyhb==1, row_number(), NA)) %>%
    ungroup() %>%
    # Just to make sure, re-sort in the expected order.
    arrange(rootlpno, servdate, mbsitem)

# Preview as in the workbook.
View(MBSdata2[704:729,], 26)
```

R_Solutions - RStudio Source Editor

MBSdata2[704:729,]

Filter

	rootlpno	mbsitem	servdate	sex	age	postcode	glyhb	fileseq	morbseq	servseq
704	4	116	1999-12-01	1	76	6026	0	704	174	NA
705	5	66209	1990-02-18	1	67	6156	0	705	1	NA
706	5	69207	1990-03-30	1	67	6156	0	706	2	NA
707	5	66211	1990-05-13	1	67	6156	0	707	3	NA
708	5	57712	1991-10-19	1	68	6156	0	708	4	NA
709	5	57712	1991-10-19	1	68	6156	0	709	5	NA
710	5	136	1992-01-15	1	69	6156	0	710	6	NA
711	5	66203	1992-01-15	1	69	6156	0	711	7	NA
712	5	66291	1992-01-15	1	69	6156	0	712	8	NA
713	5	66303	1992-01-15	1	69	6156	0	713	9	NA
714	5	65007	1992-04-17	1	69	6156	0	714	10	NA
715	5	66211	1992-04-17	1	69	6156	0	715	11	NA
716	5	66551	1992-04-17	1	69	6156	1	716	12	1
717	5	73907	1992-04-17	1	69	6156	0	717	13	NA
718	5	66596	1992-04-25	1	69	6156	0	718	14	NA
719	5	66602	1992-04-25	1	69	6156	0	719	15	NA
720	5	73907	1992-04-25	1	69	6156	0	720	16	NA
721	5	66211	1992-07-30	1	69	6156	0	721	17	NA
722	5	66291	1992-07-30	1	69	6156	0	722	18	NA
723	5	73917	1992-07-30	1	69	6156	0	723	19	NA
724	5	66211	1992-12-03	1	69	6156	0	724	20	NA
725	5	66551	1992-12-03	1	69	6156	1	725	21	2
726	5	73917	1992-12-03	1	69	6156	0	726	22	NA
727	5	65007	1993-02-26	1	70	6156	0	727	23	NA
728	5	66211	1993-02-26	1	70	6156	0	728	24	NA
729	5	66551	1993-02-26	1	70	6156	1	729	25	3

Showing 1 to 26 of 26 entries, 10 total columns

```
# Save processed data to a file. Loading the file later will restore the data-
# frame with the name specified here (in general, not necessarily the file name).
save(MBSdata2, file='data/MBSdata2.RData')


#### Step 4: Create the cutdown dataset
MBScutdown <- MBSdata2 %>%
    # Select chooses the columns and allows us to rename them in the same
    # command.
    select(rootlpno, date=servdate, sex, age, sequence=servseq) %>%
    filter(sequence>=1) %>%
    mutate(type=1)

# Check the number of records.
count(MBScutdown)
count(MBScutdown %>% filter(sequence==1))

save(MBScutdown, file='data/MBScutdown.RData')


### Steps 5-8: Tag, sequence and cut down PBSdata file

#### Step 5: Open the PBSdata data file (PBSdata.RData)
load('data/PBSdata.RData')
PBSdata2 <- PBSdata %>% arrange(rootlpno, dispdate)


#### Step 6: Tag all records that mention a diabetes-related PBS item
oral_hypoglycaemic <- c(1202, 1801, 2178, 2430, 2440, 2449, 2607, 2720, 2939, 2940,
                        8188, 8189, 8391, 8392, 8450, 8451, 8452, 8533, 8535, 8607,
                        8687, 8688, 8689, 8690, 8691, 8692, 8693, 8694, 8695, 8696,
                        8810, 8811)
insulin_compound <- c(1425, 1426, 1429, 1430, 1431, 1461, 1462, 1531, 1532, 1533,
                      1534, 1535, 1537, 1591, 1592, 1710, 1711, 1713, 1715, 1716,
                      1718, 1721, 1722, 1761, 1762, 1763, 2061, 2062, 8006, 8084,
                      8085, 8212, 8390, 8435, 8571, 8609)
PBSdata2 <- PBSdata2 %>%
    mutate(diabmed=ifelse(pbsitem %in% insulin_compound, 2,
                          ifelse(pbsitem %in% oral_hypoglycaemic, 1, 0)))


#### Step 7: Create sequence variables
PBSdata2 <- PBSdata2 %>%
    arrange(rootlpno, dispdate, pbsitem) %>%
    mutate(fileseq=row_number())

PBSdata2 <- PBSdata2 %>%
    arrange(rootlpno, dispdate, pbsitem) %>%
    group_by(rootlpno) %>%
    mutate(morbseq=row_number()) %>%
    ungroup()

PBSdata2 <- PBSdata2 %>%
    arrange(rootlpno, dispdate, pbsitem) %>%
    group_by(rootlpno) %>%
```

```
    # Arrange by rootlpno, then diabmed>0 (- puts TRUE first), then date then
    # item.
    # This works because (diabmed>0) is TRUE or FALSE which have values 1 or 0.
    arrange(rootlpno, -(diabmed>0), dispdate, pbsitem) %>%
    mutate(dispseq=ifelse(diabmed>0, row_number(), NA)) %>%
    ungroup() %>%
    arrange(rootlpno, dispdate, pbsitem)

View(PBSdata2[51:78,], 30)
```

R_Solutions - RStudio Source Editor

PBSdata2[51:78,]

Filter

	rootlpno	pbsitem	scripts	dispdate	sex	age	postcode	diabmed	fileseq	morbseq	dispseq
51	1	2417	1	1999-12-20	1	86	6010	0	51	51	NA
52	2	2367	1	1998-02-24	2	61	6108	0	52	1	NA
53	2	2449	1	1998-02-28	2	61	6108	1	53	2	1
54	2	1279	1	1998-06-23	2	61	6108	0	54	3	NA
55	2	2367	1	1998-06-23	2	61	6108	0	55	4	NA
56	2	2449	1	1998-06-23	2	61	6108	1	56	5	2
57	2	1279	1	1998-08-07	2	61	6108	0	57	6	NA
58	2	2367	1	1998-08-25	2	61	6108	0	58	7	NA
59	2	1279	1	1998-09-10	2	61	6108	0	59	8	NA
60	2	2367	1	1998-09-27	2	61	6108	0	60	9	NA
61	2	2367	1	1998-10-27	2	61	6108	0	61	10	NA
62	2	1279	1	1998-11-16	2	61	6108	0	62	11	NA
63	2	2367	1	1998-11-16	2	61	6108	0	63	12	NA
64	2	1279	1	1998-12-16	2	61	6108	0	64	13	NA
65	2	2367	1	1998-12-16	2	61	6108	0	65	14	NA
66	2	2449	1	1998-12-16	2	61	6108	1	66	15	3
67	2	1279	1	1999-01-17	2	62	6107	0	67	16	NA
68	2	2367	1	1999-01-17	2	62	6107	0	68	17	NA
69	2	3050	1	1999-01-26	2	62	6107	0	69	18	NA
70	2	1279	1	1999-02-14	2	62	6107	0	70	19	NA
71	2	2367	1	1999-02-14	2	62	6107	0	71	20	NA
72	2	2449	1	1999-02-14	2	62	6107	1	72	21	4
73	2	3050	1	1999-02-14	2	62	6107	0	73	22	NA
74	2	8213	1	1999-02-14	2	62	6107	0	74	23	NA
75	2	1279	1	1999-03-05	2	62	6107	0	75	24	NA
76	2	2367	1	1999-03-05	2	62	6107	0	76	25	NA
77	2	8213	1	1999-03-05	2	62	6107	0	77	26	NA
78	2	2449	1	1999-03-16	2	62	6107	1	78	27	5

Showing 1 to 28 of 28 entries, 11 total columns

```
save(PBSdata2, file='data/PBSdata2.RData')
```

```
#### Step 8: Create the cutdown dataset
PBScutdown <- PBSdata2 %>%
    select(rootlpno, date=dispdate, sex, age, sequence=dispseq) %>%
    filter(sequence>=1) %>%
    mutate(type=2)

count(PBScutdown)
count(PBScutdown %>% filter(sequence==1))

save(PBScutdown, file='data/PBScutdown.RData')


### Steps 9-12: Tag, sequence and cut down HMDSdata file

#### Step 9: Open the HMDSdata data file (HMDSdata.RData)
load('data/HMDSdata.RData')

HMDSdata2 <- HMDSdata %>%
    # There are a significant number of duplicated records but they aeem to have
    # been included in the workbook, so are not removed here.
    # distinct() %>%
    arrange(rootlpno, admdate)


#### Step 10: Tag all records that mention a diabetes-related code in any field
diabetes_codes <- c(format(seq(250.00,250.99, 0.01),nsmall=2),
                    'V77.1',
                    paste("E", format(seq(10.00, 14.99, 0.01), width=4), sep=""))

HMDSdata2 <- HMDSdata2 %>% mutate(diabetes=0)

# Work through columns 'diag1' to 'diag21' and look for any which contains a
# diabetes code.
# The paste() call concatenates the 'diag' with the loop counter, i to give
# strings 'diag1', 'diag2' etc., the sym() call converts the string to a
# symbol (i.e. a variable name), the !! evaluates the symbol which in this case
# produces the column name. So we end up with ifelse(diag1 %in% ...,
# ifelse(diag2 %in% ... and so on.
for(i in 1:21){
    HMDSdata2 <- HMDSdata2 %>%
        mutate(diabetes=ifelse(!!sym(paste('diag', i, sep='')) %in% diabetes_codes, 1,
diabetes))
}


#### Step 11: Create sequence variables
HMDSdata2 <- HMDSdata2 %>%
    arrange(rootlpno, admdate, sepdate) %>%
    mutate(fileseq=row_number())

HMDSdata2 <- HMDSdata2 %>%
    arrange(rootlpno, admdate, sepdate) %>%
    group_by(rootlpno) %>%
    mutate(morbseq = row_number()) %>%
    ungroup()
```

```
HMDSdata2 <- HMDSdata2 %>%
    arrange(rootlpno, admdate, sepdate) %>%
    group_by(rootlpno) %>%
    arrange(rootlpno, -(diabetes>0), admdate, sepdate) %>%
    mutate(condseq = ifelse(diabetes>0, row_number(), NA)) %>%
    ungroup() %>%
    arrange(rootlpno, admdate, sepdate)

View(HMDSdata2[1:21,40:48], 22)
```

R_Solutions - RStudio Source Editor

HMDSdata2[1:21, 40:48]

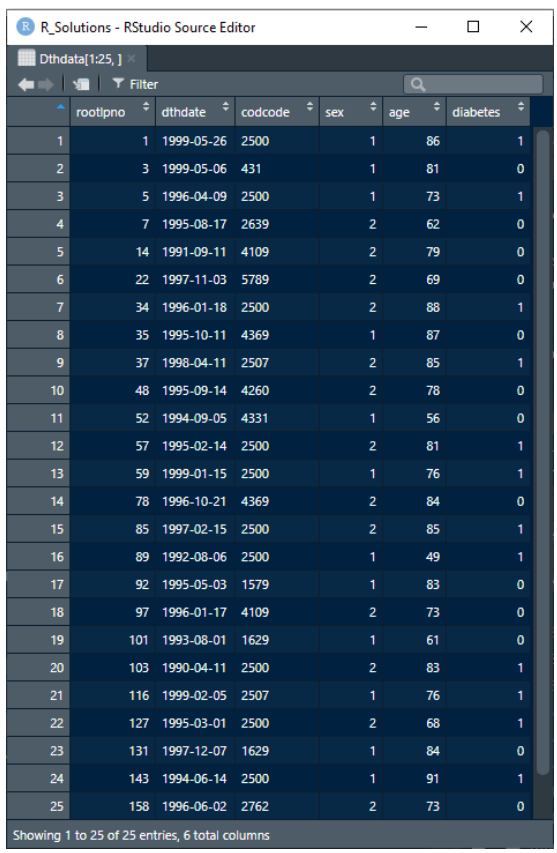
Filter

	proc11	ecode1	ecode3	ecode4	ecode5	diabetes	fileseq	morbseq	condseq
1	NA	NA	NA	NA	NA	0	1	1	NA
2	NA	NA	NA	NA	NA	1	2	2	1
3	NA	NA	NA	NA	NA	1	3	3	2
4	NA	NA	NA	NA	NA	1	4	4	3
5	NA	NA	NA	NA	NA	1	5	5	4
6	NA	NA	NA	NA	NA	1	6	6	5
7	NA	NA	NA	NA	NA	1	7	7	6
8	NA	NA	NA	NA	NA	0	8	1	NA
9	NA	NA	NA	NA	NA	1	9	2	1
10	NA	NA	NA	NA	NA	1	10	3	2
11	NA	NA	NA	NA	NA	1	11	4	3
12	NA	NA	NA	NA	NA	1	12	5	4
13	NA	NA	NA	NA	NA	1	13	6	5
14	NA	NA	NA	NA	NA	0	14	1	NA
15	NA	NA	NA	NA	NA	1	15	1	1
16	NA	NA	NA	NA	NA	1	16	2	2
17	NA	NA	NA	NA	NA	1	17	3	3
18	NA	NA	NA	NA	NA	1	18	4	4
19	NA	NA	NA	NA	NA	0	19	5	NA
20	NA	NA	NA	NA	NA	1	20	6	5
21	NA	NA	NA	NA	NA	1	21	7	6

Showing 1 to 21 of 21 entries, 9 total columns

```
save(HMDSdata2, file='data/HMDSdata2.RData')
```

```
#### Step 12: Create the cutdown dataset
HMDScutdown <- HMDSdata2 %>%
    select(rootlpno, date=admdate, sex, age, sequence=condseq) %>%
    filter(sequence>=1) %>%
    mutate(type=3)

count(HMDScutdown)
count(HMDScutdown %>% filter(sequence==1))

save(HMDScutdown, file='data/HMDScutdown.RData')


### Steps 13-16: Tag, sequence and cut down Dthdata file

#### Step 13: Open the Dthdata data file (Dthdata.RData)
load('data/Dthdata.RData')
Dthdata2 <- Dthdata %>% arrange(rootlpno)


#### Step 14: Tag all records that mention a diabetes-related code in any field
diabetes_cod_codes <- c(seq(2500, 2509), "V771", paste("E", seq(100, 149), sep=""))
Dthdata2 <- Dthdata2 %>%
    mutate(diabetes=ifelse(codcode %in% diabetes_cod_codes, 1, 0))


#### Step 15: No sequence variables to create; just check the data
View(Dthdata2, 25)
```



	rootlpno	dthdate	codcode	sex	age	diabetes
1	1	1999-05-26	2500	1	86	1
2	3	1999-05-06	431	1	81	0
3	5	1996-04-09	2500	1	73	1
4	7	1995-08-17	2639	2	62	0
5	14	1991-09-11	4109	2	79	0
6	22	1997-11-03	5789	2	69	0
7	34	1996-01-18	2500	2	88	1
8	35	1995-10-11	4369	1	87	0
9	37	1998-04-11	2507	2	85	1
10	48	1995-09-14	4260	2	78	0
11	52	1994-09-05	4331	1	56	0
12	57	1995-02-14	2500	2	81	1
13	59	1999-01-15	2500	1	76	1
14	78	1996-10-21	4369	2	84	0
15	85	1997-02-15	2500	2	85	1
16	89	1992-08-06	2500	1	49	1
17	92	1995-05-03	1579	1	83	0
18	97	1996-01-17	4109	2	73	0
19	101	1993-08-01	1629	1	61	0
20	103	1990-04-11	2500	2	83	1
21	116	1999-02-05	2507	1	76	1
22	127	1995-03-01	2500	2	68	1
23	131	1997-12-07	1629	1	84	0
24	143	1994-06-14	2500	1	91	1
25	158	1996-06-02	2762	2	73	0

```
#### Step 16: Create the cutdown dataset
save(Dthdata2, file='data/Dthdata2.RData')

Dthcutdown <- Dthdata2 %>%
    filter(diabetes==1) %>%
    select(rootlpno, date=dthdate, sex, age) %>%
    mutate(sequence=1, type=4)

count(Dthcutdown)

save(Dthcutdown, file='data/Dthcutdown.RData')


#### Steps 17-19: Create antepenultimate -> penultimate -> ultimate

#### Step 17: Create the antepenultimate platform
diabantepenult <- MBScutdown %>%
    rbind(PBScutdown, HMDScutdown, Dthcutdown)

diabantepenult <- diabantepenult %>%
    arrange(rootlpno, date) %>%
    mutate(fileseq=row_number())

diabantepenult <- diabantepenult %>%
    arrange(rootlpno, date) %>%
    group_by(rootlpno) %>%
    mutate(morbseq = row_number()) %>%
    ungroup()

View(diabantepenult[1:30,], 30)
```

R_Solutions - RStudio Source Editor

diabantepenult[1:30,]

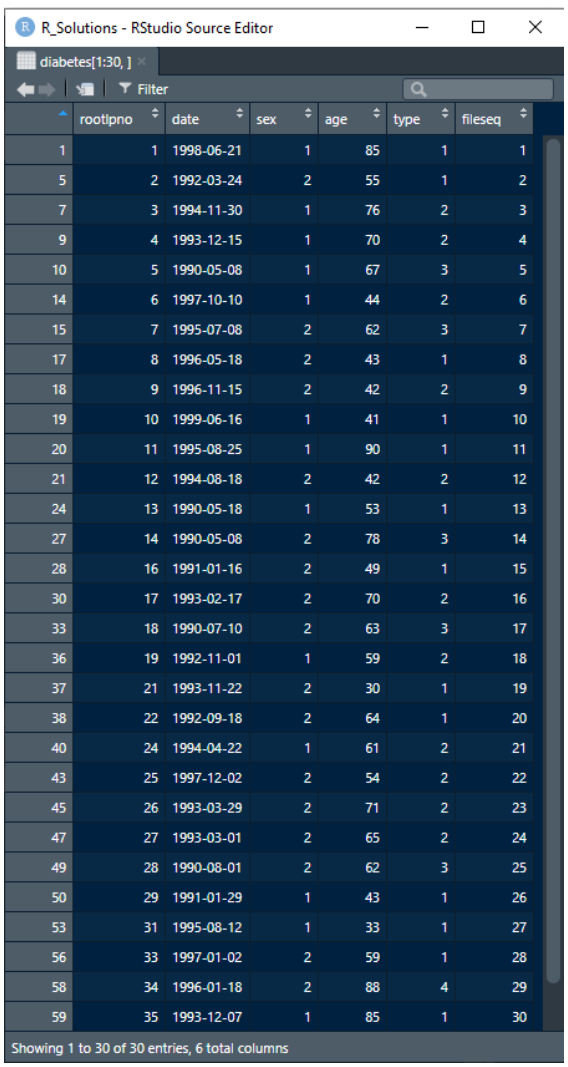
Filter

	rootlpno	date	sex	age	sequence	type	fileseq	morbseq
1	1	1998-06-21	1	85	1	1	1	1
2	1	1998-06-26	1	85	1	3	2	2
3	1	1999-01-02	1	86	1	2	3	3
4	1	1999-02-03	1	86	2	3	4	4
5	1	1999-03-15	1	86	3	3	5	5
6	1	1999-03-15	1	86	4	3	6	6
7	1	1999-04-15	1	86	5	3	7	7
8	1	1999-04-19	1	86	6	3	8	8
9	1	1999-05-26	1	86	1	4	9	9
10	2	1992-03-24	2	55	1	1	10	1
11	2	1992-10-25	2	55	2	1	11	2
12	2	1997-10-29	2	60	3	1	12	3
13	2	1998-02-28	2	61	1	2	13	4
14	2	1998-06-23	2	61	2	2	14	5
15	2	1998-12-16	2	61	3	2	15	6
16	2	1999-01-10	2	62	4	1	16	7
17	2	1999-02-14	2	62	4	2	17	8
18	2	1999-03-16	2	62	5	2	18	9
19	2	1999-04-02	2	62	6	2	19	10
20	2	1999-04-04	2	62	5	1	20	11
21	2	1999-05-03	2	62	7	2	21	12
22	2	1999-08-22	2	62	6	1	22	13
23	2	1999-09-20	2	62	8	2	23	14
24	2	1999-10-09	2	62	9	2	24	15
25	2	1999-10-31	2	62	10	2	25	16
26	2	1999-11-23	2	62	11	2	26	17
27	2	1999-11-28	2	62	12	2	27	18
28	2	1999-12-17	2	62	13	2	28	19
29	2	1999-12-18	2	62	14	2	29	20

Showing 1 to 30 of 30 entries, 8 total columns

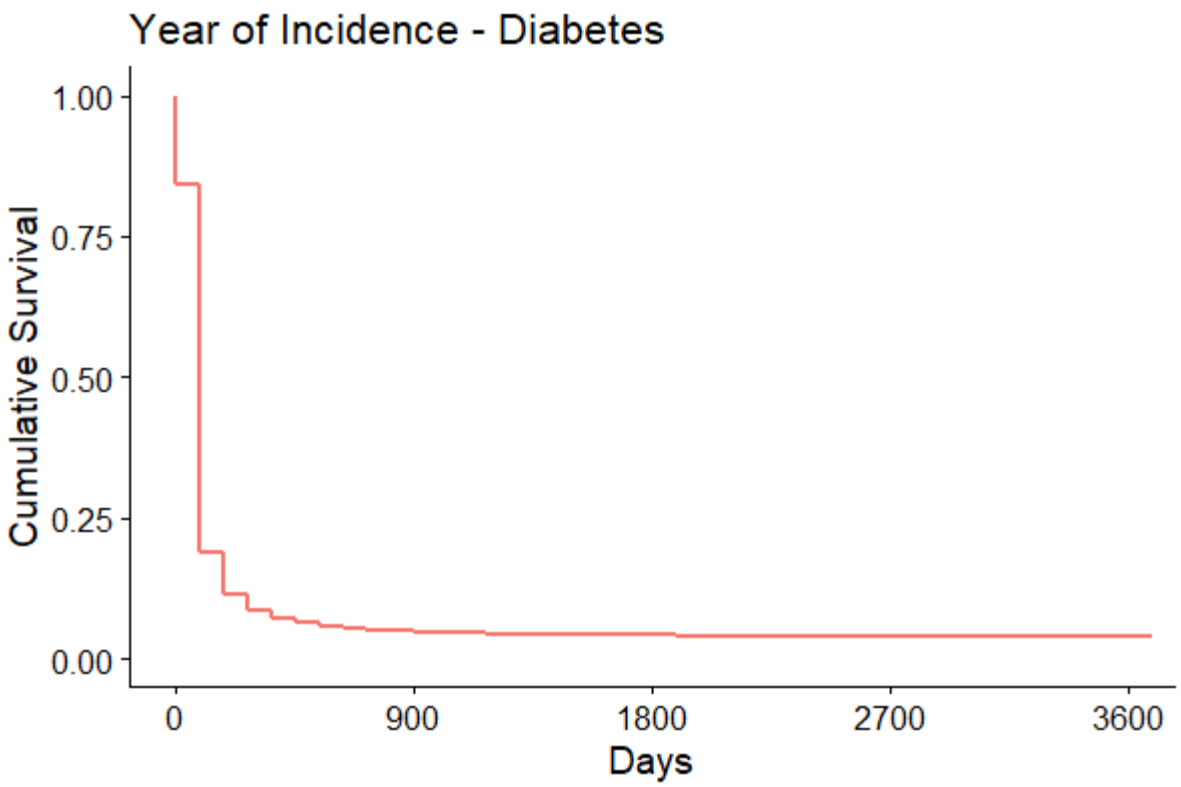
```
save(diabantepenult, file='data/diabantepenult.RData')


#### Step 18: Create the penultimate platform
diabpenult <- diabantepenult %>%
    filter(sequence==1) %>%
    # Remove the sequence variable/column using - before the name.
    select(-sequence)

diabpenult <- diabpenult %>%
```

```
    arrange(rootlpno, date, type) %>%
    mutate(fileseq=row_number())

diabpenult <- diabpenult %>%
    arrange(rootlpno, date, type) %>%
    group_by(rootlpno) %>%
    mutate(morbseq = row_number()) %>%
    ungroup()

save(diabpenult, file='data/diabpenult.RData')


#### Step 19: Create the ultimate platform
diabetes <- diabpenult %>%
    arrange(rootlpno, date) %>%
    filter(morbseq==1) %>%
    select(-morbseq)  %>%
    mutate(fileseq=row_number())

View(diabetes[1:30,], 30)
```



	rootlpno	date	sex	age	type	fileseq
1	1	1998-06-21	1	85	1	1
5	2	1992-03-24	2	55	1	2
7	3	1994-11-30	1	76	2	3
9	4	1993-12-15	1	70	2	4
10	5	1990-05-08	1	67	3	5
14	6	1997-10-10	1	44	2	6
15	7	1995-07-08	2	62	3	7
17	8	1996-05-18	2	43	1	8
18	9	1996-11-15	2	42	2	9
19	10	1999-06-16	1	41	1	10
20	11	1995-08-25	1	90	1	11
21	12	1994-08-18	2	42	2	12
24	13	1990-05-18	1	53	1	13
27	14	1990-05-08	2	78	3	14
28	16	1991-01-16	2	49	1	15
30	17	1993-02-17	2	70	2	16
33	18	1990-07-10	2	63	3	17
36	19	1992-11-01	1	59	2	18
37	21	1993-11-22	2	30	1	19
38	22	1992-09-18	2	64	1	20
40	24	1994-04-22	1	61	2	21
43	25	1997-12-02	2	54	2	22
45	26	1993-03-29	2	71	2	23
47	27	1993-03-01	2	65	2	24
49	28	1990-08-01	2	62	3	25
50	29	1991-01-29	1	43	1	26
53	31	1995-08-12	1	33	1	27
56	33	1997-01-02	2	59	1	28
58	34	1996-01-18	2	88	4	29
59	35	1993-12-07	1	85	1	30

```
save(diabetes, file='data/diabetes.RData')

# Get the frequencies of each type.
table(diabetes$type)


#### Steps 20-22: Backcasting correction for incidence estimation

####  Step 20: Create the 'yearinc' variable
# load('data/diabantepenult.RData')
# Use `year()` from the lubridate library.
diabantepenult <- diabantepenult %>%
    mutate(yearinc=year(date))

table((diabantepenult %>% filter(morbseq==1))$yearinc)

#### Step 21: Survival analysis
diabantepenult <- diabantepenult %>%
    mutate(previous=ifelse(morbseq>1, 1, 0))

diabantepenult <- diabantepenult %>%
    mutate(revsurti=ifelse(previous==0, interval(as.Date('1989-12-31'), date) %/%
days(1), NA))

diabantepenult <- diabantepenult %>%
    mutate(revsurti=ifelse(previous==0, revsurti, interval(lag(date), date) %/%
days(1)))

# Use the `survival` library for Surv and survfit.
# Note the conf.int parameter requires a value 0 not a string input.
surv1.21 <- survfit(Surv(ceiling(revsurti / 90), previous) ~ 1, conf.int=0,
        data=diabantepenult)
plot(surv1.21, ylim=c(0,1), xscale=1/90, mark.time=FALSE,
     main="Year of Incidence - Diabetes",
     xlab="Days", ylab="Cumulative Survival")
# View a summary of the survival object.
summary(surv1.21)

# Use the `survminer` library for ggplot-type plots.
ggsurvplot(surv1.21, censor=FALSE, ylim=c(0, 1), xscale=1/90,
           title='Year of Incidence - Diabetes',
           xlab='Days', ylab='Cumulative Survival',
           legend='none')
```




```
#### Step 22: Apply the correction factors
diabantepenult <- diabantepenult %>%
    mutate(cx=ifelse(revsurti>=2160, 1,
                     0.04 / (1.017 * (revsurti ^ -0.438))))
diabantepenult <- diabantepenult %>%
    mutate(cx=ifelse(cx>=1, 1, ifelse(cx<=0, 0, cx)))

# table((diabantepenult %>% filter(morbseq==1))$yearinc)
xtabs(~yearinc, data=(diabantepenult %>% filter(morbseq==1))) %>% round(digits=0)
xtabs(cx~yearinc, data=(diabantepenult %>% filter(morbseq==1))) %>% round(digits=0)


#### Steps 23-24: Capture-Recapture Correction for Period Prevalence Estimation

#### Step 23: Generate the variables for the Chapman estimator
load('data/diabpenult.RData')
diabpenult <- diabpenult %>%
    arrange(fileseq)

mbs <- diabpenult %>%
    group_by(rootlpno) %>%
    mutate(mbs=ifelse(any(type==1), 1, 0)) %>%
    ungroup() #%>%
    #distinct(rootlpno, mbs)

pbs <- diabpenult %>%
    group_by(rootlpno) %>%
```

```
    mutate(pbs=ifelse(any(type==2), 1, 0)) %>%
    ungroup() %>%
    distinct(rootlpno, pbs)

hmds <- diabpenult %>%
    group_by(rootlpno) %>%
    mutate(hmds=ifelse(any(type==3), 1, 0)) %>%
    ungroup() %>%
    distinct(rootlpno, hmds)

dth <- diabpenult %>%
    group_by(rootlpno) %>%
    mutate(dth=ifelse(any(type==4), 1, 0)) %>%
    ungroup() %>%
    distinct(rootlpno, dth)

diabpenult <- diabpenult %>%
    merge(mbs) %>% merge(pbs) %>% merge(hmds) %>% merge(dth)

diabpenult <- diabpenult %>%
    mutate(cth=ifelse(mbs==1 | pbs==1, 1, 0)) %>%
    mutate(state=ifelse(hmds==1 | dth==1, 1, 0))


#### Step 24: Perform the cross-tabulation
xtabs(~cth+state, data=(diabpenult %>% filter(morbseq==1)))
ftable(xtabs(~mbs+pbs+hmds+dth, data=(diabpenult %>% filter(morbseq==1))))

# Now do the Chapman calculation:
# N = (((n1 + 1) x (n2 + 1)) / (n1.2 + 1)) - 1
tab <- xtabs(~cth+state, data=(diabpenult %>% filter(morbseq==1)))
cat('Observed number of cases: ', as.integer(tab[1,1] + tab[1,2] + tab[2,1] +
tab[2,2]), '.\n', sep='')
n1 <- tab[2,1] + tab[2,2]
n2 <- tab[1,2] + tab[2,2]
n1.2 <- tab[2,2]
N <- (((n1 + 1) * (n2 + 1)) / (n1.2 + 1)) - 1
cat('Estimated total number of cases: ', as.integer(N), '.\n', sep='')
n95_conf <- 1.96 * sqrt(((n1 + 1) * (n2 + 1) * (n1 - n1.2) * (n2 - n1.2)) / (((n1.2 -
1) ^ 2) * (n1.2 + 2)))
n95_lower <- N - n95_conf
n95_upper <- N + n95_conf
cat('95% confidence interval: ', as.integer(n95_lower), ' - ', as.integer(n95_upper),
'.\n', sep='')

# Poisson log-linear regression model.
tab <- ftable(xtabs(~mbs+pbs+hmds+dth, data=(diabpenult %>% filter(morbseq==1))))

poisson_tab <- data.frame(mbs=c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1),
                          pbs=c(0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1),
                          hmds=c(0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1),
                          dth=c(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1),
                          weight=c(tab[1,1], tab[1,2], tab[2,1], tab[2,2],
                                   tab[3,1], tab[3,2], tab[4,1], tab[4,2],
                                   tab[5,1], tab[5,2], tab[6,1], tab[6,2],
```

```
                                  tab[7,1], tab[7,2], tab[8,1], tab[8,2]))
model <- glm(weight~mbs*pbs*hmds*dth, family=poisson, data=poisson_tab)
model1 <- update(model, .~. -mbs:pbs:hmds:dth)
model2 <- update(model1, .~. -dth)
summary(model2)
sum_betas <- sum(model2$coefficients)
estimate <- exp(sum_betas)
cat('Estimated missing cell (log-linear): ', as.integer(estimate), '.\n', sep='')

library(discSurv)
temp <- diabantepenult %>%
    mutate(time=as.character(revsurti), censor=as.character(previous)) %>%
    select(fileseq, time, censor) %>%
    as.data.frame()
lifeTable(dataSet=temp, timeColumn='time', censColumn='censor')
```

TRAINING SESSION 2: R SYNTAX SOLUTIONS

```
################################################################################
# AALHD DAY 2
# R Syntax solutions
#
# Dr Pete Arnold based on code by Dr Joanne Demmler
# Updated March 2021
################################################################################
# Hints about what has been used in this solution.
#
################################################################################

library(data.table)
# psych has the describe function used in step 10.
library(psych)

################################### Exercise 2 ###################################

# Preparation.
# Use the same project as for Day 1.

# Step 1: Load the modified PBS data from Day 1.
load('data/PBSdata2.RData')


# Step 2: Create new variables, ohgaseq and inspseq.
# lookup table for where to start the count
# tmp <- PBSdata2[which(PBSdata2$diabmed == 1), c("rootlpno","morbseq")]

DT <- data.table(PBSdata2)
setkeyv(DT, c('rootlpno', 'morbseq'))  # index table, needed for merge

# Find the minimum morbseq for each rootlpno if diabmed = 1.
DT_diabmed1 <- DT[diabmed == 1]
DT_minmorbseq <- DT[DT_diabmed1, .(morbseq = min(morbseq)), by=rootlpno]

# Create ohgaseq and set to 0 ("L" makes it an integer)
DT[, ohgaseq := 0L]

# Match rows if morbseq is greater than or equal to minimum morbseq.
DT_latermorbseq <- DT[DT_minmorbseq, .I[morbseq >= i.morbseq], by=.EACHI]$V1
# Number each matching row for each rootlpno.
DT[DT_latermorbseq, ohgaseq := 1:.N, by=rootlpno]

# Find the minimum morbseq for each rootlpno if diabmed = 2.
DT_diabmed2 <- DT[diabmed==2]
DT_minmorbseq <- DT[DT_diabmed2, .(morbseq = min(morbseq)), by=rootlpno]

# Create inspseq and set to 0.
DT[, inspseq := 0L]

# Match rows if morbseq is greater than or equal to minimum morbseq.
DT_latermorbseq <- DT[DT_minmorbseq, .I[morbseq >= i.morbseq], by=.EACHI]$V1
```

```
# Number each matching row for each rootlpno.
DT[DT_latermorbseq, inspseq := 1:.N, by=rootlpno]

# Revert to a data frame and save the data.
PBSdata3 <- as.data.frame(DT)
save(PBSdata3, file='data/PBSdata3.RData')


# Step 3: Extract and save patients with OHGAs but no insulin.
PBSohga <- PBSdata3[PBSdata3$ohgaseq == 1 & PBSdata3$inspseq == 0, c('rootlpno',
'dispdate')]
save(PBSohga, file='data/PBSohga.RData')


# Step 4: Merge these with the diabetes table from yesterday and save.
load('data/diabetes.RData')
diabetes2 <- merge(diabetes, PBSohga, all.x=TRUE)
names(diabetes2)[names(diabetes2) == 'dispdate'] <- 'stage2'
save(diabetes2, file='data/diabetes2.RData')


# Step 5: Extract and save patients with insulin and save.
PBSinsp <- PBSdata3[PBSdata3$inspseq == 1, c('rootlpno', 'dispdate')]
save(PBSinsp, file='data/PBSinsp.RData')


# Step 6:
diabetes3 <- merge(diabetes2, PBSinsp, all.x=TRUE)
names(diabetes3)[names(diabetes3) == 'dispdate'] <- 'stage3'
save(diabetes3, file='data/diabetes3.RData')


# Step 7:
load('data/Dthdata2.RData')
diabetes4 <- merge(diabetes3, Dthdata2[, c('rootlpno', 'dthdate')], all.x=TRUE)
names(diabetes4)[names(diabetes4) == 'dthdate'] <- 'exit'

diabetes4$dead <- 0
diabetes4$dead[!is.na(diabetes4$exit)] <- 1

# Recode exit date for end of study if no exit or exit after the end.
diabetes4$exit[diabetes4$dead == 0 | diabetes4$exit >= '1999-12-31'] <-
          as.Date('1999-12-31')

# Save the data.
save(diabetes4, file='data/diabetes4.RData')


# Step 8: Add the stage person-time variables.
diabetes4$stgpt1 <- 0
diabetes4$stgpt2 <- 0
diabetes4$stgpt3 <- 0
diabetes4$totpt <- 0

has_stage3 <- !is.na(diabetes4$stage3)
```

```
diabetes4$stgpt3[has_stage3] <- diabetes4$exit[has_stage3] -
                                diabetes4$stage3[has_stage3] + 1

has_stage2 <- !is.na(diabetes4$stage2)
diabetes4$stgpt2[has_stage2] <- diabetes4$exit[has_stage2] -
                                diabetes4$stage2[has_stage2] + 1 -
                                diabetes4$stgpt3[has_stage2]

diabetes4$totpt <- as.numeric(diabetes4$exit - diabetes4$date + 1)

diabetes4$stgpt1 <- diabetes4$totpt - (diabetes4$stgpt2 + diabetes4$stgpt3)

# Order, if necessary.
# diabetes4 <- diabetes4[order(diabetes4$rootlpno, diabetes4$date),]

# View the data as shown in the workbook.
View(diabetes4[184:193, 3:14])
```

R_Solutions - RStudio Source Editor

diabetes5[184:193, 3:14]

Filter

	sex	age	type	fileseq	stage2	stage3	exit	dead	stgpt1	stgpt2	stgpt3	totpt
184	2	69	2	184	1992-05-20	1996-12-23	1999-12-31	0	0	1678	1104	2782
185	2	68	1	185	NA	NA	1991-11-17	1	681	0	0	681
186	2	62	1	186	1992-02-10	1992-06-29	1999-12-31	0	681 660	140	2742	3542
187	2	74	3	187	NA	1992-03-31	1999-06-26	1	219	0	2644	2863
188	2	50	2	188	1993-03-16	1995-05-24	1999-12-31	0	0	799	1683	2482
189	1	30	2	189	1994-04-02	NA	1999-12-31	0	0	2100	0	2100
190	1	62	2	190	NA	1990-10-23	1996-05-18	1	0	0	2035	2035
191	2	71	1	191	1998-11-01	NA	1999-12-31	0	214	426	0	640
192	2	29	3	192	NA	1993-03-07	1999-12-31	0	265	0	2491	2756
193	2	33	2	193	1996-07-03	1996-09-09	1999-12-31	0	0	68	1209	1277

Showing 1 to 10 of 10 entries, 12 total columns

```
# Step 9: Handle negative person-times.
# Find rows with negative PT.
negative_pt <- which(diabetes4$stgpt1<0 | diabetes4$stgpt2<0 |
                     diabetes4$stgpt3<0 | diabetes4$totpt<0)
diabetes4$dead[negative_pt] <- 0
diabetes4$exit[negative_pt] <- as.Date("1999-12-31")

# Re-run the person-time code.
diabetes4$stgpt1 <- 0
diabetes4$stgpt2 <- 0
diabetes4$stgpt3 <- 0
diabetes4$totpt <- 0

has_stage3 <- !is.na(diabetes4$stage3)
diabetes4$stgpt3[has_stage3] <- diabetes4$exit[has_stage3] -
                                diabetes4$stage3[has_stage3] + 1
```

```
has_stage2 <- !is.na(diabetes4$stage2)
diabetes4$stgpt2[has_stage2] <- diabetes4$exit[has_stage2] -
                                diabetes4$stage2[has_stage2] + 1 -
                                diabetes4$stgpt3[has_stage2]

diabetes4$totpt <- as.numeric(diabetes4$exit - diabetes4$date + 1)

diabetes4$stgpt1 <- diabetes4$totpt - (diabetes4$stgpt2 + diabetes4$stgpt3)

diabetes5 <- diabetes4

# Save the data.
save(diabetes5, file='data/diabetes5.RData')


# Step 10: Descriptive stats for the stages.
table <- describe(diabetes5[, c('stgpt1', 'stgpt2', 'stgpt3', 'totpt')])
# Replace the vars (number) with the row name.
table$vars <- row.names(table)
# Display in the viewer panel.
flextable(table)
```

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
stgpt1	10,675	706	858	390	550	578	0	3,652	3,652	1.34	1.094	8.3
stgpt2	10,675	665	906	44	502	65	0	3,493	3,493	1.14	-0.063	8.8
stgpt3	10,675	297	717	0	89	0	0	3,441	3,441	2.47	4.973	6.9
totpt	10,675	1,668	1,040	1,593	1,643	1,320	1	3,652	3,651	0.15	-1.128	10.1

```
# Step 11: Estimating point prevalence.
prevalence_date <- '1994-06-30'
diabetes5$prev94 <- 0
diabetes5$prev94[diabetes5$date <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 1
diabetes5$prev94[diabetes5$stage3 > 0 &
                 diabetes5$stage3 <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 3
diabetes5$prev94[diabetes5$prev94 != 3 & diabetes5$stage2 > 0 &
                 diabetes5$stage2 <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 2

prevalence_date <- '1999-06-30'
diabetes5$prev99 <- 0
diabetes5$prev99[diabetes5$date <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 1
diabetes5$prev99[diabetes5$stage3 > 0 & diabetes5$stage3 <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 3
diabetes5$prev99[diabetes5$prev99 != 3 & diabetes5$stage2 > 0 &
                 diabetes5$stage2 <= prevalence_date &
                 diabetes5$exit >= prevalence_date] <- 2

table(diabetes5$prev94)
```

```
table(diabetes5$prev99)

# Display the tables with flextable.
table <- as.data.frame(table(diabetes5$prev94))
colnames(table) <- c('Prevalence', colnames(table)[2:length(table)])
table <- flextable(table)
table <- set_caption(table, 'Prevalence 1994')
table
table <- as.data.frame(table(diabetes5$prev99))
colnames(table) <- c('Prevalence', colnames(table)[2:length(table)])
table <- flextable(table)
table <- set_caption(table, 'Prevalence 1999')
table
```

Prevalence 1994

Prevalence	Freq
0	5,883
1	1,812
2	2,086
3	894

Prevalence 1999

Prevalence	Freq
0	2,540
1	3,051
2	3,465
3	1,619

```
# Step 12: Utilisation rate of primary medical care by disease severity.
load('data/MBSdata2.RData')
MBSdata2$gpvisit <- 0

# The codes that indicate a GP visit.
GP_visits <- c(seq(1, 51), seq(160, 164), seq(170, 173), seq(193, 199),
               seq(601, 602), seq(700, 712), seq(720, 779), seq(900, 903),
               seq(2501, 2509), seq(2517, 2526), seq(2546, 2559),
               seq(2574, 2578), seq(2721, 2727), seq(5000, 5267))

# Tag rows with any of these codes.
MBSdata2$gpvisit[MBSdata2$mbsitem %in% GP_visits] <- 1

# gpvisit is 1/0, summing is a count (expecting 745,336).
sum(MBSdata2$gpvisit)


# Step 13:
MBSdata3 <- merge(MBSdata2,
                  diabetes5[, c('rootlpno', 'date', 'stage2', 'stage3', 'exit')],
                  all.x=T, by='rootlpno')


# Save the data.
save(MBSdata3, file='data/MBSdata3.RData')
```

```
# Step 14: Stage the visits.
MBSdata4 <- MBSdata3

MBSdata4$stgpvis[MBSdata4$gpvisit == 1] <- 0

MBSdata4$stgpvis[MBSdata4$gpvisit == 1 & MBSdata4$date <= MBSdata4$servdate &
                 MBSdata4$exit >= MBSdata4$servdate] <- 1

MBSdata4$stgpvis[MBSdata4$gpvisit == 1 & MBSdata4$stage3 > 0 &
                 MBSdata4$stage3 <= MBSdata4$servdate &
                 MBSdata4$exit >= MBSdata4$servdate] <- 3

MBSdata4$stgpvis[MBSdata4$gpvisit == 1 & MBSdata4$stgpvis != 3 &
                 MBSdata4$stage2 > 0 & MBSdata4$stage2 <= MBSdata4$servdate &
                 MBSdata4$exit >= MBSdata4$servdate] <- 2

table(MBSdata4$stgpvis)

# Save the data.
save(MBSdata4, file='data/MBSdata4.RData')


# Step 15: Aggregate the GP visit counts in the stages to the first row.
DT <- data.table(MBSdata4)
DT <- DT[order(rootlpno, fileseq)]
setkey(DT, rootlpno)

# Get the staged rows.
GP_visits_stage1 <- MBSdata4[which(MBSdata4$stgpvis == 1), c('rootlpno', 'fileseq')]
GP_visits_stage2 <- MBSdata4[which(MBSdata4$stgpvis == 2), c('rootlpno', 'fileseq')]
GP_visits_stage3 <- MBSdata4[which(MBSdata4$stgpvis == 3), c('rootlpno', 'fileseq')]

# Make a data table for each and set the key for the joins.
DT_GP_stage1 <- data.table(GP_visits_stage1)
setkey(DT_GP_stage1, rootlpno)
DT_GP_stage2 <- data.table(GP_visits_stage2)
setkey(DT_GP_stage2, rootlpno)
DT_GP_stage3 <- data.table(GP_visits_stage3)
setkey(DT_GP_stage3, rootlpno)

# Count total matched rows per person, then remove fileseq and select unique
# rootlpnos for each stage.
DT_GP_stage1 <- DT_GP_stage1[, gpvisit1 := .N, 'rootlpno']
DT_GP_stage1 <- unique(DT_GP_stage1[, fileseq := NULL])
DT_GP_stage2 <- DT_GP_stage2[, gpvisit2 := .N, 'rootlpno']
DT_GP_stage2 <- unique(DT_GP_stage2[, fileseq := NULL])
DT_GP_stage3 <- DT_GP_stage3[, gpvisit3 := .N, 'rootlpno']
DT_GP_stage3 <- unique(DT_GP_stage3[, fileseq := NULL])

# Outer join GP visit stage tables back onto the main DT.
DT <- DT_GP_stage3[DT, ]
DT <- DT_GP_stage2[DT, ]
DT <- DT_GP_stage1[DT, ]
```

```
# cut down and save back to data.frame
MBSgpst <- as.data.frame(DT[morbseq == 1,
                        c('rootlpno', 'gpvisit1', 'gpvisit2', 'gpvisit3'),
                        with=FALSE])
MBSgpst[is.na(MBSgpst)] <- 0

# Save the data.
save(MBSgpst, file='data/MBSgpst.RData')


# Step 16: Merge with the diabetes5 platform file.
diabetes6 <- merge(diabetes5, MBSgpst, all.x=T, by='rootlpno')
# Check for NA's which happen to be NA for all three visit codes.
sum(is.na(diabetes6$gpvisit1) & is.na(diabetes6$gpvisit2) &
    is.na(diabetes6$gpvisit3))
# Replace NA with zero for all 3 variables.
diabetes6$gpvisit1[is.na(diabetes6$gpvisit1)] <-
    diabetes6$gpvisit2[is.na(diabetes6$gpvisit2)] <-
    diabetes6$gpvisit3[is.na(diabetes6$gpvisit3)] <- 0

# View the data as shown in the workbook.
View(diabetes6[21:35, c(6:14, 17:19)])
```

R_Solutions - RStudio Source Editor

diabetes6[21:35, c(6:14, 17:19)]

Filter

	fileseq	stage2	stage3	exit	dead	stgpt1	stgpt2	stgpt3	totpt	gpvisit1	gpvisit2	gpvisit3
21	21	1994-04-22	NA	1999-12-31	0	0	2080	0	2080	0	84	0
22	22	1997-12-02	NA	1999-12-31	0	0	760	0	760	0	23	0
23	23	NA	1993-03-29	1999-12-31	0	0	0	2469	2469	0	0	129
24	24	1993-03-01	NA	1999-12-31	0	0	2497	0	2497	0	29	0
25	25	NA	NA	1999-12-31	0	3440	0	0	3440	30	0	0
26	26	1994-07-23	1995-02-21	1999-12-31	0	1271	213	1775	3259	25	8	46
27	27	1998-02-24	NA	1999-12-31	0	927	676	0	1603	21	4	0
28	28	1999-12-03	NA	1999-12-31	0	1065	29	0	1094	30	3	0
29	29	NA	NA	1996-01-18	1	1	0	0	1	0	0	0
30	30	NA	NA	1995-10-11	1	674	0	0	674	19	0	0
31	31	1992-12-30	1994-10-13	1999-12-31	0	932	652	1906	3490	27	12	90
32	32	1992-03-07	1997-12-29	1998-04-11	1	56	2123	104	2283	0	147	11
33	33	NA	NA	1999-12-31	0	400	0	0	400	17	0	0
34	34	1996-09-25	NA	1999-12-31	0	0	1193	0	1193	0	0	0
35	35	1998-08-06	1998-11-27	1999-12-31	0	2856	113	400	3369	27	3	11

Showing 1 to 15 of 15 entries, 12 total columns

```
# Save the data.
save(diabetes6, file='data/diabetes6.RData')


# Step 17: Check the descriptive statistics.
table <- describe(diabetes6[, c('gpvisit1', 'gpvisit2', 'gpvisit3')])
# Replace the vars (number) with the row name.
table$vars <- row.names(table)
```

```
# Display in the viewer panel.
flextable(table)
```

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
gpvisit1	10,675	14.2	23	4	9.1	5.9	0	479	479	3.3	24.2	0.22
gpvisit2	10,675	16.7	29	0	9.9	0.0	0	336	336	2.6	8.9	0.28
gpvisit3	10,675	7.2	22	0	1.3	0.0	0	414	414	5.2	47.0	0.21

TRAINING SESSION 2: TIDYVERSE-R SYNTAX SOLUTIONS

```
## Preparation
# Start a new R session by opening the project.

# Load the libraries used in the exercise.
library(dplyr)
library(magrittr)
library(psych)
library(summarytools)   # Use 'freq' in step 11.

################################## Exercise 2 ##################################

#### Steps 1-3: Identify transition events to stage 2 in PBSdata file and load
####            onto diabetes file

#### Step 1: Load the PBSdata2 file
load('data/PBSdata2.RData')


#### Step 2: Create sequence variables
PBSdata3 <- PBSdata2 %>%
    arrange(rootlpno, morbseq) %>%
    group_by(rootlpno) %>%
    # Create the ohga sequence.
    # Find the smallest morbseq which has a diabmed flag.
    mutate(index=ifelse(diabmed==1, morbseq, NA)) %>%
    mutate(index=min(index, na.rm=TRUE)) %>%
    # For this morbseq onwards, create a sequence from 1 onwards.
    mutate(ohgaseq=ifelse(morbseq>=index, row_number()-index+1, 0)) %>%
    # Create the insp sequence.
    mutate(index=ifelse(diabmed==2, morbseq, NA)) %>%
    mutate(index=min(index, na.rm=TRUE)) %>%
    mutate(inspseq=ifelse(morbseq>=index, row_number()-index+1, 0)) %>%
    select(-index) %>%
    ungroup()

save(PBSdata3, file='data/PBSdata3.RData')


#### Step 3: Cutdown the file with only OHGA not INSP
PBSohga <- PBSdata3 %>%
    filter(ohgaseq==1 & inspseq==0) %>%
    select(rootlpno, dispdate)

save(PBSohga, file='data/PBSohga.RData')


#### Steps 4-6: Identify transition events to stage 3 in PBSdata file and load
####            onto diabetes file

#### Step 4: Merge dispdate (stage2 date) into the diabetes data
# See the code for a note about merge() - if you find you have too few records,
# check to see if you have done a merge and left out some rows from one or other
```

```
# of the merged data frames.
load('data/diabetes.RData')
diabetes2 <- diabetes %>%
    # Use all.x=TRUE (merge joins rows that are in BOTH x and y (a natural join
    # which is a special case of an inner join); to keep the x records (left
    # outer join) use all.x=TRUE and all.y=TRUE for a right outer join and
    # all=TRUE gives a full outer join).
    merge(PBSohga, all.x=TRUE) %>%
    rename(stage2=dispdate)

save(diabetes2, file='data/diabetes2.RData')


#### Step 5: Cutdown PBSdata3
# load('data/PBSdata3.RData')
PBSinsp <- PBSdata3 %>%
    filter(inspseq==1) %>%
    select(rootlpno, dispdate)

save(PBSinsp, file='data/PBSinsp.RData')


#### Step 6: Merge dispdate (stage3 date) into the diabetes data
diabetes3 <- diabetes2 %>%
    merge(PBSinsp, all.x=TRUE) %>%
    rename(stage3=dispdate)

save(diabetes3, file='data/diabetes3.RData')


#### Step 7: Load deaths from Dthdata onto diabetes file and set exit dates
load('data/Dthdata2.RData')
diabetes4 <- diabetes3 %>%
    merge((Dthdata2 %>% select(rootlpno, dthdate)), all.x=TRUE) %>%
    rename(exit=dthdate) %>%
    mutate(dead=ifelse(!is.na(exit), 1, 0)) %>%
    mutate(exit=as.Date(ifelse(dead==0 | exit >= as.Date("1999-12-31"),
                               "1999-12-31", as.character(exit))))

save(diabetes4, file='data/diabetes4.RData')


#### Steps 8-10: Derive person-time partitioned by stage

#### Step 8: Calculate the person-time variables for the stages
diabetes5 <- diabetes4 %>%
    mutate(stgpt3=ifelse(!is.na(stage3), exit - stage3 + 1, 0)) %>%
    mutate(stgpt2=ifelse(!is.na(stage2), exit - stage2 + 1 - stgpt3, 0)) %>%
    mutate(totpt=as.numeric(exit - date + 1)) %>%
    mutate(stgpt1=totpt - stgpt2 - stgpt3) %>%
    select(rootlpno, date, sex, age, type, fileseq, stage2, stage3, exit, dead,
           stgpt1, stgpt2, stgpt3, totpt) %>%
    arrange(rootlpno, date)

View(diabetes5[184:193, 3:14])
```

R_Solutions - RStudio Source Editor

diabetes5[184:193, 3:14]

Filter

	sex	age	type	fileseq	stage2	stage3	exit	dead	stgpt1	stgpt2	stgpt3	totpt
184	2	69	2	184	1992-05-20	1996-12-23	1999-12-31	0	0	1678	1104	2782
185	2	68	1	185	NA	NA	1991-11-17	1	681	0	0	681
186	2	62	1	186	1992-02-10	1992-06-29	1999-12-31	0	660	140	2742	3542
187	2	74	3	187	NA	1992-03-31	1999-06-26	1	219	0	2644	2863
188	2	50	2	188	1993-03-16	1995-05-24	1999-12-31	0	0	799	1683	2482
189	1	30	2	189	1994-04-02	NA	1999-12-31	0	0	2100	0	2100
190	1	62	2	190	NA	1990-10-23	1996-05-18	1	0	0	2035	2035
191	2	71	1	191	1998-11-01	NA	1999-12-31	0	214	426	0	640
192	2	29	3	192	NA	1993-03-07	1999-12-31	0	265	0	2491	2756
193	2	33	2	193	1996-07-03	1996-09-09	1999-12-31	0	0	68	1209	1277

Showing 1 to 10 of 10 entries, 12 total columns

```
#### Step 9: Check for negative person-time values
# This can be done by a lot of cut-and-paste.
cat('There is/are ', (diabetes5 %>% filter(stgpt1<0) %>% count())[[1]],
    ' record(s) with stgpt1 < 0.\n', sep='')
cat('There is/are ', (diabetes5 %>% filter(stgpt2<0) %>% count())[[1]],
    ' record(s) with stgpt2 < 0.\n', sep='')
cat('There is/are ', (diabetes5 %>% filter(stgpt3<0) %>% count())[[1]],
    ' record(s) with stgpt3 < 0.\n', sep='')
cat('There is/are ', (diabetes5 %>% filter(totpt<0) %>% count())[[1]],
    ' record(s) with totpt < 0.\n', sep='')

# Or by defining a function and using the rlang functionality (!! - see defusing).
report_negative <- function(frame, name) {
    cat('There is/are ', (frame %>% filter(!!name<0) %>% count())[[1]],
        ' record(s) with ', name, ' < 0.\n', sep='')
}
report_negative(diabetes5, expr(stgpt1))
report_negative(diabetes5, expr(stgpt2))
report_negative(diabetes5, expr(stgpt3))
report_negative(diabetes5, expr(totpt))

# And for an expression such as totpt<0|stgpt1<0 etc.
cat('There is/are ',
    (diabetes5 %>% filter(totpt<0|stgpt1<0|stgpt2<0|stgpt3<0) %>% count())[[1]],
    ' record(s) with one or more of these < 0.\n', sep='')

# We can pass that expression in and evaluate it.
report_negatives <- function(frame, expression) {
    cat('There is/are ', (frame %>% filter(!!expression) %>% count())[[1]],
        ' record(s) with (', deparse(expression),').\n', sep='')
}
report_negatives(diabetes5, expr(totpt<0|stgpt1<0|stgpt2<0|stgpt3<0))

# Now fix the negative person-times.
```

```
diabetes5 <- diabetes5 %>%
    mutate(dead=ifelse(totpt<0 | stgpt1<0 | stgpt2<0 | stgpt3<0, 0, dead)) %>%
    mutate(exit=as.Date(
        ifelse(totpt<0 | stgpt1<0 | stgpt2<0 | stgpt3<0,
               as.Date("1999-12-31", origin="1970-01-01"), exit),
        origin="1970-01-01"))

diabetes5 <- diabetes5 %>%
    mutate(stgpt3=ifelse(!is.na(stage3), exit - stage3 + 1, 0)) %>%
    mutate(stgpt2=ifelse(!is.na(stage2), exit - stage2 + 1 - stgpt3, 0)) %>%
    mutate(totpt=as.numeric(exit - date + 1)) %>%
    mutate(stgpt1=totpt - stgpt2 - stgpt3) %>%
    select(rootlpno, date, sex, age, type, fileseq, stage2, stage3, exit, dead,
           stgpt1, stgpt2, stgpt3, totpt) %>%
    arrange(rootlpno, date)

# And check the results.
head(diabetes5[184:193,], 10)

report_negatives(diabetes5, expr(totpt<0|stgpt1<0|stgpt2<0|stgpt3<0))

save(diabetes5, file='data/diabetes5.RData')


#### Step 10: Check the descriptive statistics
describe(diabetes5 %>% select(stgpt1, stgpt2, stgpt3, totpt))
```

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
stgpt1	10,675	706	858	390	550	578	0	3,652	3,652	1.34	1.094	8.3
stgpt2	10,675	665	906	44	502	65	0	3,493	3,493	1.14	-0.063	8.8
stgpt3	10,675	297	717	0	89	0	0	3,441	3,441	2.47	4.973	6.9
totpt	10,675	1,668	1,040	1,593	1,643	1,320	1	3,652	3,651	0.15	-1.128	10.1

```
### Step 11: Estimate point prevalence in 1994 and 1999
date_prev94 <- '1994-06-30'
diabetes5_94 <- diabetes5 %>%
    mutate(prev94=ifelse(date<=date_prev94 & exit>=date_prev94, 1, 0)) %>%
    mutate(prev94=ifelse(!is.na(stage3) & stage3<=date_prev94 &
                         exit>=date_prev94, 3, prev94)) %>%
    mutate(prev94=ifelse(prev94!=3 & !is.na(stage2) & stage2<=date_prev94 &
                         exit>=date_prev94, 2, prev94))

date_prev99 <- '1999-06-30'
diabetes5_99 <- diabetes5 %>%
    mutate(prev99=ifelse(date<=date_prev99 & exit>=date_prev99, 1, 0)) %>%
    mutate(prev99=ifelse(!is.na(stage3) & stage3<=date_prev99 &
                         exit>=date_prev99, 3, prev99)) %>%
    mutate(prev99=ifelse(prev99!=3 & !is.na(stage2) & stage2<=date_prev99 &
                         exit>=date_prev99, 2, prev99))

table(diabetes5_94$prev94)
table(diabetes5_99$prev99)
```

```
freq(diabetes5_94$prev94)
```

Prevalence 1994

Prevalence	Freq
0	5,883
1	1,812
2	2,086
3	894

Prevalence 1999

Prevalence	Freq
0	2,540
1	3,051
2	3,465
3	1,619

```
#### Steps 12-14: Load transition dates from diabetes file onto MBSdata file and
####              identify/classify GP visits

#### Step 12: Create the gpvisit variable
load('data/MBSdata2.RData')
gp_codes <- c(seq(1, 51), seq(160, 164), seq(170, 173), seq(193, 199),
              seq(601, 602), seq(700, 712), seq(720, 779), seq(900, 903),
              seq(2501, 2509), seq(2517, 2526), seq(2546, 2559),
              seq(2574, 2578), seq(2721, 2727), seq(5000, 5267))
MBSdata3 <- MBSdata2 %>%
    ungroup() %>%
    mutate(gpvisit=ifelse(mbsitem %in% gp_codes, 1, 0))


#### Step 13: Merge variables from the platform file (diabetes6)
MBSdata3 <- MBSdata3 %>%
    merge((diabetes5 %>% select(rootlpno, date, stage2, stage3, exit)),
          by='rootlpno', all.x=TRUE)

save(MBSdata3, file='data/MBSdata3.RData')


#### Step 14: Create visit stage variable
# Remember to check for NA in date fields explicitly or R will set those rows to
# NA rather than use the FALSE (second) option.
MBSdata4 <- MBSdata3 %>%
    mutate(stgpvis=ifelse(gpvisit==1, 0, NA)) %>%
    mutate(stgpvis=ifelse(gpvisit==1 & !is.na(servdate) & !is.na(date) &
                          !is.na(exit) & date<=servdate & exit>=servdate,
                          1, stgpvis)) %>%
    mutate(stgpvis=ifelse(gpvisit==1 & !is.na(servdate) & !is.na(exit) &
                          !is.na(stage3) & stage3<=servdate & exit>=servdate,
                          3, stgpvis)) %>%
    mutate(stgpvis=ifelse(gpvisit==1 & !is.na(servdate) & !is.na(exit) &
                          stgpvis!=3 & !is.na(stage2) & stage2<=servdate &
                          exit>=servdate,
                          2, stgpvis))
```

```
table(MBSdata4$stgpvis)

save(MBSdata4, file='data/MBSdata4.RData')


#### Steps 15-16: Accumulate GP visits by stage onto index records in MBSdata
####              and load onto diabetes file.

#### Step 15: Create counts of the number of GP visits in each stage
MBSgpst <- MBSdata4 %>%
    select(rootlpno, stgpvis) %>%
    arrange(rootlpno, stgpvis) %>%
    group_by(rootlpno, stgpvis) %>%
    # Get the counts of the GP visits at each stage into the first row of each
    # of the rootlpno/stgpvis groups.
    mutate(gpvisit1=ifelse(stgpvis==1 & row_number()==1, n(), 0)) %>%
    mutate(gpvisit2=ifelse(stgpvis==2 & row_number()==1, n(), 0)) %>%
    mutate(gpvisit3=ifelse(stgpvis==3 & row_number()==1, n(), 0)) %>%
    # Keep just the first row with the counts.
    filter(row_number()==1) %>%
    ungroup()

MBSgpst <- MBSgpst %>%
    group_by(rootlpno) %>%
    # Collapse the maximum values from the stgpvis records to the first record
    # and preventing the NA's being included in the maximum if no value exists
    # i.e. set to zero.
    mutate(gpvisit1=ifelse(is.na(gpvisit1), 0, max(gpvisit1, na.rm=TRUE))) %>%
    mutate(gpvisit2=ifelse(is.na(gpvisit2), 0, max(gpvisit2, na.rm=TRUE))) %>%
    mutate(gpvisit3=ifelse(is.na(gpvisit3), 0, max(gpvisit3, na.rm=TRUE))) %>%
    filter(row_number()==1) %>%
    select(-stgpvis) %>%
    ungroup() %>%
    distinct()

save(MBSgpst, file='data/MBSgpst.RData')


#### Step 16: Load the gpvisit variables into the platform file (diabetes5)
load('data/diabetes5.RData')

diabetes6 <- diabetes5 %>% merge(MBSgpst, by='rootlpno', all.x=TRUE)

# The merge has left 47 records in diabetes5 which were unspecified in MBSgpst.
sum(is.na(diabetes6$gpvisit1)&is.na(diabetes6$gpvisit2)&is.na(diabetes6$gpvisit3))

diabetes6 <- diabetes6 %>%
    mutate(gpvisit1=ifelse(is.na(gpvisit1), 0, gpvisit1),
           gpvisit2=ifelse(is.na(gpvisit2), 0, gpvisit2),
           gpvisit3=ifelse(is.na(gpvisit3), 0, gpvisit3))

View(diabetes6[21:35, 6:17])

save(diabetes6, file='data/diabetes6.RData')
```

R_Solutions - RStudio Source Editor

diabetes6[21:35, 6:17]

Filter

	fileseq	stage2	stage3	exit	dead	stgpt1	stgpt2	stgpt3	totpt	gpvisit1	gpvisit2	gpvisit3
21	21	1994-04-22	NA	1999-12-31	0	0	2080	0	2080	0	84	0
22	22	1997-12-02	NA	1999-12-31	0	0	760	0	760	0	23	0
23	23	NA	1993-03-29	1999-12-31	0	0	0	2469	2469	0	0	129
24	24	1993-03-01	NA	1999-12-31	0	0	2497	0	2497	0	29	0
25	25	NA	NA	1999-12-31	0	3440	0	0	3440	30	0	0
26	26	1994-07-23	1995-02-21	1999-12-31	0	1271	213	1775	3259	25	8	46
27	27	1998-02-24	NA	1999-12-31	0	927	676	0	1603	21	4	0
28	28	1999-12-03	NA	1999-12-31	0	1065	29	0	1094	30	3	0
29	29	NA	NA	1996-01-18	1	1	0	0	1	0	0	0
30	30	NA	NA	1995-10-11	1	674	0	0	674	19	0	0
31	31	1992-12-30	1994-10-13	1999-12-31	0	932	652	1906	3490	27	12	90
32	32	1992-03-07	1997-12-29	1998-04-11	1	56	2123	104	2283	0	147	11
33	33	NA	NA	1999-12-31	0	400	0	0	400	17	0	0
34	34	1996-09-25	NA	1999-12-31	0	0	1193	0	1193	0	0	0
35	35	1998-08-06	1998-11-27	1999-12-31	0	2856	113	400	3369	27	3	11

Showing 1 to 15 of 15 entries, 12 total columns

```
#### Step 17: Derive GP visit counts and rates partitioned by stage
# The results are not quite the same as in the workbook where n=10675 (47 out)!
describe(diabetes6 %>% select(gpvisit1, gpvisit2, gpvisit3))
```

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
gpvisit1	10,675	14.2	23	4	9.1	5.9	0	479	479	3.3	24.2	0.22
gpvisit2	10,675	16.7	29	0	9.9	0.0	0	336	336	2.6	8.9	0.28
gpvisit3	10,675	7.2	22	0	1.3	0.0	0	414	414	5.2	47.0	0.21

TRAINING SESSION 3: R SYNTAX SOLUTIONS

```
################################################################################
# AALHD DAY 3
# R Syntax solutions
#
# Dr Pete Arnold based on code by Dr Joanne Demmler
# Updated March 2021
################################################################################
# Hints about what has been used in this solution.
#
################################################################################

library(data.table)
library(survival)       # For the survival analysis at step 16.
library(flextable)      # Used to produce tables in the viewer panel (images).

################################### Exercise 3 ###################################

# Step 1: Open the diabetes complications file.
load('data/diabcomp.RData')


# Step 2: Open the relevant platform file and load the diabetes complications
#         data in the EOR.
load('data/diabetes6.RData')

diabetes7 <- merge(diabetes6, diabcomp, all.x=TRUE, by='rootlpno')

# Or, perhaps, we could have added the flag at step 1 and just merged them in.
diabetes7$complic <- 0
diabetes7$complic[!is.na(diabetes7$compdate)] <- 1

# Check the frequencies (4,479 @0 and 6,196 @1, total 10,675).
table(diabetes7$complic)

# Save the data.
save(diabetes7, file='data/diabetes7.RData')


# Step 3: Load the HMDS data (#2).
load('data/HMDSdata2.RData')


# Step 4: Load the 'date' from diabetes7 onto the end of the HMDS data.
HMDSdata3 <- merge(HMDSdata2, diabetes7[, c('rootlpno', 'date')], all.x=TRUE)

# Order the data by fileseq.
HMDSdata3 <- HMDSdata3[order(HMDSdata3$fileseq), ]
```

```
# Step 5: Tag records that mention pregnancy, childbirth or puerperium.
cond <- c(format(seq(630.00, 677.99, 0.01), digits=6),
          gsub(' ', '0', paste('O', format(seq(00.0, 99.9, 0.01), digits=5),
                               sep='')))

HMDSdata3$pregnant <- 0

# Create a list of all rows that match one of these codes.
i <- 9
rows_with_code <- 0
while (i <= 29){
    rows_with_code <- append(rows_with_code, which(HMDSdata3[, i] %in% cond))
    i <- i + 1
}

# Set the flag for the matching rows.
HMDSdata3$pregnant[rows_with_code] <- 1


# Step 6: Create the pregseq variable for pregnancies after diabetes diagnosis.
HMDSdata3 <- HMDSdata3[order(HMDSdata3$rootlpno, HMDSdata3$morbseq), ]

DT <- data.table(HMDSdata3)
# setkeyv(DT, c("rootlpno", "morbseq"))
DT <- DT[pregnant == 1 & admdate >= date, pregseq := 1:.N, by=rootlpno]
HMDSdata3 <- as.data.frame(DT)

# View the data as shown in the workbook.
View(HMDSdata3[477:483, 46:51])
```

R_Solutions - RStudio Source Editor

HMDSdata3[477:483, 46:51]

Filter

	fileseq	morbseq	condseq	date	pregnant	pregseq
477	477	1	NA	1996-07-09	1	NA
478	478	2	NA	1996-07-09	1	1
479	479	3	NA	1996-07-09	0	NA
480	480	4	NA	1996-07-09	1	2
481	481	5	NA	1996-07-09	0	NA
482	482	6	NA	1996-07-09	1	3
483	483	1	1	1995-10-25	0	NA

Showing 1 to 7 of 7 entries, 6 total columns

```
# Save the data.
save(HMDSdata3, file='data/HMDSdata3.RData')
```

```
# Step 7: Create a cut down file.
names(HMDSdata3)[5] <- 'pregdate'
HMDSpreg <- HMDSdata3[HMDSdata3$pregseq == 1 & !is.na(HMDSdata3$pregseq), c('rootlpno',
'pregdate')]

# Check that we have 213 rows.
nrow(HMDSpreg)

# Save the data.
save(HMDSpreg, file='data/HMDSpreg.RData')


# Step 8: Load the pregdate onto the EOR of diabetes7.
diabetes8 <- merge(diabetes7, HMDSpreg, all.x=TRUE)

# Or, perhaps, we could have added the flag at step 7 and just merged them in.
diabetes8$pregexp <- 0
diabetes8$pregexp[!is.na(diabetes8$pregdate)] <- 1

# Save the data.
save(diabetes8, file='data/diabetes8.RData')


# Step 9: Restrict to women of 'reproductive age'.
rpfdiab <- diabetes8[!is.na(diabetes8$sex) & diabetes8$sex == '2' &
                     diabetes8$age %in% seq(15,39), ]

# Check that we have 716 rows.
nrow(rpfdiab)

# Save the data.
save(rpfdiab, file='data/rpfdiab.RData')


# Step 10: Create exposure/non-exposure variables.
# Order if needed.
# rpfdiab <- rpfdiab[order(rpfdiab$rootlpno, rpfdiab$date),]
rpfexp <- rpfdiab[rpfdiab$pregexp == 1, ]

# Define the random number seed (to make the results repeatable).
set.seed(2000000)

# Assign a random number to all rows.
rpfexp$randsort <- runif(nrow(rpfexp), min=0, max=1)

# Order them by this random number.
rpfexp <- rpfexp[order(rpfexp$randsort), ]

# Create a randseq variable counting from 1 to 200.
rpfexp$randseq <- seq_len(nrow(rpfexp))

# Calculate the time from the inaugural index date and the first pregnancy.
rpfexp$timepreg <- rpfexp$pregdate - rpfexp$date
```

```
# View the data as shown in the workbook.
View(rpfexp[1:15, 20:26])
```

R_Solutions - RStudio Source Editor

rpfexp[1:15, 20:26]

Filter

	compdate	complic	pregdate	pregexp	randsort	randseq	timepreg
4306	1999-07-14	1	1999-07-11	1	0.005695261	1	255
4399	NA	0	1994-05-29	1	0.014691459	2	204
6280	NA	0	1996-03-13	1	0.016302094	3	0
7743	1994-08-15	1	1992-06-07	1	0.017216894	4	820
10463	NA	0	1990-07-15	1	0.019267581	5	147
5705	NA	0	1999-04-12	1	0.022334185	6	0
7503	NA	0	1998-10-14	1	0.026472706	7	122
6216	1998-11-21	1	1991-12-21	1	0.028545107	8	29
6061	NA	0	1994-02-12	1	0.030311104	9	48
77	NA	0	1996-07-13	1	0.031194580	10	4
4171	NA	0	1994-11-16	1	0.032503811	11	43
2655	NA	0	1993-04-13	1	0.035442332	12	911
5751	1990-07-24	1	1991-01-23	1	0.045416619	13	305
8428	NA	0	1995-01-12	1	0.055990095	14	99
7492	NA	0	1999-12-08	1	0.065924422	15	48

Showing 1 to 15 of 15 entries, 7 total columns

```
# Cut down the file.
rpfexp<- rpfexp[, c('rootlpno', 'randseq', 'timepreg')]

# Save the data.
save(rpfexp, file='data/rpfexp.RData')


# Step 11: Tag the non-exposed women.
rpfnon <- rpfdiab[rpfdiab$pregexp == 0, ]

# Define the random number seed (to make the results repeatable).
set.seed(3000000)

# Assign a random number to all rows.
rpfnon$randsort <- runif(nrow(rpfnon), min=0, max=1)

# Order them by this random number.
rpfnon <- rpfnon[order(rpfnon$randsort), ]
```

```
# Create a randseq variable counting from 1 to 200 (as many times as necessary).
# i.e. twice with the remainder up to 116.
rpfnon$randseq <- rep(1:200, 3)[1:nrow(rpfnon)]


# Step 12: Sort by randseq and load the timepreg value.
rpfnon <- rpfnon[order(rpfnon$randseq), ]

rpfnon <- merge(rpfnon, rpfexp[, c('randseq', 'timepreg')], by='randseq',
                all.x=TRUE)

# Now order them by rootlpno.
rpfnon <- rpfnon[order(rpfnon$rootlpno), ]

# Cut down.
rpfnon <- rpfnon[, c('rootlpno', 'timepreg')]

# Save the data.
save(rpfnon, file='data/rpfnon.RData')


# Step 13: Load the virtual time to pregnancy for the non-exposed women.
rpfdiab2 <- merge(rpfdiab, rpfnon, all.x=TRUE)

# Calculate the real timepreg for the exposed women.
rpfdiab2$timepreg[rpfdiab2$pregexp == 1] <-
    rpfdiab2$pregdate[rpfdiab2$pregexp == 1] - rpfdiab2$date[rpfdiab2$pregexp == 1]

# Compare the distributions.
tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, length)
tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, mean)
tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, sd)

# And, if you want to make a nice-looking table:
df <- data.frame(tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, length))
colnames(df) <- 'Frequency'
df$Mean <- tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, mean)
df$SD <- tapply(rpfdiab2$timepreg, rpfdiab2$pregexp, sd)
df$Pregnancy <- c('0', '1')
df <- df[, c(4, 1:3)]
colformat_num(flextable(df), digits=5)
```

Pregnancy	Frequency	Mean	SD
0	516	449.17	616.65
1	200	458.81	623.53

```
# Re-run from Q10 if your values are diverging too much.
```

```
# Flag non-exposed women with virtual times to pregnancy exceeding their total
# follow-up time.
rpfdiab2$flag <- 0
rpfdiab2$flag[rpfdiab2$exit - rpfdiab2$date < rpfdiab2$timepreg] <- 1

# Check the number of women flagged (56 when run previously).
xtabs(~flag+pregexp, data=rpfdiab2)

# Remove rows that are flagged and the flag column (25).
rpfdiab2 <- rpfdiab2[rpfdiab2$flag == 0, 1:24]

# Save the data.
save(rpfdiab2, file='data/rpfdiab2.RData')


# Step 14: Exclude women who developed diabetes-related complications prior to
#          pregnancy admission.
rpfdiab3 <- rpfdiab2

rpfdiab3$flag <- 0
# Flag for the exposed women if complications occurred before pregnancy.
rpfdiab3$flag[rpfdiab3$pregexp == 1 & rpfdiab3$compdate < rpfdiab3$pregdate] <- 1
# Flag for the non-exposed women if complication occurred before the virtual
# pregnancy date.
rpfdiab3$flag[rpfdiab3$pregexp == 0 & rpfdiab3$compdate < rpfdiab3$date +
rpfdiab3$timepreg] <- 1

# Check the number of women flagged (14 exposed and 55 non-exposed).
xtabs(~flag+pregexp, data=rpfdiab3)

# Remove rows that are flagged and the flag column (25).
rpfdiab3 <- rpfdiab3[rpfdiab3$flag == 0, 1:24]

# Save the data.
save(rpfdiab3, file='data/rpfdiab3.RData')


# Step 15: Create the survival variables.
# Sort by rootlpno. Not essential, but this will make the results easier to
# compare.
rpfdiab4 <- rpfdiab3[order(rpfdiab3$rootlpno), ]

# Calculate the time to complications/exit variable for those without
# complications.
no_complications <- which(rpfdiab4$complic == 0)
rpfdiab4$compsurv[no_complications] <-
    rpfdiab3$exit[no_complications] - rpfdiab3$date[no_complications]
# And those with.
complications <- which(rpfdiab4$complic == 1)
rpfdiab4$compsurv[complications] <-
    rpfdiab3$compdate[complications] - rpfdiab3$date[complications]

# Move time-zero forward to the real or virtual date by subtracting timepreg.
rpfdiab4$compsurv <- rpfdiab4$compsurv - rpfdiab4$timepreg
```

```
# And reformat as an integer, important for survival analysis.
rpfdiab4$compsurv <- as.numeric(rpfdiab4$compsurv)

# View the data as shown in the workbook.
View(rpfdiab4[1:15, 20:25])
```

R_Solutions - RStudio Source Editor

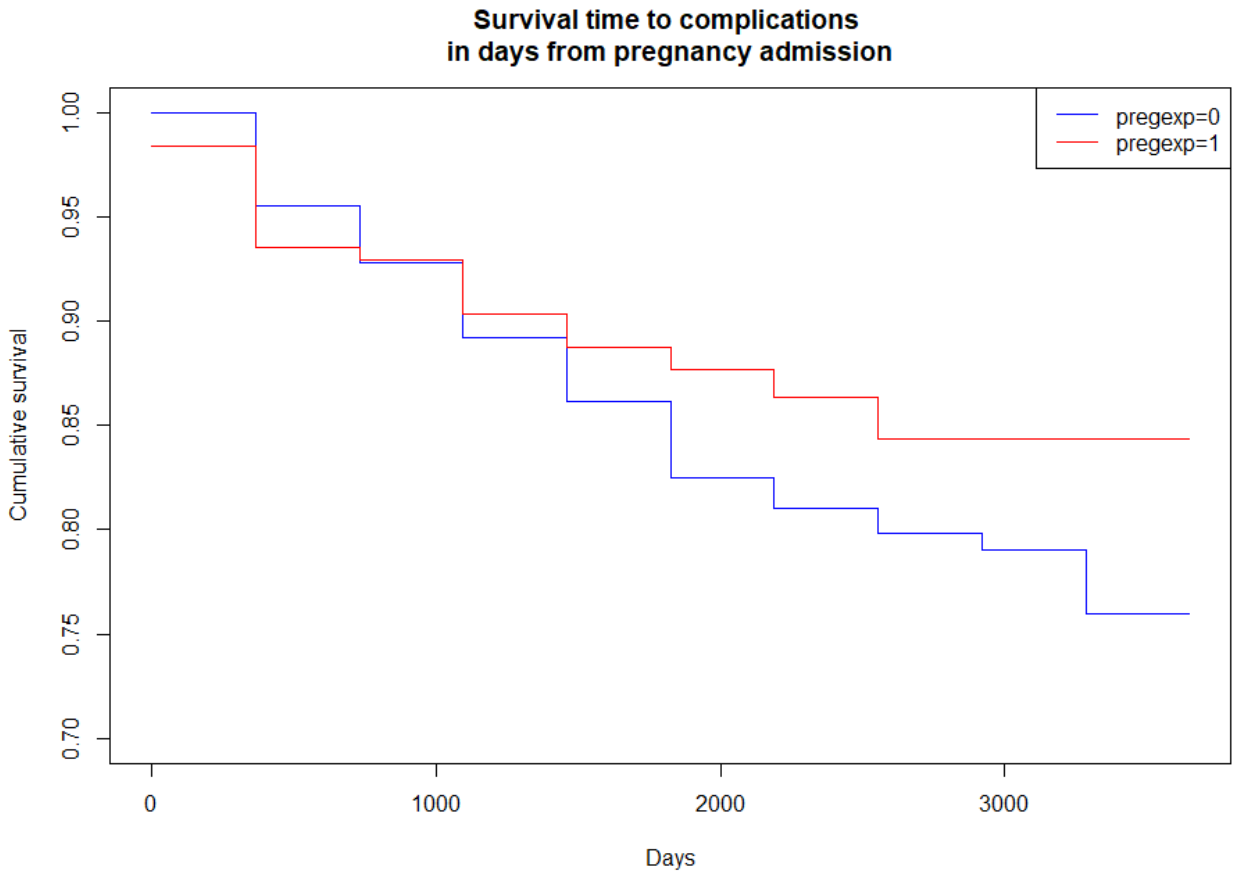
rpfdiab4[1:15, 20:25]

Filter

	compdate	complic	pregdate	pregexp	timepreg	compsurv
1	NA	0	1995-07-13	1	598	1632
2	NA	0	1996-07-13	1	4	1266
3	1997-08-12	1	NA	0	1189	1568
4	NA	0	NA	0	1094	2117
5	NA	0	NA	0	31	1241
7	NA	0	NA	0	360	1421
8	NA	0	1994-06-29	1	1116	2011
9	1998-08-21	1	NA	0	360	158
10	1995-02-23	1	1994-11-04	1	243	111
12	NA	0	NA	0	17	1259
13	NA	0	NA	0	0	599
14	1998-07-21	1	1993-10-21	1	712	1734
16	NA	0	NA	0	194	242
17	1995-09-19	1	NA	0	650	4
18	NA	0	1992-10-04	1	560	2644

Showing 1 to 15 of 15 entries, 6 total columns

```
# Save the data.
save(rpfdiab4, file='data/rpfdiab4.RData')


# Step 16: Survival analysis.
surv3.16 <- survfit(Surv(ceiling(compsurv / 365), complic) ~ pregexp,
                    conf.type='none', data=rpfdiab4)
# Plot the survival curve.
plot(surv3.16,
     main='Survival time to complications \nin days from pregnancy admission',
     xlab='Days', ylab='Cumulative survival', col=c('blue', 'red'),
     xscale=1/365, ylim=c(0.7,1), mark.time=F)
# Add a legend.
legend('topright', c('pregexp=0','pregexp=1'), lty=c(1, 1), col=c('blue', 'red'))
```




```
# Step 17: Investigate confounding by age and disease severity.
rpfdiab5 <- rpfdiab4
# Classify women into stages 1,2 and 3 according to values of stgpt1 and 2.
rpfdiab5$stagpreg[rpfdiab5$timepreg <= rpfdiab5$stgpt1] <- 1
rpfdiab5$stagpreg[rpfdiab5$timepreg >= rpfdiab5$stgpt1 & rpfdiab5$timepreg <=
(rpfdiab5$stgpt1+rpfdiab5$stgpt2)] <- 2
rpfdiab5$stagpreg[rpfdiab5$timepreg >= (rpfdiab5$stgpt1+rpfdiab5$stgpt2)] <- 3
# Check the numbers - expecting something like 417, 85 and 87 for stages 1-3.
table(rpfdiab5$stagpreg)

# Save the data.
save(rpfdiab5, file='data/rpfdiab5.RData')


# Step 18: Cox regressions.
# #1 with pregexp only (exp(coeff) = 0.77).
cox3.1 <- coxph(Surv(compsurv, complic) ~ pregexp, data=rpfdiab5)
summary(cox3.1)

# #2 with pregexp and age (exp(coeff) = 0.83, 1.03).
cox3.2 <- coxph(Surv(compsurv, complic) ~ pregexp + age, data=rpfdiab5)
summary(cox3.2)
```

```
# #3 with pregexp, age and stagpreg (exp(coeff) = 0.85, 1.03, 1.08).
cox3.3 <- coxph(Surv(compsurv, complic) ~ pregexp + age + stagpreg, data=rpfdiab5)
summary(cox3.3)


# Step 19: Repeat the third model for women with timepreg > 270 or <= 270 days.
timepreg_gt_270 <- subset(rpfdiab5, timepreg > 270)
timepreg_lte_270 <- subset(rpfdiab5, timepreg <= 270)

# #1 Time to pregnancy is greater than 270 days (exp(coeff) = 1.01, 1.07, 1.35).
cox3.4 <- coxph(Surv(compsurv, complic) ~ pregexp + age + stagpreg,
data=timepreg_gt_270)
summary(cox3.4)

# #2 Time to pregnancy is less than or equal to 270 days (exp(coeff) = 0.69, 0.99,
1.18).
cox3.5 <- coxph(Surv(compsurv, complic) ~ pregexp + age + stagpreg,
data=timepreg_lte_270)
summary(cox3.5)
```

TRAINING SESSION 3: TIDYVERSE-R SYNTAX SOLUTIONS

```
# Preparation
# Start a new R session by opening the project.

# Load the libraries used in the exercise.
library(dplyr)
library(magrittr)
library(stringr)
library(psych)
library(survival)
library(survminer)

################################### Exercise 5 ###################################

#### Steps 1-2: Load dates of first diabetic complication dates and create
####            indicator

#### Step 1: Load the preprepared file
load('data/diabcomp.RData')


#### Step 2: Create the complications indicator and date
load('data/diabetes6.RData')

diabetes7 <- diabetes6 %>% merge(diabcomp, by='rootlpno', all.x=TRUE)

diabetes7 <- diabetes7 %>%
    mutate(complic=ifelse(is.na(compdate), 0, 1))

table(diabetes7$complic)

save(diabetes7, file='data/diabetes7.RData')

#### Steps 3-4: Load aug date from platform file onto the HMDSdata file

#### Step 3: Reload the HMDSdata as we left it
load('data/HMDSdata2.RData')


#### Step 4: Merge the date from the updated platform file
HMDSdata3 <- HMDSdata2 %>%
    merge(diabetes7 %>% select(rootlpno, date), by='rootlpno', all.x=TRUE) %>%
    arrange(rootlpno, fileseq)


#### Steps 5-7: Tag pregnancy related records, sequence and cut-down.

#### Step 5: Tag all records that mention a pregnancy
pregnancy_codes <- c(
    # Codes 630.00 through 677.99 as strings.
    format(seq(630.00, 677.99, 0.01), digits=6),
    # Codes O00.0 through O99.9 as strings (notes that seq produces ' ' as the
    # leading character where we need '00' not ' 0'. We can do this using
```

```
    # sprintf.
    paste("O", sprintf("%05.2f", seq(00.0, 99.99, 0.01)), sep=""))
# Work through columns 'diag1' to 'diag21' and look for any which contains a
# diabetes code.
# The paste() call concatenates the 'diag' with the loop counter, i to give
# strings 'diag1', 'diag2' etc., the sym() call converts the string to a
# symbol (i.e. a variable name), the !! evaluates the symbol which in this case
# produces the column name. So we end up with ifelse(diag1 %in% ...,
# ifelse(diag2 %in% ... and so on.
HMDSdata3 <- HMDSdata3 %>% mutate(pregnant=0)
for(i in 1:21){
    HMDSdata3 <- HMDSdata3 %>%
        mutate(pregnant=ifelse(!!sym(paste('diag', i, sep='')) %in%
                                pregnancy_codes, 1, pregnant))
}

table(HMDSdata3$pregnant)


#### Step 6: Create pregnancy sequence variable
HMDSdata3 <- HMDSdata3 %>%
    mutate(admdate_after_date=ifelse(admdate>=date, 1, 0)) %>%
    arrange(rootlpno, -admdate_after_date, -pregnant) %>%
    group_by(rootlpno) %>%
    mutate(pregseq=ifelse(admdate>=date & pregnant==1, row_number(), NA)) %>%
    ungroup() %>%
    select(-admdate_after_date) %>%
    arrange(rootlpno, morbseq, fileseq)

View(HMDSdata3[477:483, 46:51])
```

R_Solutions - RStudio Source Editor

HMDSdata3[477:483, 47:51]

Filter

	morbseq	condseq	date	pregnant	pregseq
1	1	NA	1996-07-09	1	NA
2	2	NA	1996-07-09	1	1
3	3	NA	1996-07-09	0	NA
4	4	NA	1996-07-09	1	2
5	5	NA	1996-07-09	0	NA
6	6	NA	1996-07-09	1	3
7	1	1	1995-10-25	0	NA

Showing 1 to 7 of 7 entries, 5 total columns

```
save(HMDSdata3, file='data/HMDSdata3.RData')


#### Step 7: Cutdown with first-date pregnancies
HMDSpreg <- HMDSdata3 %>%
    filter(pregseq==1) %>% select(rootlpno, pregdate=admdate)

save(HMDSpreg, file='data/HMDSpreg.RData')


### Steps 8-9: Load pregnancy dates onto diabetes platform and restrict study domain to
women aged 15-39 at aug date

#### Step 8: Load the pregnancy dates into the platform file and create the exposure
variable
diabetes8 <- diabetes7 %>%
    merge(HMDSpreg, all.x=TRUE) %>%
    mutate(pregexp=ifelse(is.na(pregdate), 0, 1))

save(diabetes8, file='data/diabetes8.RData')


#### Step 9: Restrict to women of reproductive age
rpfdiab <- diabetes8 %>%
    filter(!is.na(sex) & sex=='2' & (age %in% 15:39))

save(rpfdiab, file='data/rpfdiab.RData')


#### Step 10: Create derivative file of women with a pregnancy, randomly sort,
####          number and cut-down
set.seed(2000000)
rpfexp <- rpfdiab %>%
    filter(pregexp==1) %>% mutate(randsort=runif(n(), min=0, max=1)) %>%
    arrange(randsort) %>% mutate(randseq=row_number()) %>%
    mutate(timepreg=pregdate-date)

View(rpfexp[1:15, 20:26])
```

R_Solutions - RStudio Source Editor

rpfexp[1:15, 20:26]

Filter

	compdate	complic	pregdate	pregexp	randsort	randseq	timepreg
1	1999-07-14	1	1999-07-11	1	0.005695261	1	255
2	NA	0	1994-05-29	1	0.014691459	2	204
3	NA	0	1996-03-13	1	0.016302094	3	0
4	1994-08-15	1	1992-06-07	1	0.017216894	4	820
5	NA	0	1990-07-15	1	0.019267581	5	147
6	NA	0	1999-04-12	1	0.022334185	6	0
7	NA	0	1998-10-14	1	0.026472706	7	122
8	1998-11-21	1	1991-12-21	1	0.028545107	8	29
9	NA	0	1994-02-12	1	0.030311104	9	48
10	NA	0	1996-07-13	1	0.031194580	10	4
11	NA	0	1994-11-16	1	0.032503811	11	43
12	NA	0	1993-04-13	1	0.035442332	12	911
13	1990-07-24	1	1991-01-23	1	0.045416619	13	305
14	NA	0	1995-01-12	1	0.055990095	14	99
15	NA	0	1999-12-08	1	0.065924422	15	48

Showing 1 to 15 of 15 entries, 7 total columns

```
rpfexp <- rpfexp %>%
    select(rootlpno, randseq, timepreg)

save(rpfexp, file='data/rpfexp.RData')


#### Step 11: Create derivative file of women with no pregnancy, randomly sort,
####          number and cut-down
set.seed(3000000)
rpfnon <- rpfdiab %>%
    filter(pregexp==0) %>% mutate(randsort=runif(n(), min=0, max=1)) %>%
    arrange(randsort)
rpfnon <- rpfnon %>%
    # The following line is a little more complicated than it essential, but it
    # it creating the sequence length based on the matching sequence in the
    # exposed file rather than using a 'magic' constant.
    mutate(randseq=rep(1:nrow(rpfexp),
               ceiling(nrow(rpfnon)/nrow(rpfexp)))[1:nrow(rpfnon)])

head(rpfnon[1:15,], 15)
```

```
#### Steps 12-13: Assign time to pregnancy from exposed to non-exposed women,
####               load into platform and exclude women with impossible dates

#### Step 12: Sort, load timepreg from rpfexp and cutdown
rpfnon <- rpfnon %>%
    merge(rpfexp %>% select(randseq, timepreg), by='randseq', all.x=TRUE) %>%
    arrange(rootlpno) %>%
    select(rootlpno, timepreg)

save(rpfnon, file='data/rpfnon.RData')


#### Step 13: Excluse non-exposed women with an impossible virtual pregnancy date
# Gets the time to virtual pregnancy for the non-exposed women into the platform
# file.
rpfdiab2 <- rpfdiab %>% merge(rpfnon, all.x=TRUE)
# Calculate the actual time from entry date to pregnancy for the exposed women.
rpfdiab2 <- rpfdiab2 %>%
    mutate(timepreg=ifelse(pregexp==1, pregdate-date, timepreg))

# Compare the time to pregnancy in the two groups (as defined by the values in
# pregexp). We could use individual calls to functions or we could create our
# own function which aggregates the results we want.
overview <- function(vector){
    cat('Length', names(summary(vector)), 'SD', '\n', sep='\t')
    cat(length(vector), summary(vector, digits=4), sd(vector), '\n', sep='\t')
}
with(rpfdiab2, tapply(timepreg, pregexp, overview))
overview(rpfdiab2$timepreg)
```

Pregnancy	Frequency	Mean	SD
0	516	449.17	616.65
1	200	458.81	623.53

```
# *** Re-run from Step 10 if your values are too different. ***

# Check and clear non-exposed women whose virtual time-to-pregnancy exceeds
# their available follow-up time.
# The pregexp test is not necessary unless you are being pedantic.
rpfdiab2 <- rpfdiab2 %>%
    mutate(flag=ifelse(pregexp==0 & (exit-date < timepreg), 1, 0))

xtabs(~flag+pregexp, data=rpfdiab2)
# 56 non-exposed women are flagged.

rpfdiab2 <- rpfdiab2 %>% filter(flag==0) %>% select(-flag)
# Leaves the 660 records.
```

```
save(rpfdiab2, file='data/rpfdiab2.RData')


### Steps 14-15: Exclude women with prior complications and calculate survival time to
complication

#### Step 14: Exclude women who developed complications prior to their pregnancy date
rpfdiab3 <- rpfdiab2 %>%
    # Set the column to zero to simplify what happens with NA's.
    mutate(flag=0) %>%
    mutate(flag=ifelse(pregexp==1,
                       ifelse(pregdate>compdate, 1, 0),
                       ifelse(timepreg>compdate-date, 1, 0)))

xtabs(~flag+pregexp, data=rpfdiab3, addNA=TRUE)
# 14 exposed and 55 non-exposed flagged.
# 339 / 164 NAs.

rpfdiab3 <- rpfdiab3 %>% filter(is.na(flag) | flag==0) %>% select(-flag)

save(rpfdiab3, file='data/rpfdiab3.RData')


#### Step 15: Survival time to complication
rpfdiab4 <- rpfdiab3 %>% arrange(rootlpno) %>%
    mutate(compsurv=ifelse(complic==0, exit-date, compdate-date)) %>%
    mutate(compsurv=as.numeric(compsurv-timepreg))

View(rpfdiab4[1:15, 20:25])
```

R_Solutions - RStudio Source Editor

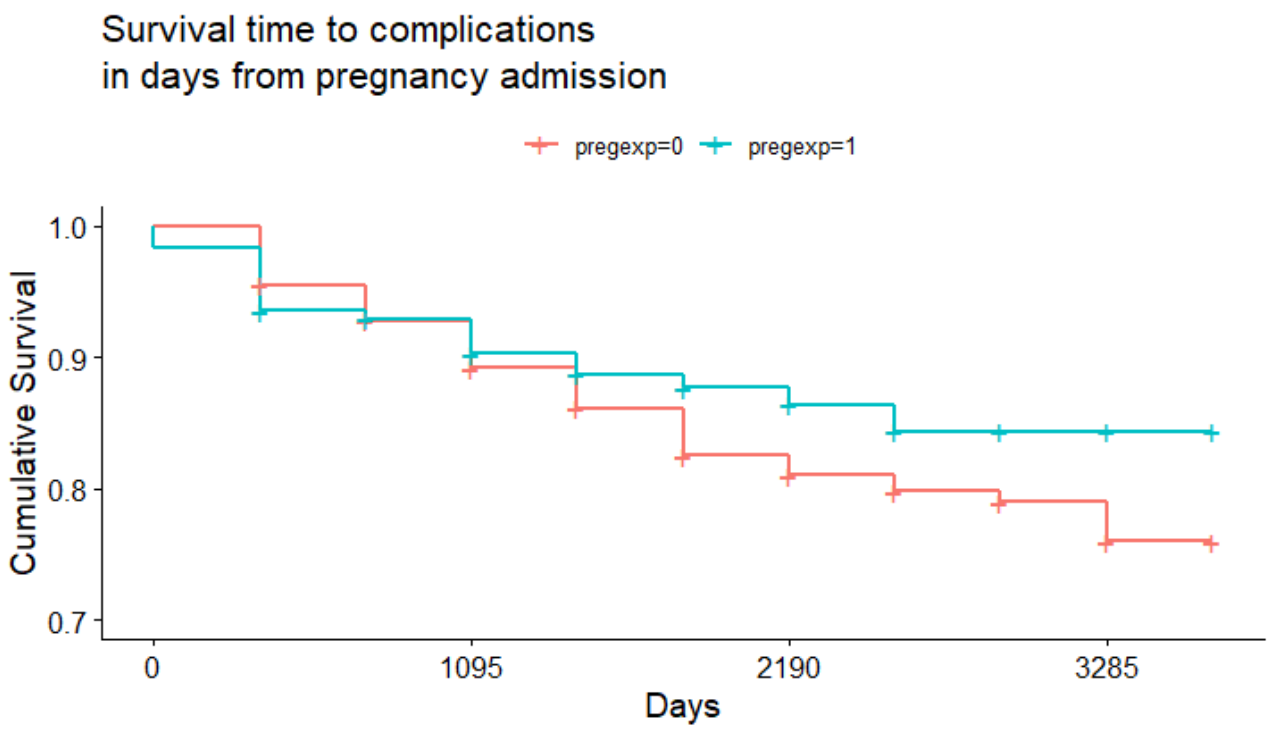
rpfdiab4[1:15, 20:25]

Filter

	compdate	complic	pregdate	pregexp	timepreg	compsurv
1	NA	0	1995-07-13	1	598	1632
2	NA	0	1996-07-13	1	4	1266
3	1997-08-12	1	NA	0	1189	1568
4	NA	0	NA	0	1094	2117
5	NA	0	NA	0	31	1241
6	NA	0	NA	0	360	1421
7	NA	0	1994-06-29	1	1116	2011
8	1998-08-21	1	NA	0	360	158
9	1995-02-23	1	1994-11-04	1	243	111
10	NA	0	NA	0	17	1259
11	NA	0	NA	0	0	599
12	1998-07-21	1	1993-10-21	1	712	1734
13	NA	0	NA	0	194	242
14	1995-09-19	1	NA	0	650	4
15	NA	0	1992-10-04	1	560	2644

Showing 1 to 15 of 15 entries, 6 total columns

```
save(rpfdiab4, file='data/rpfdiab4.RData')


#### Steps 16-19: Survival analyses

#### Step 16: Survival plot
surv3.16 <- survfit(Surv(ceiling(compsurv/365), complic)~pregexp,
                    conf.int=0, data=rpfdiab4)
x_scale <- 1/365
y_limits <- c(0.7, 1)
colours <- c('blue', 'red')
labels <- c('pregexp=0', 'pregexp=1')
plot(surv3.16, xscale=x_scale, ylim=y_limits, mark.time=F,
     main="Survival time to complications \nin days from pregnancy admission",
     xlab="Days", ylab="Cumulative survival", col=colours
    )
legend("topright", labels, lty=c(1,1), col=colours)

ggsurvplot(surv3.16, conf.int=FALSE, xscale=x_scale, break.time.by=3,
           ylim=y_limits,
           title='Survival time to complications \nin days from pregnancy admission',
           xlab='Days', ylab='Cumulative Survival',
           legend='top', legend.title='',
           legend.labs=labels)
```




```
#### Step 17: Check for confounding of age and disease severity
rpfdiab5 <- rpfdiab4 %>%
    mutate(stagpreg=ifelse(timepreg<=stgpt1, 1, 0)) %>%
    mutate(stagpreg=ifelse(timepreg>=stgpt1 & timepreg<=(stgpt1+stgpt2),
                           2, stagpreg)) %>%
```

```
    mutate(stagpreg=ifelse(timepreg>=stgpt1+stgpt2, 3, stagpreg))

table(rpfdiab5$stagpreg)

save(rpfdiab5, file='data/rpfdiab5.RData')


#### Step 18: Cox regressions
cox3.1 <- coxph(Surv(compsurv,complic)~pregexp, data=rpfdiab5)
summary(cox3.1)

cox3.2 <- coxph(Surv(compsurv,complic)~pregexp+age, data=rpfdiab5)
summary(cox3.2)

cox3.3 <- coxph(Surv(compsurv,complic)~pregexp+age+stagpreg, data=rpfdiab5)
summary(cox3.3)
# 0.77-0.84-0.85


#### Step 19: Repeat with subsets with timepreg > or <= 270 days
sub1 <- rpfdiab5 %>% filter(timepreg > 270)
sub2 <- rpfdiab5 %>% filter(timepreg <= 270)

cox3.4 <- coxph(Surv(compsurv,complic)~pregexp+age+stagpreg, data=sub1)
summary(cox3.4)

cox3.5 <- coxph(Surv(compsurv,complic)~pregexp+age+stagpreg, data=sub2)
summary(cox3.5)
# 1.01-0.70
```

TRAINING SESSION 4: R SYNTAX SOLUTIONS

```
################################################################################
# AALHD DAY 4
# R Syntax solutions
#
# Dr Pete Arnold based on code by Dr Joanne Demmler
# Updated March 2021
################################################################################
# Hints about what has been used in this solution.
#
################################################################################

library(data.table)
library(flextable)      # Used to produce tables in the viewer panel (images).

################################### Exercise 4 ###################################

# Step 1: In the HMDS3 data file, tag records with UGC (upper gastrointestinal
#         complications) codes.
load('data/HMDSdata3.RData')

HMDSdata3$ugc <- 0

UGC_codes <- c(format(seq(531.00, 535.99, 0.01), digits=6),
               format(seq(578.00, 578.99, 0.01), digits=6),
               paste('K', format(seq(25.00, 29.99, 0.01), digits=5), sep=''),
               paste('K', format(seq(92.00, 92.29, 0.01), digits=5), sep=''))

HMDSdata3$ugc[HMDSdata3$diag1 %in% UGC_codes] <- 1


# Step 2: Create a UGC sequence variable.
# Probably as well to get the file in order.
HMDSdata3 <- HMDSdata3[order(HMDSdata3$fileseq), ]
# setkeyv(temp_DT, "rootlpno")
temp_DT <- data.table(HMDSdata3)
temp_DT <- temp_DT[ugc == 1, ugcseq := 1:.N, by=rootlpno]

index <- as.data.frame(temp_DT)
rm(temp_DT)

# Restrict to records after January 1991.
index_records_after_date <- which(index$ugcseq == 1 & index$admdate >= '1991-01-31')
index <- index[index_records_after_date, ]

# Cut down the file.
index <- index[, c('rootlpno', 'sex', 'age', 'admdate')]

# View the data as shown in the workbook.
View(index[1:12, ])
```

R_Solutions - RStudio Source Editor

index[1:12,]

Filter

	rootlpno	sex	age	admdate
206	41	1	69	1997-07-19
239	44	2	41	1999-03-06
364	66	2	78	1995-11-22
445	84	2	38	1996-08-24
563	116	1	75	1998-10-04
605	124	1	64	1997-06-07
628	131	1	79	1992-03-07
657	143	1	88	1991-09-11
684	148	2	44	1992-01-17
752	164	2	32	1994-07-21
768	169	2	71	1994-11-19
789	172	1	41	1993-02-12

Showing 1 to 12 of 12 entries, 4 total columns

```
# Save the data.
save(index, file='data/index.RData')


# Step 3: Tag indomethacin exposure.
load('data/PBSdata3.RData')

indomethacin_codes <- c(2454, 2459, 2757, 5126, 5127, 5128)

PBSdata3$indometh <- 0
PBSdata3$indometh[PBSdata3$pbsitem %in% indomethacin_codes] <- 1

indexposure <- PBSdata3[PBSdata3$indometh == 1, ]

# Check the numbers (5,351 rows).
nrow(indexposure)

# Save the data.
save(indexposure, file='data/indexposure.RData')


# Step 4: Load the admdate from the index file onto the indexposure.
indexposure2 <- merge(indexposure, index[, c('rootlpno', 'admdate')], all.x=TRUE)
# Remove missing data.
```

```
indexposure2 <- indexposure2[!is.na(indexposure2$admdate), ]

# Save the data.
save(indexposure2, file='data/indexposure2.RData')


# Step 5: Create the exposure variable.
indexposure3 <- indexposure2

# Get the time from dispdate to the admdate.
indexposure3$exposure_time <- indexposure3$admdate - indexposure3$dispdate
# Set the exposure flag to none by default.
indexposure3$exposure <- 0
# And flag is 1 for the case window.
indexposure3$exposure[indexposure3$exposure_time >= 0 &
                      indexposure3$exposure_time <= 30] <- 1
# And flag is 2 for the control window.
indexposure3$exposure[indexposure3$exposure_time >= 365 &
                      indexposure3$exposure_time <= 395] <- 2
# Delete the time variable.
indexposure3$exposure_time <- NULL

# Check the counts (0-1-2 = 425, 12, 5).
table(indexposure3$exposure)

# Just keep the cases and controls.
indexposure3 <- indexposure3[indexposure3$exposure %in% c(1, 2), ]

# Save the data.
save(indexposure3, file='data/indexposure3.RData')


# Step 6: Select the last exposures to indomethacin in the cases & controls.
# We don't need to turn the file upside down, we can just count down in the
# rootlpno groups.
#indexposure4 <- indexposure3[order(-indexposure3$fileseq), ]

temp_DT <- data.table(indexposure3)
# setkeyv(DT,"rootlpno")
# Note that we are counting down from .N to 1 this time.
temp_DT <- temp_DT[exposure == 1, casexseq := .N:1, by=rootlpno]
temp_DT <- temp_DT[exposure == 2, conexseq := .N:1, by=rootlpno]

indexposure4 <- as.data.frame(temp_DT)
rm(temp_DT)
# And we don't need to turn the file back up the right way.
# indexposure4 <- indexposure4[order(indexposure4$fileseq), ]

indexposure4 <- subset(indexposure4, casexseq == 1 | conexseq == 1,
                       select = c(rootlpno, dispdate, exposure), na.rm=TRUE)

# View the data as shown in the workbook.
View(indexposure4[1:14, ])
```

R_Solutions - RStudio Source...

indexposure4[1:14,]

Filter

	rootlpno	dispdate	exposure
1	305	1995-11-10	1
3	2777	1994-02-17	1
4	3074	1996-04-20	2
5	5732	1994-04-20	1
6	6250	1997-03-27	2
7	6250	1998-03-13	1
9	6410	1995-01-10	1
10	7175	1993-07-11	1
11	7591	1994-07-12	1
12	11187	1992-03-04	2
13	11191	1992-09-05	1
14	11551	1998-09-26	1
16	12432	1994-12-28	1
17	12432	1994-01-19	2

Showing 1 to 14 of 14 entries, 3 total columns

```
# Save the data.
save(indexposure4, file='data/indexposure4.RData')


# Step 7: Separate the cases and controls.
indcasexp <- subset(indexposure4, exposure == 1)
indconexp <- subset(indexposure4, exposure == 2)

# Save the data.
save(indcasexp, file='data/indcasexp.RData')
save(indconexp, file='data/indconexp.RData')


# Step 8: Create the case-crossover variables.
# Get the index data and merge with the case exposures (renaming the dispdate
# and exposure variables to avoid confusion).
index1 <- merge(index, indcasexp, all.x=TRUE)
names(index1)[5:6] <- c('casexdat', 'casexp')

# Don't need to recode the caseexp as they are already coded '1' but convert the
# NAs to '0'.
# index1$casexp[!is.na(index1$casexp)] <- 1
```

```
index1$casexp[is.na(index1$casexp)] <- 0

# Merge with the control exposures (again renaming the dispdate and exposure
# variables to avoid confusion).
index2 <- merge(index1, indconexp, all.x=TRUE)
names(index2)[7:8] <- c('conexdat', 'conexp')

# Recode control exposures from '2' to '1' and the NAs to '0'.
index2$conexp[!is.na(index2$conexp)] <- 1
index2$conexp[is.na(index2$conexp)] <- 0

# Save the data.
save(index2, file='data/index2.RData')


# Step 9: Crosstabulation of case and control exposures.
xtabs(~ casexp + conexp, data=index2)

# And as a rough flextable.
flextable(as.data.frame(xtabs(~ casexp + conexp, data=index2)))
```

casexp	conexp	Freq
0	0	958
1	0	8
0	1	2
1	1	2

```
# Step 10: Creation of reference subject file.

# Check that we have 970 index subjects.
nrow(index2)

indexadm <- index2
set.seed(2500000)

# Create a random sort variable and order the file.
indexadm$randsort <- runif(nrow(indexadm), min=0, max=1)
indexadm <- indexadm[order(indexadm$randsort), ]

# Create a randseq variable.
indexadm$randseq <- seq_len(nrow(indexadm))

# Cut down the file.
indexadm <- subset(indexadm, select = c(randseq, admdate))

# Save the data.
save(indexadm, file='data/indexadm.RData')
```

```
# Step 11: Load and randomly sort the diabetes file.
load('data/diabetes8.RData')

# Check that we have 10,675 diabetes subjects.
nrow(diabetes8)

set.seed(3500000)
diabetes8$randsort <- runif(nrow(diabetes8), min=0, max=1)

diabetes8 <- diabetes8[order(diabetes8$randsort), ]

# Check 10675/970 = 11 times and a little bit.
# Is there a smoother way to do this?
diabetes8$randseq <- c(rep(1:970, nrow(diabetes8) %/% 970),
                       seq(1:(nrow(diabetes8) %% 970)))


# Step 12: Assign the randomly sorted indexadms to the diabetes file and remove
#          any where the admdate exceeds the diabetes exit date.
# Not really necessary to sort here uing R.
diabetes8 <- diabetes8[order(diabetes8$randseq), ]

reference <- merge(diabetes8, indexadm, all.x=TRUE)
reference <- reference[order(reference$rootlpno), ]

reference <- subset(reference, exit >= admdate, select = c(rootlpno, admdate))

# How many subjects remain (something around 9,698).
nrow(reference)

# Save the data.
save(reference, file='data/reference.RData')


# Step 13: Tag indomethacin records in PBS.
load('data/PBSdata3.RData')
refexposure <- PBSdata3

# Defined above.
# indomethacin_codes <- c(2454, 2459, 2757, 5126, 5127, 5128)

refexposure$indometh[refexposure$pbsitem %in% indomethacin_codes] <- 1
refexposure <- subset(refexposure, indometh == 1, na.rm=TRUE)

refexposure <- refexposure[order(refexposure$rootlpno), ]

# Save the data.
save(refexposure, file='data/refexposure.RData')


# Step 14: Load the reference admdrates and remove blanks.
refexposure2 <- merge(refexposure, reference, all.x=TRUE)
refexposure2 <- subset(refexposure2, !is.na(admdate))
```

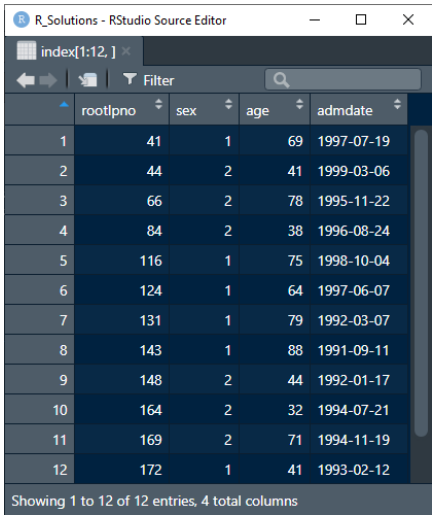
```
# Check how many records remain (approx. 4,728-4,828).
nrow(refexposure2)

# Save the data.
save(refexposure2, file='data/refexposure2.RData')


# Step 15: Create the exposure variable.
refexposure3 <- refexposure2
refexposure3$exposure_time <- refexposure3$admdate - refexposure3$dispdate
refexposure3$exposure <- 0
refexposure3$exposure[refexposure3$exposure_time >= 0 & refexposure3$exposure_time <=
30] <- 1
refexposure3$exposure[refexposure3$exposure_time >= 365 & refexposure3$exposure_time <=
395] <- 2
refexposure3$exposure_time <- NULL

# Check the numbers.
table(refexposure3$exposure)

# Only keep those records which are in the specified periods.
refexposure3 <- subset(refexposure3, exposure %in% c(1, 2))

# Check the table size.
nrow(refexposure3)

# Save the data.
save(refexposure3, file='data/refexposure3.RData')


# Step 16: Create the case and control exposure sequence variables.
# refexposure4 <- refexposure3[order(-refexposure3$fileseq), ]

temp_DT <- data.table(refexposure3)
# setkeyv(DT, "rootlpno")
temp_DT <- temp_DT[exposure == 1, casexseq := .N:1, by=rootlpno]
temp_DT <- temp_DT[exposure == 2, conexseq := .N:1, by=rootlpno]

refexposure4 <- as.data.frame(temp_DT)
rm(temp_DT)
# refexposure4 <- refexposure4[order(refexposure4$fileseq), ]

refexposure4 <- subset(refexposure4, casexseq == 1 | conexseq == 1,
                       select = c(rootlpno, dispdate, exposure), na.rm=TRUE)
# Check the size (ca. 84).
nrow(refexposure4)

# Save the data.
save(refexposure4, file='data/refexposure4.RData')


# Step 17: Separate the case and control records.
refcasexp <- subset(refexposure4, exposure == 1)
refconexp <- subset(refexposure4, exposure == 2)
```

```
# Save the data.
save(refcasexp, file='data/refcasexp.RData')
save(refconexp, file='data/refconexp.RData')


# Step 18:
# Merge the cases with the reference file.
reference1 <- merge(reference, refcasexp, all.x=TRUE)
names(reference1)[3:4] <- c('casexdat', 'casexp')

reference1$casexp[!is.na(reference1$casexp)] <- 1
reference1$casexp[is.na(reference1$casexp)] <- 0

# Merge the controls with the cases/reference file.
reference2 <- merge(reference1, refconexp, all.x=TRUE)
names(reference2)[5:6] <- c('conexdat', 'conexp')

reference2$conexp[!is.na(reference2$conexp)] <- 1
reference2$conexp[is.na(reference2$conexp)] <- 0

# Save the data.
save(reference2, file='data/reference2.RData')


# Step 19: Cross-tabulate.
xtabs(~ casexp + conexp, data=reference2)

# And as a rough flextable.
flextable(as.data.frame(xtabs(~ casexp + conexp, data=reference2)))
```

casexp	conexp	Freq
0	0	9,626
1	0	37
0	1	23
1	1	12

TRAINING SESSION 4: TIDYVERSE-R SYNTAX SOLUTIONS

```
# Preparation
# Start a new R session by opening the project.

# Load the libraries used in the exercise.
library(dplyr)
library(magrittr)


################################## Exercise 4 ##################################

#### Steps 1-2: Open HMDS data, tag, sequence, cutdown and save as index

#### Step 1: Load the preprepared file and flag UGC as principle diagnosis
load('data/HMDSdata3.RData')

UGC <- c(format(seq(531.00, 535.99, 0.01), digits=6),
         format(seq(578.00, 578.99, 0.01), digits=6),
         paste("K", format(seq(25.00, 29.99, 0.01), digits=5), sep=""),
         paste("K", format(seq(92.00, 92.29, 0.01), digits=5), sep="")
        )

index <- HMDSdata3 %>%
    mutate(ugc=ifelse(diag1 %in% UGC, 1, 0))


#### Step 2: Sequence and cut-down to produce the index file
index <- index %>% arrange(rootlpno, fileseq) %>%
    filter(ugc==1) %>%
    group_by(rootlpno) %>%
    mutate(ugcseq=row_number()) %>%
    ungroup() %>%
    filter(ugcseq==1 & admdate>='1991-01-31') %>%
    select(rootlpno, sex, age, admdate)

View(index[1:12, ])
```



	rootlpno	sex	age	admdate
1	41	1	69	1997-07-19
2	44	2	41	1999-03-06
3	66	2	78	1995-11-22
4	84	2	38	1996-08-24
5	116	1	75	1998-10-04
6	124	1	64	1997-06-07
7	131	1	79	1992-03-07
8	143	1	88	1991-09-11
9	148	2	44	1992-01-17
10	164	2	32	1994-07-21
11	169	2	71	1994-11-19
12	172	1	41	1993-02-12

Showing 1 to 12 of 12 entries, 4 total columns

```
save(index, file='data/index.RData')


#### Steps 3-4: Open PBSdata, tag indomethacin records, delete the rest, load UGC
####            admdate and delete non-UGC records

#### Step 3: Select records with an indomethacin PBS item
load('data/PBSdata3.RData')
indomethacin <- c(2454, 2459, 2757, 5126, 5127, 5128)

indexposure <- PBSdata3 %>%
    mutate(indometh=ifelse(pbsitem %in% indomethacin, 1, 0)) %>%
    filter(indometh==1) %>%
    select(-indometh)

save(indexposure, file='data/indexposure.RData')


#### Step 4: Join with the admdate from index
indexposure2 <- indexposure %>%
    merge(index %>% select(rootlpno, admdate), all.x=TRUE) %>%
    filter(!is.na(admdate))

save(indexposure2, file='data/indexposure2.RData')


#### Step 5: Select those in the case or control windows
indexposure3 <- indexposure2 %>%
    mutate(exposure_time=admdate-dispdate) %>%
    mutate(exposure=ifelse(exposure_time>0 & exposure_time<=30, 1,
                           ifelse(exposure_time>=365 & exposure_time<=395, 2, 0)))

table(indexposure3$exposure)

indexposure3 <- indexposure3 %>%
    select(-exposure_time) %>%
    filter(exposure==1 | exposure==2)

save(indexposure3, file='data/indexposure3.RData')


#### Steps 6-7: Create casex and conex sequences and save; separately save case
#### window and control window exposure records

#### Step 6: Sequence and cut-down
indexposure4 <- indexposure3 %>%
    arrange(-fileseq) %>%
    group_by(rootlpno, exposure) %>%
    mutate(casexseq=ifelse(exposure==1, row_number(), 0),
           conexseq=ifelse(exposure==2, row_number(), 0)) %>%
    ungroup() %>%
    filter(casexseq==1 | conexseq==1) %>%
    arrange(fileseq) %>%
    select(rootlpno, dispdate, exposure)
```

```
View(indexposure4[1:14, ])
```

R_Solutions - RStudio Source Editor

indexposure4[1:14,]

Filter

	rootlpno	dispdate	exposure
1	305	1995-11-10	1
2	2777	1994-02-17	1
3	3074	1996-04-20	2
4	5732	1994-04-20	1
5	6250	1997-03-27	2
6	6250	1998-03-13	1
7	6410	1995-01-20	1
8	7175	1993-07-11	1
9	7591	1994-07-12	1
10	11187	1992-03-04	2
11	11191	1992-09-05	1
12	11551	1998-09-26	1
13	12432	1994-01-19	2
14	12432	1994-12-28	1

Showing 1 to 14 of 14 entries, 3 total columns

```
save(indexposure4, file='data/indexposure4.RData')


#### Step 7: Separate the case and control records
indcasexp <- indexposure4 %>% filter(exposure == 1)
indconexp <- indexposure4 %>% filter(exposure == 2)

save(indcasexp, file='data/indcasexp.RData')
save(indconexp, file='data/indconexp.RData')


#### Steps 8-9: Load exposure data onto the index platform and perform
####            matched-pair analysis

#### Step 8: Create a file suitable for tabular analysis of cross-over pairs
```

```
# load('data/index.RData')
index1 <- index %>% merge(indcasexp, all.x=TRUE) %>%
    rename(casexdat=dispdate, casexp=exposure) %>%
    mutate(casexp=ifelse(is.na(casexp), 0, 1))

index2 <- index1 %>% merge(indconexp, all.x=TRUE) %>%
    rename(conexdat=dispdate, conexp=exposure) %>%
    mutate(conexp=ifelse(is.na(conexp), 0, 1))

save(index2, file='data/index2.RData')


#### Step 9: Cross-tabulation
xtabs(~casexp+conexp, data=index2)
```

casexp	conexp	Freq
0	0	958
1	0	8
0	1	2
1	1	2

```
#### Steps 10-12: Randomly assign index admdates to platform and keep
#### non-exited-at-admdate diabetics

#### Step 10: Create a random list of admdates
# load('data/index2.RData'))
set.seed(2500000)

indexadm <- index2 %>%
    mutate(randsort=runif(nrow(index2), min=0, max=1)) %>%
    arrange(randsort) %>%
    mutate(randseq=row_number()) %>%
    select(randseq, admdate)

save(indexadm, file='data/indexadm.RData')


#### Step 11: Create a random index in the platform data frame
load('data/diabetes8.RData')
set.seed(3500000)
index_size <- nrow(indexadm)
platform_size <- nrow(diabetes8)
diabetes8 <- diabetes8 %>%
    mutate(randsort=runif(platform_size, min=0, max=1)) %>%
    arrange(randsort) %>%
    mutate(randseq=rep(seq(1:index_size),
                       ceiling(platform_size/index_size))[1:platform_size])
```

```
#### Step 12: Create a random list of admdates for the reference group
reference <- diabetes8 %>%
    merge(indexadm, all.x=TRUE) %>%
    arrange(rootlpno) %>%
    filter(exit>=admdate) %>%
    select(rootlpno, admdate)

nrow(reference)

save(reference, file='data/reference.RData')


#### Steps 13-17: Repeat steps 3-7 for reference subjects for exposures in case
#### and control windows

#### Step 13: Select records with an indomethacin PBS item
# load('data/PBSdata3.RData')
refexposure <- PBSdata3 %>%
    mutate(indometh=ifelse(pbsitem %in% indomethacin, 1, 0)) %>%
    filter(indometh==1) %>%
    arrange(rootlpno)

save(refexposure, file='data/refexposure.RData')


#### Step 14: Join with the admdate from reference
refexposure2 <- refexposure %>%
    merge(reference, all.x=TRUE) %>%
    filter(!is.na(admdate))

nrow(refexposure2)
# 4704

save(refexposure2, file='data/refexposure2.RData')


#### Step 15: Select those in the case or control windows
refexposure3 <- refexposure2 %>%
    mutate(exposure_time=admdate-dispdate) %>%
    mutate(exposure=ifelse(exposure_time>=0 & exposure_time<=30, 1,
                           ifelse(exposure_time>=365 & exposure_time<=395,
                                  2, 0)))

table(refexposure3$exposure)

refexposure3 <- refexposure3 %>%
    select(-exposure_time) %>%
    filter(exposure==1 | exposure==2)

save(refexposure3, file='data/refexposure3.RData')


#### Step 16: Sequence and cut-down
refexposure4 <- refexposure3 %>%
```

```
    arrange(-fileseq) %>%
    group_by(rootlpno, exposure) %>%
    mutate(casexseq=ifelse(exposure==1, row_number(), 0),
           conexseq=ifelse(exposure==2, row_number(), 0)) %>%
    ungroup() %>%
    filter(casexseq==1 | conexseq==1) %>%
    arrange(fileseq) %>%
    select(rootlpno, dispdate, exposure)

nrow(refexposure4)

save(refexposure4, file='data/refexposure4.RData')


#### Step 17: Separate the case and control records
refcasexp <- refexposure4 %>% filter(exposure == 1)
refconexp <- refexposure4 %>% filter(exposure == 2)

save(refcasexp, file='data/refcasexp.RData')
save(refconexp, file='data/refconexp.RData')


#### Steps 18-19: Repeat steps 8 and 9 to perform matched pair analysis on
####              reference subjects

#### Step 18: Create a file suitable for tabular analysis of cross-over pairs
reference1 <- reference %>%
    merge(refcasexp, all.x = TRUE) %>%
    rename(casexdat=dispdate, casexp=exposure) %>%
    mutate(casexp=ifelse(is.na(casexp), 0, 1))

reference2 <- reference1 %>%
    merge(refconexp, all.x=TRUE) %>%
    rename(conexdat=dispdate, conexp=exposure) %>%
    mutate(conexp=ifelse(is.na(conexp), 0, 1))

save(reference2, file='data/reference2.RData')


#### Step 19: Cross-tabulation
xtabs(~casexp+conexp, data=reference2)
```

casexp	conexp	Freq
0	0	9,626
1	0	37
0	1	23
1	1	12

TRAINING SESSION 5: R SYNTAX SOLUTIONS

```
################################################################################
# AALHD DAY 5
# R Syntax solutions
#
# Dr Pete Arnold based on code by Dr Joanne Demmler
# Updated March 2021
################################################################################
# Hints about what has been used in this solution.
#
################################################################################

library(data.table)
library(chron)          # Used for dates in step 4.
library(survival)       # Used for the survival analyses from step 8.
library(psych)          # Used for the describe function in step 12.

################################### Exercise 5 ##################################

# Step 1: Tag PBS records with a statin item.
load('data/PBSdata3.RData')

PBSdata3$anystatin <- 0
statins <- c(1224, 1453, 1687, 1942, 2011, 2012, 2013, 2831, 2833, 2834, 2892,
             2967, 2978, 8023, 8024, 8173, 8197, 8213, 8214, 8215, 8303, 8304,
             8313, 8419, 8521, 8721, 8722, 8757)
PBSdata3$anystatin[ PBSdata3$pbsitem %in% statins] <- 1


# Step 2: Create the dispensing sequence variable, statseq.
temp_DT <- data.table(PBSdata3)
# setkeyv(temp_DT, 'rootlpno')
temp_DT <- temp_DT[anystatin == 1, statseq := 1:.N, by=rootlpno]

PBSstatin <- as.data.frame(temp_DT)
rm(temp_DT)

PBSstatin <- subset(PBSstatin, statseq == 1, select = c(rootlpno, dispdate))
names(PBSstatin)[2] <- 'statdate'

# Save the data.
save(PBSstatin, file='data/PBSstatin.RData')


# Step 3: Load the statdate onto the diabetes platform file.
# is this the correct file?
load('data/diabetes8.RData')

diabetes9 <- merge(diabetes8, PBSstatin, all.x=TRUE)

# Default flag is 0.
diabetes9$statin <- 0
# If the statin date is before date, flag is 9.
```

```
diabetes9$statin[!is.na(diabetes9$statdate) &
                 diabetes9$statdate < diabetes9$date] <- 9
# If the statin date is within 365 days from date, flag is 1.
diabetes9$statin[diabetes9$statin != 9 &
                 (diabetes9$statdate - diabetes9$date) <= 364] <- 1
# Remove the 9 flagged records.
diabetes9 <- subset(diabetes9, statin != 9)
# Check the count (9,713).
nrow(diabetes9)


# Step 4: Create a year variable.
diabetes9$year <- years(diabetes9$date)
# Select only records during or after 1991.
diabetes9 <- subset(diabetes9, year >= 1991)
# Check this is 8,263.
nrow(diabetes9)
# Save the data.
save(diabetes9, file='data/diabetes9.RData')


# Step 5: Merge in the supplied diabsupp variables.
load('data/diabsupp.RData')
diabetes10 <- merge(diabetes9, diabsupp, all.x=TRUE)

# Save the data.
save(diabetes10, file='data/diabetes10.RData')


# Step 6: Create the stage variable.
diabetes10$stage <- 1
diabetes10$stage[diabetes10$stgpt1 == 0 & diabetes10$stgpt2 > 0] <- 2
diabetes10$stage[diabetes10$stgpt1 == 0 & diabetes10$stgpt2 == 0 &
                 diabetes10$stgpt3 > 0] <- 3
# Check the counts.
table(diabetes10$stage)


# Step 7: Create the survival time and event variables.
# The time is the time from the diabetes date to the study exit date.
diabetes10$surv3 <- diabetes10$exit - diabetes10$date
diabetes10$dead3 <- diabetes10$dead

# If the survival time is greater than 3 years then flag as not dead and set the
# time to 3 years.
diabetes10$dead3[diabetes10$surv3 > 1095] <- 0
diabetes10$surv3[diabetes10$surv3 > 1095] <- 1095

# Make sure that the time is seen as a number.
diabetes10$surv3 <- as.numeric(diabetes10$surv3)

# Select only records where year is before 1998 and the useful columns.
diabstat <- subset(diabetes10, year < 1998,
                   select = c(rootlpno, sex, age, statin, year, seifagp, ariagp,
                              macss, stage, surv3, dead3))
```

```
# Check the data size (6,795).
nrow(diabstat)

# Save the data.
save(diabstat, file='data/diabstat.RData')


# Step 8: Cox regression by statin status.
cox5.8 <- coxph(Surv(surv3, dead3) ~ statin, data=diabstat)
# Expect an exp(coefficient) of 0.6084.
summary(cox5.8)
```

Console Terminal Jobs

~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/

```
> # Expect an exp(coefficient) of 0.6084.
> summary(cox5.8)
Call:
coxph(formula = Surv(surv3, dead3) ~ statin, data = diabstat)

  n= 6795, number of events= 709

          coef exp(coef) se(coef)      z Pr(>|z|)
statin -0.4969    0.6084   0.1866 -2.663  0.00774 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

       exp(coef) exp(-coef) lower .95 upper .95
statin    0.6084      1.644    0.4221     0.877

Concordance= 0.513  (se = 0.004 )
Likelihood ratio test= 8.28  on 1 df,   p=0.004
Wald test            = 7.09  on 1 df,   p=0.008
Score (logrank) test = 7.24  on 1 df,   p=0.007

>
```

```
# Step 9: Variable transformations for the multivariate fit.
diabstat2 <- diabstat
# Sex: recode female (2) as 0.
diabstat2$sex[diabstat2$sex == 2] <- 0
# Recode NAs as 0.5.
diabstat2$sex[is.na(diabstat2$sex)] <- 0.5
# Age: recode NAs as 59.2.
diabstat2$age[is.na(diabstat2$age)] <- 59.2

# Add variables for a fractional polynomial.
diabstat2$agem2 <- 1 / (diabstat2$age ^ 2)
diabstat2$agesqrt <- sqrt(diabstat2$age)
diabstat2$anycom <- 0
diabstat2$anycom[diabstat2$macss > 0] <- 1
diabstat2$macssln <- log2(diabstat2$macss + 0.01)
```

```
# Add binary indicators for the stages.
diabstat2$stage2 <- 0
diabstat2$stage2[ diabstat2$stage == 2] <- 1
diabstat2$stage3 <- 0
diabstat2$stage3[ diabstat2$stage == 3] <- 1

# Create grouped seifa and ariagp variables.
diabstat2$seifa25 <- diabstat2$seifagp
diabstat2$seifa25[diabstat2$seifa25 == 1] <- 0
diabstat2$seifa25[diabstat2$seifa25 %in% seq(2, 5)] <- 1
diabstat2$aria5 <- diabstat2$ariagp
diabstat2$aria5[diabstat2$ariagp %in% seq(1, 4)] <- 0
diabstat2$aria5[diabstat2$ariagp == 5] <- 1

# Multivariate Cox regression.
cox5.9 <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                     stage3 + anycom + macssln + seifa25 + aria5,
                data=diabstat2)
# Expect the MRR = exp(coef)[statin] = 0.72935.
summary(cox5.9)
```

```
Console   Terminal ×   Jobs ×
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> # Expect the MRR = exp(coef)[statin] = 0.72935.
> summary(cox5.9)
Call:
coxph(formula = Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt +
    stage2 + stage3 + anycom + macssln + seifa25 + aria5, data = diabstat2)

  n= 6795, number of events= 709

              coef  exp(coef)   se(coef)      z Pr(>|z|)
statin    -0.31560    0.72935    0.18802 -1.679  0.09325 .
sex        0.43536    1.54553    0.07669  5.677 1.37e-08 ***
agem2   2181.33957        Inf  437.91216  4.981 6.32e-07 ***
agesqrt    1.46506    4.32780    0.09194 15.934  < 2e-16 ***
stage2    -0.10794    0.89768    0.09733 -1.109  0.26740
stage3     0.31203    1.36620    0.17832  1.750  0.08013 .
anycom    -0.96986    0.37914    0.36486 -2.658  0.00786 **
macssln    0.37766    1.45887    0.06121  6.170 6.83e-10 ***
seifa25    0.24523    1.27792    0.09790  2.505  0.01225 *
aria5      0.40334    1.49681    0.18980  2.125  0.03358 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

        exp(coef) exp(-coef) lower .95 upper .95
statin     0.7294     1.3711    0.5045    1.0543
sex        1.5455     0.6470    1.3298    1.7962
agem2         Inf     0.0000       Inf       Inf
agesqrt    4.3278     0.2311    3.6141    5.1824
stage2     0.8977     1.1140    0.7418    1.0863
stage3     1.3662     0.7320    0.9632    1.9377
anycom     0.3791     2.6376    0.1855    0.7751
macssln    1.4589     0.6855    1.2939    1.6448
seifa25    1.2779     0.7825    1.0548    1.5482
aria5      1.4968     0.6681    1.0318    2.1713

Concordance= 0.825  (se = 0.007 )
Likelihood ratio test= 1083  on 10 df,   p=<2e-16
Wald test            = 997.4  on 10 df,   p=<2e-16
Score (logrank) test = 1276  on 10 df,   p=<2e-16

>
```

```
# Save the data.
save(diabstat2, file='data/diabstat2.RData')


# Step 10: Crosstabulate statin by year.
# Refactor the year variable to discard the now missing factors.
diabstat2$year <- factor(diabstat2$year, ordered=FALSE)
# Create the cross tabulation.
tab <- xtabs(~ year + statin, data=diabstat2)
# Show it and also do some processing on the numbers to get the percentages.
tab
# Calculate the total for each year.
year_counts <- apply(tab, 1, sum)
# Redisplay the table as percentages for each year.
tab_percents <- round((tab * 100) / year_counts, 2)
tab_percents

# Produce a nice flextable.
df <- merge(as.data.frame(tab),
            as.data.frame(tab_percents), by=c('year', 'statin'))
colnames(df) <- c('year', 'statin', 'count', 'percent')
flextable(df)
```

year	statin	count	percent
1991	0	1,038	98.0
1991	1	21	2.0
1992	0	1,361	95.7
1992	1	61	4.3
1993	0	899	95.4
1993	1	43	4.6
1994	0	826	94.5
1994	1	48	5.5
1995	0	793	90.9
1995	1	79	9.1
1996	0	773	87.3
1996	1	112	12.7
1997	0	646	87.2
1997	1	95	12.8

```
# Step 11: Cox regression by year.
cox5.11 <- coxph(Surv(surv3, dead3) ~ year, data=diabstat2)
summary(cox5.11)
```

Console Terminal Jobs

~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/

```
> summary(cox5.11)
Call:
coxph(formula = Surv(surv3, dead3) ~ year, data = diabstat2)

  n= 6795, number of events= 709

              coef exp(coef)  se(coef)      z Pr(>|z|)
year1992  0.008384  1.008419  0.111863  0.075 0.940254
year1993 -0.176702  0.838029  0.129652 -1.363 0.172917
year1994 -0.356475  0.700140  0.139787 -2.550 0.010768 *
year1995 -0.480854  0.618255  0.145202 -3.312 0.000928 ***
year1996 -0.413337  0.661440  0.141475 -2.922 0.003482 **
year1997 -0.565865  0.567869  0.169239 -3.344 0.000827 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

         exp(coef) exp(-coef) lower .95 upper .95
year1992    1.0084     0.9917    0.8099    1.2556
year1993    0.8380     1.1933    0.6500    1.0805
year1994    0.7001     1.4283    0.5324    0.9208
year1995    0.6183     1.6175    0.4651    0.8218
year1996    0.6614     1.5119    0.5013    0.8728
year1997    0.5679     1.7610    0.4076    0.7912

Concordance= 0.558  (se = 0.01 )
Likelihood ratio test= 31.88  on 6 df,   p=2e-05
Wald test            = 31.23  on 6 df,   p=2e-05
Score (logrank) test = 31.76  on 6 df,   p=2e-05

>
```

```
# Step 12: Propensity ranks.
logreg <- glm(statin ~ sex + age + stage2 + stage3 + macss + anycom,
              data=diabstat2, family='binomial')
summary(logreg)
```

```
Console   Terminal ×   Jobs ×
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> summary(logreg)

Call:
glm(formula = statin ~ sex + age + stage2 + stage3 + macss +
    anycom, family = "binomial", data = diabstat2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6557  -0.4467  -0.3294  -0.3202   2.4675

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.041324   0.208015 -14.621  < 2e-16 ***
sex         -0.014311   0.098021  -0.146    0.884
age          0.002002   0.003240   0.618    0.537
stage2       0.745549   0.106703   6.987 2.81e-12 ***
stage3       0.179164   0.215765   0.830    0.406
macss        0.388520   0.252889   1.536    0.124
anycom      -0.109320   0.178366  -0.613    0.540
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3360.2  on 6794  degrees of freedom
Residual deviance: 3307.0  on 6788  degrees of freedom
AIC: 3321

Number of Fisher Scoring iterations: 5

>
```

```
# The coefficients are available with the coef() function. Index 1 is the
# intercept, then in the order listed.
# We can then calculate the propensity score for each row in the table by
# multiplying the parameter by the coefficient.
diabstat2$propen <- coef(logreg)[2]*diabstat2$sex +
                    coef(logreg)[3]*diabstat2$age +
                    coef(logreg)[4]*diabstat2$stage2 +
                    coef(logreg)[5]*diabstat2$stage3 +
                    coef(logreg)[6]*diabstat2$macss +
                    coef(logreg)[7]*diabstat2$anycom
# Get the descriptive stats for the propensity score.
describe(diabstat2$propen)

# Look at the quintiles.
quintiles <- c(0.0, 0.2, 0.4, 0.6, 0.8, 1.0)
quantile(diabstat2$propen, prob=quintiles)

# With a flextable.
quantiles <- as.data.frame(quantile(diabstat2$propen, prob=quintiles))
```

```
# Replace the vars (number) with the row name.
quantiles$rank <- row.names(quantiles)
colnames(quantiles)[1] <- 'propen'
flextable(as.data.frame(quantiles[, c(2, 1)]))
```

rank	propen
0%	-0.0018
20%	0.0981
40%	0.1373
60%	0.2710
80%	0.8514
100%	1.6134

```
# Create a propensity group variable using these quintiles.
diabstat2$propengp <- cut(diabstat2$propen,
                          quantile(diabstat2$propen, prob=quintiles),
                          include.lowest=TRUE,
                          labels=1:5)

# Display the frequencies.
xtabs(~propengp+statin, data=diabstat2)

# And as a flextable.
flextable(as.data.frame(xtabs(~propengp+statin, data=diabstat2)))
```

propengp	statin	Freq
1	0	1,330
2	0	1,258
3	0	1,283
4	0	1,264
5	0	1,201
1	1	44
2	1	86
3	1	77
4	1	118
5	1	134

```
# Step 13: Effect modification by propensity score.
diabstat2$lopropen <- as.numeric(diabstat2$propengp)
diabstat2$lopropen[diabstat2$propengp %in% seq(3, 5)] <- 0
diabstat2$lopropen[diabstat2$propengp %in% seq(1, 2)] <- 1

diabstat2$intstlo <- diabstat2$statin * diabstat2$lopropen

cox5.13 <- coxph(Surv(surv3, dead3) ~ statin + lopropen + intstlo + sex +
                                      agem2 + agesqrt + stage2 + stage3 +
                                      anycom + macssln + seifa25 + aria5,
                 data=diabstat2)
summary(cox5.13)


# Step 14: Cox regressions without the lopropen and intstlo variables for each
#          of the lopropen groups.
cox5.14a <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                       stage3 + anycom + macssln + seifa25 +
                                       aria5,
                  data=diabstat2[diabstat2$lopropen==1, ])
summary(cox5.14a)

cox5.14b <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                       stage3 + anycom + macssln + seifa25 +
                                       aria5,
                  data=diabstat2[diabstat2$lopropen==0, ])
summary(cox5.14b)
```

TRAINING SESSION 5: TIDYVERSE-R SYNTAX SOLUTIONS

```
# Preparation
# Start a new R session by opening the project.

# Load the libraries used in the exercise.
library(dplyr)
library(magrittr)
library(lubridate)
library(psych)
library(survival)


################################### Exercise 5 ###################################

#### Steps 1-2: Open PBSdata, tag and sequence statins, cutdown.

#### Step 1: Load the preprepared file and tag.
load('data/PBSdata3.RData')

statins <- c(1224, 1453, 1687, 1942, 2011, 2012, 2013, 2831, 2833, 2834, 2892,
             2967, 2978, 8023, 8024, 8173, 8197, 8213, 8214, 8215, 8303, 8304,
             8313, 8419, 8521, 8721, 8722, 8757)

PBSstatin <- PBSdata3 %>%
    mutate(anystatin=ifelse(pbsitem %in% statins, 1, 0))


#### Step 2: Sequence, cutdown and save.
PBSstatin <- PBSstatin %>%
    arrange(rootlpno, -anystatin) %>%
    group_by(rootlpno, anystatin) %>%
    mutate(statseq=ifelse(anystatin==1, row_number(), 0)) %>%
    ungroup()

PBSstatin <- PBSstatin %>%
    filter(statseq==1) %>% select(rootlpno, statdate=dispdate)
nrow(PBSstatin)
save(PBSstatin, file='data/PBSstatin.RData')


#### Step 3: Open diabetes platform, load statin date, exclude patients with
####         prior statins, classify.
load('data/diabetes8.RData')

diabetes9 <- diabetes8 %>% merge(PBSstatin, all.x=TRUE) %>%
    mutate(statin=0) %>%
    mutate(statin=ifelse(!is.na(statdate) & (statdate<date), 9, statin)) %>%
    mutate(statin=ifelse((statin!=9) & (!is.na(statdate)) & (statdate-date<=364), 1,
statin))
table(diabetes9$statin)
diabetes9 <- diabetes9 %>% filter(statin!=9)
nrow(diabetes9)
```

```
#### Steps 4-6: Preparation of Additional Covariates.

#### Step 4: Select records in 1991 or later.
diabetes9 <- diabetes9 %>%
    mutate(year=year(date)) %>%
    filter(year>=1991)
nrow(diabetes9)
save(diabetes9, file='data/diabetes9.RData')

#### Step 5: Append additional covariates to the platform file.
load('data/diabsupp.RData')
diabetes10 <- diabetes9 %>% merge(diabsupp, all.x=TRUE)

save(diabetes10, file='data/diabetes10.RData')

#### Step 6: Create stage variable.
diabetes10 <- diabetes10 %>%
    mutate(stage=ifelse(stgpt1==0 & stgpt2==0 & stgpt3>0, 3,
                        ifelse(stgpt1==0 & stgpt2>0, 2, 1)))
table(diabetes10$stage)

#### Step 7: Create survival variables for follow-up truncated at 3 years and
####         records before 1998.
diabstat <- diabetes10 %>%
    mutate(surv3=exit-date, dead3=dead) %>%
    mutate(dead3=ifelse(surv3>1095, 0, dead)) %>%
    mutate(surv3=as.numeric(ifelse(surv3>1095, 1095, surv3))) %>%
    filter(year<1998) %>%
    select(rootlpno, sex, age, statin, year, seifagp, ariagp, macss, stage,
           surv3, dead3)
nrow(diabstat)
save(diabstat, file='data/diabstat.RData')

#### Step 8: Bivariate Cox regression analysis.
cox5.8 <- coxph(Surv(surv3, dead3)~statin, data=diabstat)
summary(cox5.8)
```

```
Console   Terminal   Jobs
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> summary(cox5.8)
Call:
coxph(formula = Surv(surv3, dead3) ~ statin, data = diabstat)

  n= 6795, number of events= 709

          coef exp(coef) se(coef)      z Pr(>|z|)
statin -0.4969    0.6084   0.1866 -2.663  0.00774 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

       exp(coef) exp(-coef) lower .95 upper .95
statin    0.6084      1.644    0.4221     0.877

Concordance= 0.513  (se = 0.004 )
Likelihood ratio test= 8.28  on 1 df,   p=0.004
Wald test            = 7.09  on 1 df,   p=0.008
Score (logrank) test = 7.24  on 1 df,   p=0.007

>
```

```
#### Step 9: Optimising Covariates for Multivariate Regression Model.
diabstat2 <- diabstat %>%
    mutate(sex=ifelse(sex==2, 0, sex)) %>%
    mutate(sex=ifelse(is.na(sex), 0.5, sex)) %>%
    mutate(age=ifelse(is.na(age), 59.2, age))

diabstat2 <- diabstat2 %>%
    mutate(agem2=(1/age^2), agesqrt=(sqrt(age))) %>%
    mutate(anycom=ifelse(macss>0, 1, 0))  %>%
    mutate(macssln=(log(macss+0.01))) %>%
    mutate(stage2=ifelse(stage==2, 1, 0), stage3=ifelse(stage==3, 1, 0)) %>%
    mutate(seifa25=ifelse(seifagp==1, 0,
                          ifelse(seifagp %in% 2:5, 1, seifagp))) %>%
    mutate(aria5=ifelse(ariagp %in% 1:4, 0,
                        ifelse(ariagp==5, 1, ariagp)))

cox5.9 <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                     stage3 + anycom + macssln + seifa25 + aria5,
                data=diabstat2)
summary(cox5.9)
```

```
Console   Terminal ×   Jobs ×
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> summary(cox5.9)
Call:
coxph(formula = Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt +
    stage2 + stage3 + anycom + macssln + seifa25 + aria5, data = diabstat2)

  n= 6795, number of events= 709

              coef  exp(coef)   se(coef)      z Pr(>|z|)
statin    -0.31560    0.72935    0.18802 -1.679  0.09325 .
sex        0.43536    1.54553    0.07669  5.677 1.37e-08 ***
agem2   2181.33957        Inf  437.91216  4.981 6.32e-07 ***
agesqrt    1.46506    4.32780    0.09194 15.934  < 2e-16 ***
stage2    -0.10794    0.89768    0.09733 -1.109  0.26740
stage3     0.31203    1.36620    0.17832  1.750  0.08013 .
anycom    -0.96986    0.37914    0.36486 -2.658  0.00786 **
macssln    0.54485    1.72436    0.08831  6.170 6.83e-10 ***
seifa25    0.24523    1.27792    0.09790  2.505  0.01225 *
aria5      0.40334    1.49681    0.18980  2.125  0.03358 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

        exp(coef) exp(-coef) lower .95 upper .95
statin     0.7294     1.3711    0.5045    1.0543
sex        1.5455     0.6470    1.3298    1.7962
agem2         Inf     0.0000       Inf       Inf
agesqrt    4.3278     0.2311    3.6141    5.1824
stage2     0.8977     1.1140    0.7418    1.0863
stage3     1.3662     0.7320    0.9632    1.9377
anycom     0.3791     2.6376    0.1855    0.7751
macssln    1.7244     0.5799    1.4503    2.0502
seifa25    1.2779     0.7825    1.0548    1.5482
aria5      1.4968     0.6681    1.0318    2.1713

Concordance= 0.825  (se = 0.007 )
Likelihood ratio test= 1083  on 10 df,   p=<2e-16
Wald test            = 997.4  on 10 df,   p=<2e-16
Score (logrank) test = 1276  on 10 df,   p=<2e-16

>
```

```
save(diabstat2, file='data/diabstat2.RData')


#### Step 10: Using time as an instrumental variable.
# Assumes year is a numeric value not a factor.
xtabs(~year+statin, data=diabstat2)
rowPerc(xtabs(~year+statin, data=diabstat2))

# In a tidy table format for display in the viewer.
df1 <- as.data.frame(xtabs(~year+statin, data=diabstat2))
df2 <- as.data.frame(rowPerc(xtabs(~year+statin, data=diabstat2)))
df1 <- pivot_wider(df1, names_from=statin,
    values_from=Freq, names_prefix="n_statin_")
df2 <- pivot_wider(df2, names_from=statin,
    values_from=Freq, names_prefix="pc_statin_")
df <- merge(df1, df2, by='year') %>% select(-pc_statin_Total) %>%
    relocate(year, n_statin_0, pc_statin_0, n_statin_1, pc_statin_1)
flextable(df)
rm(df, df1, df2)
```

year	n_statin_0	pc_statin_0	n_statin_1	pc_statin_1
1991	1,038	98	21	2.0
1992	1,361	96	61	4.3
1993	899	95	43	4.6
1994	826	95	48	5.5
1995	793	91	79	9.1
1996	773	87	112	12.7
1997	646	87	95	12.8

```
#### Step 11: Bivariate Cox regression analysis of year on mortality.
# Need to convert year to a factor rather than a numeric value.
cox5.11 <- coxph(Surv(surv3, dead3)~as.factor(year), data=diabstat2)
summary(cox5.11)
```

```
Console   Terminal ×   Jobs ×
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> summary(cox5.11)
Call:
coxph(formula = Surv(surv3, dead3) ~ as.factor(year), data = diabstat2)

  n= 6795, number of events= 709

                         coef exp(coef)  se(coef)      z Pr(>|z|)
as.factor(year)1992  0.008384  1.008419  0.111863  0.075 0.940254
as.factor(year)1993 -0.176702  0.838029  0.129652 -1.363 0.172917
as.factor(year)1994 -0.356475  0.700140  0.139787 -2.550 0.010768 *
as.factor(year)1995 -0.480854  0.618255  0.145202 -3.312 0.000928 ***
as.factor(year)1996 -0.413337  0.661440  0.141475 -2.922 0.003482 **
as.factor(year)1997 -0.565865  0.567869  0.169239 -3.344 0.000827 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                    exp(coef) exp(-coef) lower .95 upper .95
as.factor(year)1992    1.0084     0.9917    0.8099    1.2556
as.factor(year)1993    0.8380     1.1933    0.6500    1.0805
as.factor(year)1994    0.7001     1.4283    0.5324    0.9208
as.factor(year)1995    0.6183     1.6175    0.4651    0.8218
as.factor(year)1996    0.6614     1.5119    0.5013    0.8728
as.factor(year)1997    0.5679     1.7610    0.4076    0.7912

Concordance= 0.558  (se = 0.01 )
Likelihood ratio test= 31.88  on 6 df,   p=2e-05
Wald test            = 31.23  on 6 df,   p=2e-05
Score (logrank) test = 31.76  on 6 df,   p=2e-05

>
```

```
#### Step 12: Derivation of propensity ranks
logistic <- glm(statin ~ sex + age + stage2 + stage3 + macss + anycom,
                data=diabstat2, family="binomial")
summary(logistic)
```

```
Console   Terminal ×   Jobs ×
~/SAIL/HDS/Modules/PMIM602/2021/2021_PMIM-602/R_Solutions/
> summary(logistic)

Call:
glm(formula = statin ~ sex + age + stage2 + stage3 + macss +
    anycom, family = "binomial", data = diabstat2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6557  -0.4467  -0.3294  -0.3202   2.4675

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.041324   0.208015 -14.621  < 2e-16 ***
sex         -0.014311   0.098021  -0.146    0.884
age          0.002002   0.003240   0.618    0.537
stage2       0.745549   0.106703   6.987 2.81e-12 ***
stage3       0.179164   0.215765   0.830    0.406
macss        0.388520   0.252889   1.536    0.124
anycom      -0.109320   0.178366  -0.613    0.540
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3360.2  on 6794  degrees of freedom
Residual deviance: 3307.0  on 6788  degrees of freedom
AIC: 3321

Number of Fisher Scoring iterations: 5

>
```

```
# The coefficients from the model are found in coef(logistic[n]).
diabstat2 <- diabstat2 %>%
    mutate(propen = coef(logistic)[2]*diabstat2$sex +
                    coef(logistic)[3]*diabstat2$age +
                    coef(logistic)[4]*diabstat2$stage2 +
                    coef(logistic)[5]*diabstat2$stage3 +
                    coef(logistic)[6]*diabstat2$macss +
                    coef(logistic)[7]*diabstat2$anycom)

describe(diabstat2$propen)

# you can see that the digits are slightly different to the SPSS, SAS and Stata
# solutions, which does lead to slightly different numbers later on
quantile(diabstat2$propen, prob=c(.2,.4,.6,.8,1))

diabstat2$propengp <- cut(diabstat2$propen,
                          quantile(diabstat2$propen, prob=c(0,.2,.4,.6,.8,1)),
                          include.lowest=TRUE,
                          labels = 1:5)
```

```
xtabs(~propengp+statin, data=diabstat2)
```

rank	propen
0%	-0.0018
20%	0.0981
40%	0.1373
60%	0.2710
80%	0.8514
100%	1.6134

propengp	statin	Freq
1	0	1,330
2	0	1,258
3	0	1,283
4	0	1,264
5	0	1,201
1	1	44
2	1	86
3	1	77
4	1	118
5	1	134

```
#### Steps 13-14: Effect Modification by Propensity Scores

#### Step 13: Assess the effect modification by propensity score using an
####          interaction term
diabstat2 <- diabstat2 %>%
    mutate(lopropen=as.numeric(propengp)) %>%
    mutate(lopropen=ifelse(propengp %in% 3:5, 0,
                           ifelse(propengp %in% 1:2, 1, propengp))) %>%
    mutate(intstlo=statin*lopropen)

cox5.13 <- coxph(Surv(surv3, dead3) ~ statin + lopropen + intstlo + sex +
                                      agem2 + agesqrt + stage2 + stage3 +
                                      anycom + macssln + seifa25 + aria5,
                 data=diabstat2)
summary(cox5.13)


#### Step 14: Assess the effect modification by propensity score using
####          stratum-specific analysis.
cox5.14a <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                       stage3 + anycom + macssln + seifa25 +
                                       aria5,
                  data=diabstat2 %>% filter(lopropen==1))
summary(cox5.14a)

cox5.14b <- coxph(Surv(surv3, dead3) ~ statin + sex + agem2 + agesqrt + stage2 +
                                       stage3 + anycom + macssln + seifa25 +
                                       aria5,
                  data=diabstat2 %>% filter(lopropen==0))
summary(cox5.14b)
```