Appendix A simulation 1: imputing multiple missing predictor values scenarios using local data

#	Variables missing	Methods	MSE of the LP (% difference to M-Imp)	C-index	CITL	Calibration slope
	Apparent performance (reference)			0.7051	-0.0001	0.9999
1	(1) SBP, (2) smoking	M-Imp	0.0702	0.6908	0.0228	0.9415
		JMI	0.0685 (-2.35%)	0.6913	0.0242	0.9552
		JMI^{aux}	0.0649 (-7.50%)	0.6975	0.0221	0.9928
2	(1) TC, (2) HDL-c	M-Imp	0.0265	0.7005	0.0230	0.9766
		JMI	0.0241(-8.97%)	0.7013	0.0192	0.9751
		JMI^{aux}	0.0220 (-16.84%)	0.7046	0.0153	0.9901
3	(1) TC, (2) HDL-c, (3) SBP	M-Imp	0.0333	0.6994	0.0406	1.0003
		JMI	0.0314 (-5.74%)	0.7000	0.0268	0.9787
		JMI^{aux}	0.0276 (-17.26%)	0.7040	0.0212	0.9935
4	(1) TC, (2) HDL-c, (3) SBP, (4) AD	M-Imp	0.0459	0.6981	0.0546	1.0217
		JMI	0.0425 (-7.41%)	0.6983	0.0255	0.9702
		JMI^{aux}	0.0394 (-14.13%)	0.7044	0.0231	1.0010
5	(1) TC, (2) HDL-c, (3) AD (4) smoking, (5) DM	M-Imp	0.1300	0.6797	0.0581	0.9199
		JMI	0.1262 (-2.98%)	0.6803	0.0549	0.9211
		JMI^{aux}	0.1014 (-21.98%)	0.6960	0.0369	1.0052
6	(1) TC, (2) HDL-c, (3) AD, (4) smoking, (5) DM, (6) SBP	M-Imp	0.1383	0.6785	0.0758	0.9430
		JMI	0.1351 (-2.31%)	0.6788	0.0637	0.9249
		JMI^{aux}	0.1087 (-21.45%)	0.6955	0.0441	1.0128
7	(1) Age, (2) gender, (3) TC, (4) HDL-c, (5) AD, (6) smoking, (7) DM, (8) SBP	M-Imp	0.9137	0.5112	0.2897	77.819
		JMI	0.9137 (0.00%)	0.5112	0.2897	77.819
		JMI^{aux}	0.5990 (-34.44%)	0.6892	0.1544	1.3754
8	(1) Age, (2) gender	M-Imp	0.7438	0.6063	0.1958	0.8225
		JMI	0.6373 (-14.32%)	0.6223	0.1616	0.8052
		JMI^{aux}	0.4517 (-39.26%)	0.6931	0.0794	1.0828

Legend – MSE: mean squared error, LP: linear predictor, CITL: calibration in the large, M-Imp: mean imputation, JMI: joint modelling imputation, JMI^{aux}: joint modelling imputation with auxiliary variables, SBP: systolic blood pressure, TC: total cholesterol, HDL-c: HDL-cholesterol, AD: antihypertensive drug, DM: diabetes mellitus.

Appendix B simulation 2: imputing multiple missing predictor values scenarios using external data

#	Variables missing	Methods	MSE of the LP (% difference to M-Imp)	C-index	CITL	Calibration slope
	Apparent performance (reference)			0.7051	-0.0001	0.9999
1	(1) SBP, (2) smoking	M-Imp	0.0802	0.6908	0.1227	0.9415
		JMI	0.0782 (-2.56%)	0.6911	0.1018	0.9269
		JMI^{aux}	0.0801 (0.001%)	0.6902	0.1123	0.9251
2	(1) TC, (2) HDL-c	M-Imp	0.0280	0.7005	0.0618	0.9766
		JMI	0.0251 (-10.13%)	0.7010	0.0244	0.9639
		JMI^{aux}	0.0248 (-11.35%)	0.7017	0.0155	0.9724
3	(1) TC, (2) HDL-c, (3) SBP	M-Imp	0.0343	0.6994	0.0715	1.0003
		JMI	0.0325 (-5.14%)	0.6995	0.0308	0.9629
		JMI^{aux}	0.0324 (-5.29%)	0.7003	0.0192	0.9703
4	(1) TC, (2) HDL-c, (3) SBP, (4) AD	M-Imp	0.0654	0.6982	0.1943	1.0217
		JMI	0.0615 (-5.95%)	0.6978	0.1661	0.9756
		JMI^{aux}	0.0583 (-10.91%)	0.6998	0.1530	0.9881
5	(1) TC, (2) HDL-c, (3) AD (4) smoking, (5) DM	M-Imp	0.1930	0.6797	0.3090	0.9199
		JMI	0.1806 (-6.42%)	0.6803	0.2797	0.9067
		JMI^{aux}	0.1591 (-17.57%)	0.6844	0.2595	0.9475
6	(1) TC, (2) HDL-c, (3) AD, (4) smoking, (5) DM, (6) SBP	M-Imp	0.1974	0.6785	0.3189	0.9431
		JMI	0.1914 (-3.01%)	0.6788	0.2889	0.9045
		JMI^{aux}	0.1664 (-15.70%)	0.6831	0.2663	0.9483
7	(1) Age, (2) gender, (3) TC, (4) HDL-c, (5) AD, (6) smoking, (7) DM, (8) SBP	M-Imp	0.9167	0.5157	0.2332	86.984
		JMI	0.9167 (0.00%)	0.5157	0.2332	86.984
		JMI^{aux}	0.7269 (-20.70%)	0.6589	-0.0783	0.9687
8	(1) Age, (2) gender	M-Imp	0.8334	0.6064	-0.1037	0.8230
		JMI	0.7963 (-4.45%)	0.6116	-0.2221	0.5769
		JMI^{aux}	0.7018 (-15.79%)	0.6721	-0.3649	0.8453

Legend – MSE: mean squared error, LP: linear predictor, CITL: calibration in the large, M-Imp: mean imputation, JMI: joint modelling imputation, JMI^{aux}: joint modelling imputation with auxiliary variables, SBP: systolic blood pressure, TC: total cholesterol, HDL-c: HDL-cholesterol, AD: antihypertensive drug, DM: diabetes mellitus.

Appendix C imputing multiple missing predictor values scenarios using enriched external data; simulation 3: scenario 1

#	Variables missing	Methods	MSE of the LP (% difference to M-Imp)	C-index	CITL	Calibration slope
	Apparent performance			0.7051	-0.0001	0.9999
1	MD scenario 1* (i.e. without local data, reference)	M-Imp	0.0802	0.6908	0.1227	0.9415
		JMI	0.0782 (-2.56%)	0.6911	0.1018	0.9269
		JMI+	0.0801 (0.001%)	0.6902	0.1123	0.9251
2	+100 local patients	M-Imp	0.0794	0.6908	0.1169	0.9402
		JMI	0.0769 (-3.25%)	0.6912	0.0915	0.9252
		JMI+	0.0780 (-1.79%)	0.6904	0.0987	0.9246
3	+300 local patients	M-Imp	0.0792	0.6902	0.1190	0.9393
		JMI	0.0763 (-3.80%)	0.6904	0.0910	0.9242
		JMI+	0.0771 (-2.72%)	0.6897	0.0966	0.9229
4	+750 local patients	M-Imp	0.0765	0.6902	0.1091	0.9396
		JMI	0.0733 (-4.37%)	0.6904	0.0710	0.9233
		JMI+	0.0725 (-5.52%)	0.6902	0.0693	0.9268
5	+1500 local patients	M-Imp	0.0746	0.6909	0.0845	0.9393
		JMI	0.0718 (-3.90%)	0.6913	0.0510	0.9278
		JMI+	0.0708 (-5.37%)	0.6911	0.0485	0.9315
6	+5000 local patients	M-Imp	0.0713	0.6869	0.0779	0.9420
		JMI	0.0699 (-2.00%)	0.6874	0.0545	0.9389
		JMI+	0.0683 (-4.39%)	0.6875	0.0482	0.9464
7	+10000 local patients	M-Imp	0.0704	0.6732	0.0863	0.8778
		JMI	0.0686 (-2.65%)	0.6733	0.0743	0.8797
		JMI+	0.0671 (-4.92%)	0.6739	0.0681	0.8887

Legend – MSE: mean squared error, LP: linear predictor, CITL: calibration in the large, M-Imp: mean imputation, JMI: joint modelling imputation, JMIaux: joint modelling imputation with auxiliary variables, *(1) systolic blood pressure, (2) smoking.

Appendix C (cont.) simulation 3: scenario 5

#	Variables missing	Methods	MSE of the LP (% difference to M-Imp)	C-index	CITL	Calibration slope
	Apparent performance			0.7051	-0.0001	0.9999
1	MD scenario 5* (i.e. without local data, reference)	M-Imp	0.1974	0.6785	0.3189	0.9431
		JMI	0.1914 (-3.01%)	0.6788	0.2889	0.9045
		JMI+	0.1664 (-15.70%)	0.6831	0.2663	0.9483
2	+100 local patients	M-Imp	0.1929	0.6785	0.3081	0.9418
		JMI	0.1865 (-3.33%)	0.6788	0.2775	0.9046
		JMI+	0.1605 (-16.81%)	0.6836	0.2509	0.9502
3	+300 local patients	M-Imp	0.1899	0.6785	0.3054	0.9464
		JMI	0.1827 (-3.78%)	0.6788	0.2749	0.9114
		JMI+	0.1557 (-18.01%)	0.6834	0.2445	0.9557
4	+750 local patients	M-Imp	0.1794	0.6787	0.2851	0.9506
		JMI	0.1714 (-4.42%)	0.6790	0.2521	0.9167
		JMI+	0.1425 (-20.56%)	0.6844	0.2105	0.9607
5	+1500 local patients	M-Imp	0.1683	0.6790	0.2418	0.9387
		JMI	0.1603 (-4.78%)	0.6792	0.2078	0.9095
		JMI+	0.1334 (-20.72%)	0.6851	0.1677	0.9573
6	+5000 local patients	M-Imp	0.1472	0.6736	0.1960	0.9377
		JMI	0.1424 (-3.31%)	0.6737	0.1696	0.9166
		JMI+	0.1206 (-18.11%)	0.6789	0.1292	0.9585
7	+10000 local patients	M-Imp	0.1454	0.6630	0.1832	0.8820
		JMI	0.1410 (-3.06%)	0.6629	0.1603	0.8668
		JMI+	0.1199 (-17.54%)	0.6703	0.1291	0.9201

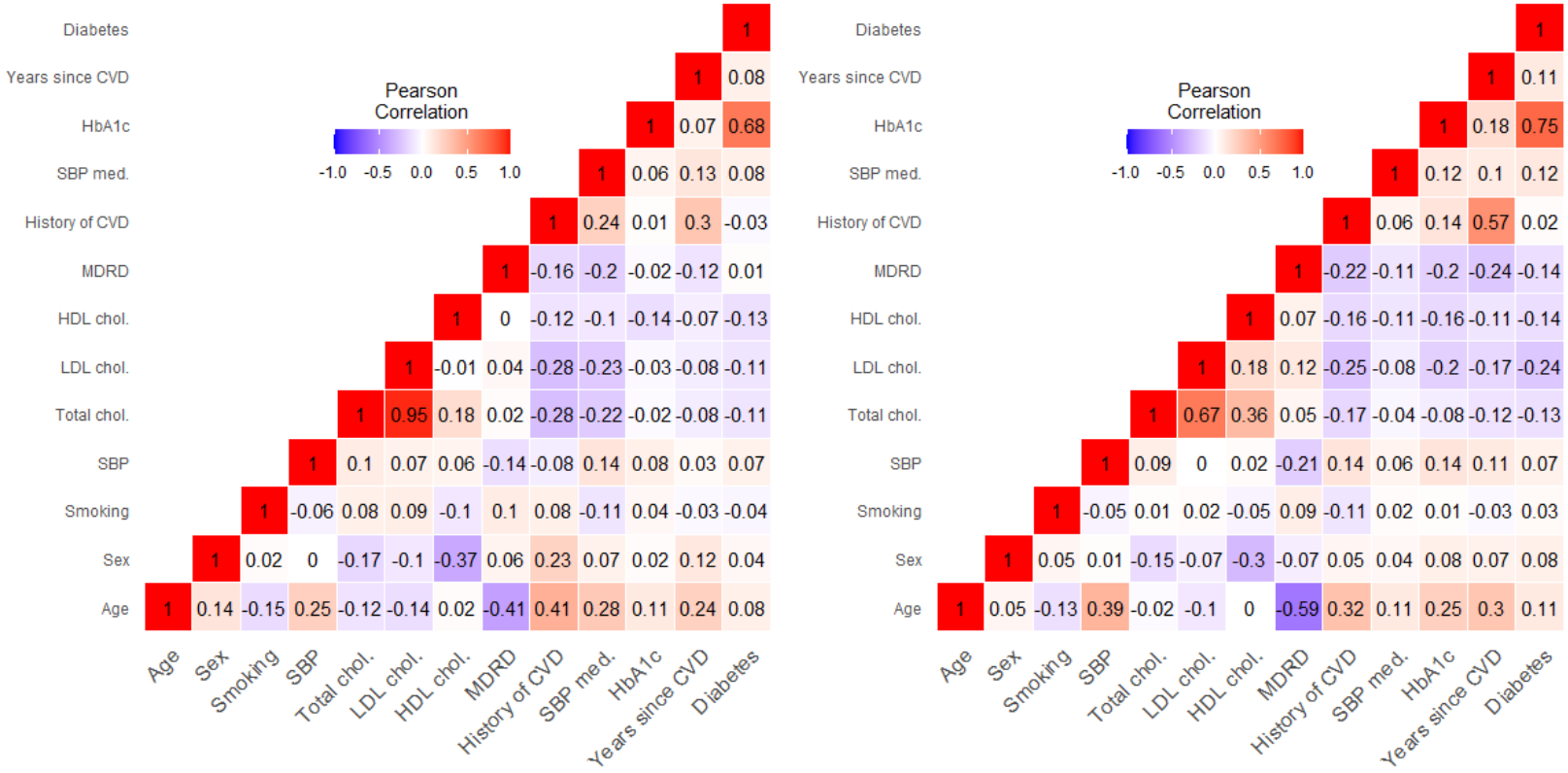
Legend – MSE: mean squared error, LP: linear predictor, CITL: calibration in the large, M-Imp: mean imputation, JMI: joint modelling imputation, JMI^{aux}: joint modelling imputation with auxiliary variables, *(1) systolic blood pressure, (2) total cholesterol, (3) HDL-cholesterol, (4) smoking, (5) antihypertensive drugs, (6) Diabetes mellitus.

Appendix C (cont.) simulation 3: scenario 8

#	Variables missing	Methods	MSE of the LP (% difference to M-Imp)	C-index	CITL	Calibration slope
	Apparent performance			0.7051	-0.0001	0.9999
1	MD scenario 8* (i.e. without local data, reference)	M-Imp	0.8334	0.6064	-0.1037	0.8230
		JMI	0.7963 (-4.45%)	0.6116	-0.2221	0.5769
		JMI+	0.7018 (-15.79%)	0.6721	-0.3649	0.8453
2	+100 local patients	M-Imp	0.8281	0.6064	-0.0965	0.8178
		JMI	0.7787 (-6.34%)	0.6122	-0.1910	0.5893
		JMI+	0.6764 (-22.43%)	0.6733	-0.3225	0.8608
3	+300 local patients	M-Imp	0.8173	0.6044	-0.0803	0.8118
		JMI	0.7677 (-6.46%)	0.6103	-0.1662	0.5841
		JMI+	0.6510 (-25.55%)	0.6734	-0.2786	0.8665
4	+750 local patients	M-Imp	0.8089	0.6050	-0.0540	0.8005
		JMI	0.7455 (-8.50%)	0.6108	-0.1166	0.6132
		JMI+	0.6117 (-32.24%)	0.6778	-0.2138	0.9232
5	+1500 local patients	M-Imp	0.7923	0.6107	-0.0205	0.8429
		JMI	0.7253 (-9.24%)	0.6131	-0.0659	0.6480
		JMI+	0.5740 (-38.03%)	0.6856	-0.1451	0.9654
6	+5000 local patients	M-Imp	0.7461	0.6144	0.0905	0.9096
		JMI	0.6695 (-11.44%)	0.6202	0.0387	0.7332
		JMI+	0.5094 (-46.47%)	0.6914	-0.0503	1.0142
7	+10000 local patients	M-Imp	0.7560	0.6024	0.1577	0.7556
		JMI	0.6734 (-12.27%)	0.6065	0.1049	0.6707
		JMI+	0.4999 (-51.23%	0.6973	0.0328	1.0664

Legend – MSE: mean squared error, LP: linear predictor, CITL: calibration in the large, M-Imp: mean imputation, JMI: joint modelling imputation, JMI^{aux}: joint modelling imputation with auxiliary variables, *(1) age, (2) gender.

Appendix D: Correlation matrix (with additional patient variables) – left: local data (SMART), right: external data (UCC)



Appendix E – R code

The completed UCC-CVRM data is available from *knn1*. The completed UCC-SMART data is available from *smart*. These data frames were used in the various simulation studies as follows:

```
load("knn1.RData")
load("smart.RData")
source("functions.r")

# test imputation model
frh_vars <-
c("leeftijd","geslacht","labchol","labhdl","bdsys","mht_all","roken","vz_
DM")
testMSE(ds[,frh_vars],missing_var="labchol",n.imp=1,method="internal",see
d=1221)

apparent_performance(data=data_imp) # c = 0.7061156, intercept = -
0.0000517122, slope = 0.9999569, eo = 1.000052

# optimism corrected performance
# age_gender_perf <- oc_performance(data=smart)
oc_performance(data=smart)

apparent_performance(data=smart)
oc_performance(data=smart)

################################################
# 1. Simulation study without ldl cholesterol #
################################################

smart.chol <- smart
smart.chol[,"ldlchol"] <- NULL

ucc.chol <- ucc
ucc.chol[,"ldlchol"] <- NULL

scenarios <- list(scen1 = c("labchol"),
         scen2 = c("labhdl"),
         scen3 = c("labchol","labhdl"),
         scen4 = c("labchol","labhdl","roken","mht_all","vz_DM"),
         scen5 = c("labchol","labhdl","bdsys","mht_all","roken","vz_DM"),
         scen6 = c("labchol","labhdl","bdsys"),
         scen7 = c("labchol","labhdl","bdsys","mht_all"),
         scen8 =
c("leeftijd","geslacht","labchol","labhdl","bdsys","mht_all","roken","vz_
DM"))

sim_1_smart <- run_simulation(ref_data  = smart.chol,
               scenarios = scenarios,
               validation = "jackknife",
               seed    = 12345)

sim_1_ucc  <- run_simulation(ref_data  = smart.chol,
               ext_data  = ucc.chol,
               scenarios = scenarios,
               validation = "external",
               seed    = 12345)
```

```
############################
# Results simulation study #
############################

eval_jk(results=sim_1_smart,imputation_method=1,pattern=1)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=1)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=1)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=1)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=1)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=1)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=2)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=2)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=2)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=2)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=2)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=2)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=3)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=3)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=3)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=3)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=3)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=3)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=4)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=4)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=4)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=4)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=4)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=4)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=5)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=5)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=5)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=5)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=5)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=5)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=6)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=6)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=6)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=6)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=6)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=6)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=7)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=7)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=7)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=7)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=7)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=7)

eval_jk(results=sim_1_smart,imputation_method=1,pattern=8)
eval_jk(results=sim_1_smart,imputation_method=2,pattern=8)
eval_jk(results=sim_1_smart,imputation_method=3,pattern=8)
eval_jk(results=sim_1_ucc,imputation_method=1,pattern=8)
eval_jk(results=sim_1_ucc,imputation_method=2,pattern=8)
eval_jk(results=sim_1_ucc,imputation_method=3,pattern=8)
```

```
################################################
# 2. Simulation study with remaining scenarios #
################################################

scenarios <- list(scen1 = c("bdsys"),
         scen2 = c("mht_all"),
         scen3 = c("roken"),
         scen4 = c("vz_DM"),
         scen5 = c("roken","mht_all"),
         scen6 = c("bdsys","mht_all","roken"),
         scen7 = c("leeftijd","geslacht"))

sim_2_smart <- run_simulation(ref_data  = smart,
               scenarios = scenarios,
               validation = "jackknife",
               seed    = 12345)
sim_2_ucc  <- run_simulation(ref_data  = smart,
               ext_data  = ucc,
               scenarios = scenarios,
               validation = "external",
               seed    = 12345)

############################
# Results simulation study #
############################

eval_jk(results=sim_2_smart,imputation_method=1,pattern=1)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=1)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=1)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=1)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=1)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=1)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=2)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=2)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=2)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=2)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=2)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=2)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=3)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=3)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=3)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=3)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=3)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=3)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=4)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=4)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=4)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=4)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=4)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=4)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=5)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=5)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=5)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=5)
```

```
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=5)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=5)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=6)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=6)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=6)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=6)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=6)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=6)

eval_jk(results=sim_2_smart,imputation_method=1,pattern=7)
eval_jk(results=sim_2_smart,imputation_method=2,pattern=7)
eval_jk(results=sim_2_smart,imputation_method=3,pattern=7)
eval_jk(results=sim_2_ucc,imputation_method=1,pattern=7)
eval_jk(results=sim_2_ucc,imputation_method=2,pattern=7)
eval_jk(results=sim_2_ucc,imputation_method=3,pattern=7)

 ##############################################
# 3. simulation study with enriched UCC data #
##############################################

# when chol is part of the scenario:
smart.chol <- smart
ucc.chol <- ucc
smart.chol[,"ldlchol"] <- NULL
ucc.chol[,"ldlchol"] <- NULL

# separate common variables in smart & ucc
smart_comvar <- smart.chol[,which(names(smart.chol) %in%
names(ucc.chol))]
ucc_comvar <- ucc.chol[,which(names(ucc.chol) %in% names(smart.chol))]

# set aside different lenghts 'local' data from smart
set.seed(12345)
add1 <- smart_comvar[sample(nrow(smart_comvar),100),]
add2 <- smart_comvar[sample(nrow(smart_comvar),300),]
add3 <- smart_comvar[sample(nrow(smart_comvar),750),]
add4 <- smart_comvar[sample(nrow(smart_comvar),1500),]
add5 <- smart_comvar[sample(nrow(smart_comvar),5000),]
add6 <- smart_comvar[sample(nrow(smart_comvar),10000),]

# enrich ucc data with different lenghts 'local' data from smart
ucc1 <- rbind(ucc_comvar,add1)
ucc2 <- rbind(ucc_comvar,add2)
ucc3 <- rbind(ucc_comvar,add3)
ucc4 <- rbind(ucc_comvar,add4)
ucc5 <- rbind(ucc_comvar,add5)
ucc6 <- rbind(ucc_comvar,add6)

# remove data used for enrichment from smart
smart1 <- smart_comvar[-as.numeric(rownames(add1)),]
smart2 <- smart_comvar[-as.numeric(rownames(add2)),]
smart3 <- smart_comvar[-as.numeric(rownames(add3)),]
smart4 <- smart_comvar[-as.numeric(rownames(add4)),]
smart5 <- smart_comvar[-as.numeric(rownames(add5)),]
smart6 <- smart_comvar[-as.numeric(rownames(add6)),]

# add back time/status variables
smart1$time <- smart[as.numeric(rownames(smart1)),"time"]
smart1$status <- smart[as.numeric(rownames(smart1)),"status"]
```

```
smart2$time <- smart[as.numeric(rownames(smart2)),"time"]
smart2$status <- smart[as.numeric(rownames(smart2)),"status"]
smart3$time <- smart[as.numeric(rownames(smart3)),"time"]
smart3$status <- smart[as.numeric(rownames(smart3)),"status"]
smart4$time <- smart[as.numeric(rownames(smart4)),"time"]
smart4$status <- smart[as.numeric(rownames(smart4)),"status"]
smart5$time <- smart[as.numeric(rownames(smart5)),"time"]
smart5$status <- smart[as.numeric(rownames(smart5)),"status"]
smart6$time <- smart[as.numeric(rownames(smart6)),"time"]
smart6$status <- smart[as.numeric(rownames(smart6)),"status"]

rm(add1,add2,add3,add4,add5,add6,smart_comvar,ucc_comvar,smart.chol,ucc.c
hol)

# scenarios with chol
scenarios <- list(scen1 =
c("bdsys","labchol","labhdl","roken","mht_all","vz_DM"))

# scenarios without chol
scenarios <- list(scen1 = c("bdsys","roken"))

sim_3_enri1  <- run_simulation(ref_data = smart1,
                ext_data  = ucc1,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
sim_3_enri2  <- run_simulation(ref_data = smart2,
                ext_data  = ucc2,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
sim_3_enri3  <- run_simulation(ref_data = smart3,
                ext_data  = ucc3,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
sim_3_enri4  <- run_simulation(ref_data = smart4,
                ext_data  = ucc4,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
sim_3_enri5  <- run_simulation(ref_data = smart5,
                ext_data  = ucc5,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
sim_3_enri6  <- run_simulation(ref_data = smart6,
                ext_data  = ucc6,
                model_data = smart,
                scenarios = scenarios,
                validation = "enrich",
                seed    = 12345)
save(sim_3_enri4, file="simulation_3_age_gender.RData")

############################
```

```
# Results simulation study #
############################

eval_jk(results=sim_3_enri1,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri1,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri1,imputation_method=3,pattern=1)

eval_jk(results=sim_3_enri2,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri2,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri2,imputation_method=3,pattern=1)

eval_jk(results=sim_3_enri3,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri3,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri3,imputation_method=3,pattern=1)

eval_jk(results=sim_3_enri4,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri4,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri4,imputation_method=3,pattern=1)

eval_jk(results=sim_3_enri5,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri5,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri5,imputation_method=3,pattern=1)

eval_jk(results=sim_3_enri6,imputation_method=1,pattern=1)
eval_jk(results=sim_3_enri6,imputation_method=2,pattern=1)
eval_jk(results=sim_3_enri6,imputation_method=3,pattern=1)
```

The content of the file *functions.r* is as follows:

```
############
# Authors #
############

# Steven Nijman
# Jeroen Hoogland
# Thomas Debray

########################
# Package Requirements #
########################

library(foreign)
library(mice)    # version 3.6.0
library(condMVNorm) # version 2015.2-1
library(survival)

########################

#' Single patient joint imputation
#'
#' @author Steven W J Nijman \email{S.W.J.Nijman@@umcutrecht.nl}
#' @author Jeroen Hoogland  \email{J.Hoogland-2@@umcutrecht.nl}
#' @author Thomas P A Debray \email{T.Debray@@umcutrecht.nl}
#'
#' @param data single patient data
#' @param imp_means prior estimated column means of data
#' @param imp_cov prior estimated covariance matrix of data
#' @param n.imp number of imputations to make; default==1, which means it
uses expected values
```

```
#'
#' @return Returns single patient data with imputed values
impJoint <-
function(data=data,imp_means=imp_means,imp_cov=imp_cov,n.imp=1,...) {
 if (class(data) != "data.frame") {
  stop ("Data object should be a data frame")
 }
 if (nrow(data) > 1) {
  stop ("Data should contain a single patient")
 }
 if ("time" %in% colnames(data)) {
  stop ("No time info should be used for imputation")
 }
 if ("status" %in% colnames(data)) {
  stop ("No event info should be used for imputation")
 }

 dep  <- which(is.na(data))
 given <- which(!is.na(data))
 depnames <- colnames(data[dep])
 givennames <- colnames(data[given])
 missing.col <- which(is.na(data))
 data <- as.matrix(data)

 if(length(given)==0) {
  data_imp <- t(as.data.frame(imp_means))
 } else if(length(given)>0) {
  # Extract conditional Mean and conditional var
  cond <-
condMVN(mean=imp_means,sigma=imp_cov,dep=dep,given=given,X=data[given])

  if (n.imp == 1) {
   # Just impute the expected value if we only need 1 imputation
   x.imp <- matrix(cond$condMean, nrow = 1)
  } else if (n.imp < 1000) {
   # We should not use multiple imputation of the number of imputed
values is very low.
   # The empirical covariance of imputed values is very unreliable in
such circumstances
   stop ("A minimum of 1000 imputations should be generated when drawing
random samples, instead of using the conditional mean.")
  } else {
   # Draw from a multivariate normal if multiple imputations required
   x.imp <- mvrnorm(n = n.imp, mu = cond$condMean, Sigma = cond$condVar,
tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  }

  x.obs <- matrix(data[given], nrow=n.imp, ncol=length(given), byrow =
TRUE)
  data_imp <- as.data.frame(cbind(x.obs, x.imp))
  colnames(data_imp) <- c(givennames, depnames)
 }
 data.frame(data_imp)
}

# calculate mse of imputation methods
testMSE <-
function(data=data,missing_var=missing_var,n.imp=n.imp,method="jackknife"
,seed=12345,...) {
 jmimpdat  <- meanimpdat <- data
```

```
 jmimpdat[,] <- meanimpdat[,] <- NA

 if (!is.na(seed)) {
  set.seed(seed)
 }

 pb <- txtProgressBar(min = 0, max = nrow(data), style = 3)
 for (i in 1:nrow(data)) {
  setTxtProgressBar(pb, i)

  if (method == "jackknife") {
   training_data <- data[-i,]
   test_case   <- data[i,]
  } else if (method == "internal") {
   training_data <- data
   test_case   <- data[i,]
  } else if (method == "external") {
   training_data <- model_data
   test_case   <- data[i,]
  } else {
   stop ("Validation method not supported")
  }

  mu  <- colMeans(training_data)
  sigma <- cov(training_data)

  test_case[,missing_var] <- NA

  meanimpdat[i,] <- jmimpdat[i,] <- test_case
  meanimpdat[i,missing_var] <- mu[missing_var]
  jmimpdat[i,missing_var]  <-
impJoint(data=test_case,imp_means=mu,imp_cov=sigma)[missing_var]
 }
 close(pb)
 if(length(missing_var)==1) {
  result <- data.frame(mse_meanimp = mean((meanimpdat[,missing_var] -
data[,missing_var])**2),
             mse_jmimp  = mean((jmimpdat[,missing_var] -
data[,missing_var])**2))
 } else if(length(missing_var>=2)) {
  result <- data.frame(mse_meanimp = colMeans((meanimpdat[,missing_var] -
data[,missing_var])**2),
             mse_jmimp  = colMeans((jmimpdat[,missing_var] -
data[,missing_var])**2))
 }

 return(result)
}

# calibration intercept
cal_intercept <- function(model, data) {
 p <- log(predict(model, newdata = data, type="expected")) # Expected
number of events
 fit1 <- glm(status ~ offset(p), family="poisson", data = data)
 coef(fit1) # Should be 0
}

# calibration slope
cal_slope <- function(model, data) {
 p <- log(predict(model, newdata = data, type="expected"))
```

```
 # Calculate the linear predictor
 # lp <- model.matrix(model$formula,data)[,-1] %*% model$coefficients
 lp <- matrix(data$lp) %*% model$coefficients
 lpc <- predict(model, newdata = data, type="lp") #Centered linear
predictor
 logbase <- p - lp
 fit2 <- glm(status ~ lpc + offset(logbase), family = poisson, data =
data)
 coef(fit2)["lpc"] # Should be 1
}

# ten year risk
ten_risk <- function(model=model,lp=lp) {
 base <- basehaz(fit,centered=F)
 timediff <- (abs(base$time-365*10)) # Identify patient with 10y follow-
up
 base10 <- base$hazard[which(timediff == min(timediff))] # Extract
cumulative baseline hazard for a patient with 10y follow-up
 predsurv <- exp(-base10)**(exp(lp))
 predrisk <- 1-predsurv

 # risk <- rep(NA,nrow(ds))
 # bh  <- summary(survfit(Surv(time,status)~1,data=ds),time=3650)$surv #
10yr cumulative survival
 # for(i in 1:nrow(ds)) risk[i] <- 1-bh^exp(lp[i])

 return(predrisk)
}

# apparent performance of prediction model
apparent_performance <- function(data=data,...) {
 # fit <-
coxph(Surv(time,status)~leeftijd+geslacht+labchol+labhdl+bdsys+mht_all+ro
ken+vz_DM, data=data)
 fit <- coxph(Surv(time,status)~leeftijd+geslacht, data=data)

 # Calculate the linear predictor
 lp  <- model.matrix(fit$formula,data)[,-1] %*% fit$coefficients

 # c-index
 cindex <-
as.numeric(survConcordance(Surv(time,status)~lp,data=data)$concordance)

 # calibration intercept
 intercept <- cal_intercept(fit,data)

 # calibration slope
 slope   <- cal_slope(fit,data)

 # EO
 eo  <- sum(predict(fit,newdata=data,type="expected"))/sum(data$status)

 return(c(cindex=cindex,
      cal_inter=intercept,
      cal_slope=slope,
      eo=eo))
}

# optimism corrected performance of prediction model
```

```
oc_performance <- function(data=data,...) {
 result <- array(NA,dim=c(nrow(data),3))
 colnames(result) <- c("lp","time","status")
 result[,c("time","status")] <- c(data[,"time"],data[,"status"])

 pb <- txtProgressBar(min = 0, max = nrow(data), style = 3)
 for(i in 1:nrow(data)) {
  setTxtProgressBar(pb, i)
  training_data <- data[-i,]
  test_case <- data[i,]

  # fit <-
coxph(Surv(time,status)~leeftijd+geslacht+labchol+labhdl+bdsys+mht_all+ro
ken+vz_DM,data=training_data)
  fit <- coxph(Surv(time,status)~leeftijd+geslacht,data=training_data)

  # Calculate the linear predictor
  result[i,"lp"]  <- model.matrix(fit$formula,test_case)[,-1] %*%
fit$coefficients
 }
 result <- as.data.frame(result)

 # refit cox
 fit2 <- coxph(Surv(time,status)~lp,data=result)

 # c-index
 cindex <-
as.numeric(survConcordance(Surv(time,status)~lp,data=result)$concordance)

 # calibration intercept
 intercept <- cal_intercept(fit2,result)

 # calibration slope
 slope   <- cal_slope(fit2,result)

 # EO
 eo  <-
sum(predict(fit2,newdata=result,type="expected"))/sum(result$status)

 return(c(cindex=cindex,
      cal_inter=intercept,
      cal_slope=slope,
      eo=eo))
}

#' Simulate missing data and impute using joint modelling imputation
#'
#' @author Steven W J Nijman \email{S.W.J.Nijman@@umcutrecht.nl}
#' @author Thomas P A Debray \email{T.Debray@@umcutrecht.nl}
#'
#' @param ref_data reference data, used to estimate prediction model in
#' @param ext_data used when validation==external, to specify data on
which imputation model should be estimated.
#' @param model_data used when validation==enrich, to specify prediction
model in jackknife sample of reference data, of which part is used to
enrich external data.
#' @param scenarios list of missing value scenarios using matching
colnames.
```

```
#' @param validation specifies which method is used for validation,
defalut is "jackknife". Options are internal, external, jackknife or
enrich.
#' @param seed set seed for stochastic processes in simulation.
#'
#' @return Returns a matrix with reference linear predictor, imputed
linear predictor, expected number of events, time-to-event, status,
method used, scenario imputed and row reference.
run_simulation <- function(ref_data=ref_data,
              ext_data=ext_data,
              model_data=model_data,
              scenarios=scenarios,
              validation="jackknife",
              seed=seed,...) {

 # Create large matrix based on amount of scenarios, patients in
reference data and amount of imputation methods (3)
 sim_frame <- matrix(NA,nrow=(nrow(ref_data)*length(scenarios)*3),ncol=8)
 colnames(sim_frame) <-
c("lp_ref","lp_est","Expected","time","status","scenario","method","rowre
f")
 sim_frame[,"scenario"] <-
rep(1:length(scenarios),each=nrow(ref_data),times=3)
 sim_frame[,"method"]  <- rep(1:3,each=nrow(ref_data)*length(scenarios))
 sim_frame[,"time"]   <- rep(ref_data$time,times=3*length(scenarios))
 sim_frame[,"status"]  <- rep(ref_data$status,times=3*length(scenarios))

 # set row reference in large matrix for correct row in reference data
 sim_frame[,"rowref"]  <- rep(1:nrow(ref_data),times=3*length(scenarios))

 # added to specify which row to take when enriching external data
 if(validation=="enrich") {
  modelref <- matrix(NA,nrow=nrow(sim_frame),ncol=1)
  colnames(modelref) <- "modelref"
  for(row in 1:nrow(sim_frame)) {
   j <- as.numeric(sim_frame[row,"rowref"])
   modelref[row] <- as.numeric(rownames(model_data[which(model_data$time
== ref_data[j,"time"] &
                                model_data$status == ref_data[j,"status"]
&
                                model_data$leeftijd ==
ref_data[j,"leeftijd"] &
                                model_data$geslacht ==
ref_data[j,"geslacht"] &
                                model_data$labchol ==
ref_data[j,"labchol"] &
                                model_data$bdsys ==
ref_data[j,"bdsys"]),]))
  }
  sim_frame <- cbind(sim_frame, modelref)
 }

 # separate framingham predictors and all covariates (including
predictors and auxiliary variables)
 frh_vars <-
c("leeftijd","geslacht","labchol","labhdl","bdsys","mht_all","roken","vz_
DM")

 # determine additional patient variables when using local or external
data
```

```
 if(validation=="external") {
  aux_vars <- names(ref_data[which(names(ref_data) %in%
names(ext_data))])
 } else {
  aux_vars <- names(ref_data[-which(names(ref_data) %in%
c("time","status"))])
 }

 if (!is.na(seed)) {
  set.seed(seed)
 }

 pb <- txtProgressBar(min = 0, max = nrow(sim_frame), style = 3)

 # run simulation
 for (i in 1:nrow(sim_frame)) {
  setTxtProgressBar(pb, i)
  j <- sim_frame[i,"rowref"]
  if("modelref" %in% colnames(sim_frame)) k <- sim_frame[i,"modelref"]

  # validation selection
  if (validation == "jackknife") {
   training_data <- imp_data <- ref_data[-j,]
   test_case   <- ref_data[j,]
  } else if (validation == "internal") {
   training_data <- imp_data <- ref_data
   test_case   <- ref_data[j,]
  } else if (validation == "external") {
   training_data <- ref_data[-j,]
   imp_data   <- ext_data
   test_case   <- ref_data[j,]
  } else if (validation == "enrich") {
   training_data <- model_data[-k,]
   imp_data   <- ext_data
   test_case   <- ref_data[j,]
  } else {
   stop ("Validation method not supported")
  }

  # estimate cox
  fit <-
coxph(Surv(time,status)~leeftijd+geslacht+labchol+labhdl+bdsys+mht_all+ro
ken+vz_DM, data=training_data)

  # calculate lp given complete data
  sim_frame[i,"lp_ref"] <- model.matrix(fit$formula,test_case)[,-1] %*%
fit$coefficients

  # given scenario make specified predictors missing
  missing_var <- scenarios[sim_frame[i,"scenario"]][[1]]
  test_case[,missing_var] <- NA

  # estimate means and covariance based on pre-specified training data
  mu  <- colMeans(imp_data[,aux_vars])
  sigma <- cov(imp_data[,aux_vars])

  # given method specify imputation (mean, joint or joint with auxiliary)
  method_num <- as.numeric(sim_frame[i,"method"])
  if(method_num==1) {
   test_case[,missing_var]  <- mu[missing_var]
```

```
  } else if(method_num==2) {
   test_case[,missing_var]  <- impJoint(data=test_case[,frh_vars],
                       imp_means=mu[frh_vars],
                       imp_cov=sigma[frh_vars,frh_vars])[missing_var]
  } else if(method_num==3) {
   test_case[,missing_var]  <- impJoint(data=test_case[,aux_vars],
                       imp_means=mu,
                       imp_cov=sigma)[missing_var]
  }

  # extract expected number of events
  sim_frame[i,"Expected"]  <- predict(fit, newdata = test_case,
type="expected")

  # calculate estimated lp given imputed data
  sim_frame[i,"lp_est"]   <- model.matrix(fit$formula,test_case)[,-1] %*%
fit$coefficients
 }
 return(as.data.frame(sim_frame))
}

eval_jk <- function(results = results,
          imputation_method = 1,
          pattern = 1,...) {
 data  <- subset(results, method==imputation_method & scenario==pattern)
 mse  <- (sum((data$lp_ref-data$lp_est)**2))/nrow(data)
 cindex <- as.numeric(survConcordance(Surv(time, status)~lp_est,
data=data)$concordance)

 # calculate required estimates (Crowson 2016)
 p    <- log(data$Expected)
 data$lpc <- data$lp_est - mean(data$lp_est)
 logbase <- p - data$lp_est

 # extract quantities of interest
 slope  <- as.numeric(glm(status ~ lpc + offset(logbase), family =
poisson, data = data)$coefficients["lpc"])
 inter  <- as.numeric(glm(status ~ offset(p), family="poisson", data =
data)$coefficients["(Intercept)"])

 # calculate base OE
 OE1 <- sum(data$Expected) / sum(data$status)
 data$group <- cut(data$lpc,quantile(data$lpc,probs=seq(0,1,0.33)))
 OE <- rep(NA,length(unique(data$group)))
 for(i in 1:length(unique(data$group))) {
  OE[i] <- sum(data$Expected[which(data$group==levels(data$group)[i])]) /
sum(data$status[which(data$group==levels(data$group)[i])])
 }

 return(list(MSE      = mse,
       Cindex    = cindex,
       Cal_intercept = inter,
       Cal_slope   = slope,
       baseOE    = OE1,
       seqOE     = OE))
}

get_groupkm <- function(sim=sim,m=3,scen=scen) {
 km    <- list()
 for(i in 1:m) {
```

```
  data     <- subset(sim, method==i & scenario==scen)
  timediff   <- abs(data$time-365*10)
  tenyrpatient <- data[which(timediff == min(timediff)),]
  base.risk.10y <- tenyrpatient$Expected/exp(tenyrpatient$lp_ref)
  surv.10y   <- exp(-base.risk.10y)
  predrisk   <- 1 - (surv.10y^exp(data$lp_est))
  km[[i]]    <-
groupkm(predrisk,Surv(data$time,data$status),g=5,u=(10*365),pl=F)
 }
 km1  <- data.frame(km[[1]])
 km2  <- data.frame(km[[2]])
 km3  <- data.frame(km[[3]])
 data  <- rbind(km1,km2,km3)
 data$method <- as.factor(c(rep("MI",5),rep("JMI",5),rep("JMI.adj",5)))
 return(data)
}
```