An Introduction to the VDRA package

Thomas Kent

February 10, 2020

1 Ideas that are coming out as I'm writing the vignette.

1. Add references

2 Background

In medical research, it is common for researchers to need to gather data from multiple sources. However, due to privacy concerns (HIPAA) or the propitiatory nature of the data, one or more of the data partners may not be in a position to share the data. Various methods have been proposed in the literature, and some R packages have been written (`distcomp` and `ppmHR`), which allow analyses to be performed on distributed data in a secure setting. That is, the analyses are performed as if the data were aggregated, but in reality each data partner maintains control over their own data and only shares high level statistics in such a way that the original data cannot be deduced by the other data partners (references).

Each of these methods assumes that the data is horizontally partitioned. That is, each data partner has the same response variable and covariates for distinct cohort of observations. What makes the approach in this package novel is that we assume that the data is vertically partitioned. That is, each data partner holds a set of unique covariates for the same cohort of observations. (One way to think about this is with horizontally partitioned data, the observation matrix is partitioned by horizontal lines and in vertically partitioned data, the observation matrix is partitioned by vertical lines.)

In this package, we implement three different protocols for vertical distributed regression analysis (VDRA) in a secure setting. They are called 2-party, 2^T-party, and k^T-party. Each of these protocols can be used to perform linear, logistic, and Cox regression. Details these protocols can be found in *Distributed Regression Analysis on Vertically Partitioned Data*, Q. Her, T. Kent, Y. Samizo, A. Slackovic, S. Toh, Y. Vilk, in preparation. (references to our other papers and to JOSS paper)

2.1 Dependencies

All efforts were made so that this package would run in Base-R. The only exception is that there are select cases when using functions from the `survival` package is possible and more efficient than using our own functions. However, even if the `survival` package is not installed, full functionality of the package can be expected.

2.2 2-party VDRA

With 2-party VDRA, there are only two data holders, DP0 and DP1 (short for "data partner 0" and "data partner 1"). This protocol allows DP0 and DP1 to communicate directly with each other, but they never share raw data with each other. The final regression results are computed by DP0 and shared it with DP1. Later on, the name DP0 is reserved for the analysis center, which supplies no data, but assists in the computations. The reason that we use DP0 for the first data partner is that it also serves as the analysis center.

2.3 2^T-party VDRA

With 2^T-party VDRA, there are only two data holders, DP1 and DP2, and an analysis center, DP0. DP1 and DP2 cannot communicate directly with each other, but all communication must pass through DP0. DP0 is a trusted third party that helps facilitate communication and performs much of the computation that was performed by DP1 in the 2-party protocol. Furthermore, any information shared with DP0 from one data partner (never raw data), is multiplied by a random orthonormal matrix before being sent to the other data partner. This adds an extra layer of security at the expense of sending more data. The final regression results are computed by DP0 and shared with DP1 and DP2.

2.4 k^T-party VDRA

With k^T-party VDRA, there are at least two data holders, DP1, DP2, ... DPk, and one analysis center, DP0. With this protocol, all data partners are able to communicate with each other and with the benefit that less data is transferred. As with 2^T-party VDRA, DP0 facilitates the computations and compute the final regression results, which are then sent to all the data partners. The one possible concern with this method is that a data breach at both the analysis center and one of the data partners could expose another data partners data, even though neither the analysis center nor any data partner have enough information to reconstruct any part of any other data partners data.

3 PopMedNet and the PopMedNet Simulator

PopMedNet, short for Population Medicine Network is a "scalable and extensible open-source informatics platform designed to facilitate the implementation and operation of distributed health data networks." PopMedNet is maintained by Harvard.... As part of the project to create this software under NIH Grant...., modifications were made to PopMedNet that allowed it to act as a secure means of communication between the data partners and analysis center. While this package was designed with PopMedNet in mind as the means of communication, it should be possible to use other file transfer software perform the same task.

If a group of data partners wishes to use PopMedNet and this package for analysis of vertically distributed data, please refer to the vignette *How to use PopMedNet* for further information. However, for the individuals which are interested in testing out the package to see how it works, a PopMedNet simulator, `pmn()`, has been supplied as part of the package. This allows the individual to play the part of the all the data partners and the analysis center on a single computer in order to gain an understanding of the steps involved.

We demonstrate the usage of this package, using `pmn()` as the file transfer protocol, for 2-party, 2^T-party, and k^T-party situations. Note the use of the parameter `popmednet = FALSE` in all function calls. When `popmednet = TRUE` is used, an offset is added to make sure that when two or more data partners are running in parallel, there is a at least a 15 second window between when they signal PopMedNet that they are ready to upload files. There are technical reasons for this that are specific to PopMednet, but are not applicable when other file transfer protocols are utilized.

4 Data

The `VDRA` package comes with a data set `vdra_data` which contains simulated data from a BMI study. There are 5,740 subjects with four possible response variables and 37 covariates. The 4 response variables are:

Variable	Intended Regression
`Change_BMI`	Linear
`WtLost`	Logistic
`Time`, `Status`	Cox

If each data partner is using their own data, it is assumed that each data partner has the same number of observations and that observations on corresponding rows correspond to each other. In other words, it is assumed that the data are already aligned according to some common key, and it will be treated as such. Additionally, it is expected that the data will have already been cleaned and and only valid values are presented. Some checking of the data is performed, including looking for invalid numerical or categorical values, and if there are are any problems, an descriptive error message will be given and the program will terminate.

5 2-party Vertically Distributed Regression

For this portion of the vignette, we assume that the directory `~/vdra` is empty. If it is not empty, such as having left over files from previous runs of the program, then unintended results may happen causing the program to terminate prematurely. We must first install and load the package `vdra`. Additionally, we will run `pmn()` telling it that there will be one data partner beyond the analysis center data partner, and that the working directory is `~/vdra`.

```
> install.packages("vdra")
> library(vdra)
> if (!dir.exists("~/vdra")) dir.create("~/vdra")
> pmn(1, "~/vdra")
```

At this point, `pmn()` will have created directories `~/vdra/dp0` and `~/vdra/dp1` along with appropriate subdirectories used in the communication process. For the sake of this vignette, Data Partner 0 has the response variable(s) and covariates 5 through 23, while Data Partner 1 has the response variable and covariates 24 through 41

5.1 Linear Regression

In order to perform linear regression in a 2-party setting, open up a second instance of R, which will run simultaneously with `pmn()`, and run the following script. This will be the analysis center. In reality, we can run either data partner first.

```
> library(vdra)
> fit = AnalysisCenter.2Party(regression    = "linear",
                              data          = vdra_data[, c(1, 5:23)],
                              response      = "Change_BMI",
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

We now open up a third instance of R, which will also run simultaneously with the other two instances, and run the following script. This will be Data Partner 1.

```
> library(vdra)
> fit = DataPartner.2Party(regression    = "linear",
                           data          = vdra_data[, 24:41],
                           monitorFolder = "~/vdra/dp1",
                           popmednet     = FALSE)
> summary(fit)
```

After a few minutes, you should see the following output which is very close to the output if we had used `lm()` on the aggregated data.

```
              Party   Estimate Std. Error    t value Pr(>|t|)
(Intercept)    dp0  -2.891e+01  1.407e-01 -2.055e+02  < 2e-16 ***
Exposure       dp0  -4.966e+00  2.637e-02 -1.884e+02  < 2e-16 ***
Age            dp0   2.006e-01  1.343e-03  1.494e+02  < 2e-16 ***
NumAV          dp0   9.980e-01  1.990e-03  5.014e+02  < 2e-16 ***
NumOA          dp0   1.994e+00  7.167e-03  2.783e+02  < 2e-16 ***
```

```
NumIP          dp0   3.022e+00  1.319e-02  2.291e+02  < 2e-16 ***
NumIS          dp0   3.996e+00  1.272e-02  3.142e+02  < 2e-16 ***
NumED          dp0   5.015e+00  1.303e-02  3.849e+02  < 2e-16 ***
ComorbidScore  dp0   2.943e-01  7.097e-03  4.146e+01  < 2e-16 ***
Covar1         dp0  -1.162e-01  6.190e-02 -1.878e+00 0.060426 .
Covar2         dp0  -2.570e-02  4.440e-02 -5.788e-01 0.562736
Covar3         dp0   4.126e-05  3.663e-02  1.126e-03 0.999101
Covar4         dp0   1.956e-02  3.281e-02  5.962e-01 0.551065
Covar5         dp0   4.494e-02  3.053e-02  1.472e+00 0.141048
Covar6         dp0  -4.832e-02  2.881e-02 -1.677e+00 0.093569 .
Covar7         dp0  -1.785e-02  5.994e-02 -2.978e-01 0.765888
Covar8         dp0   4.579e-02  4.462e-02  1.026e+00 0.304826
Covar9         dp0   6.121e-03  3.282e-02  1.865e-01 0.852043
Covar10        dp0  -3.879e-03  3.063e-02 -1.266e-01 0.899234
Covar11        dp0   1.778e-02  2.879e-02  6.177e-01 0.536798
Covar12        dp1   6.485e-02  6.267e-02  1.035e+00 0.300859
Covar13        dp1   2.399e-02  4.266e-02  5.624e-01 0.573846
Covar14        dp1   1.872e-02  3.687e-02  5.077e-01 0.611686
Covar15        dp1   3.553e-03  3.324e-02  1.069e-01 0.914863
Covar16        dp1   1.554e-02  3.075e-02  5.054e-01 0.613284
Covar17        dp1  -3.393e-02  2.862e-02 -1.186e+00 0.235795
Covar18        dp1  -7.879e-02  6.182e-02 -1.275e+00 0.202514
Covar19        dp1  -5.725e-02  4.466e-02 -1.282e+00 0.199983
NumGeneric     dp1  -8.675e-04  2.244e-03 -3.866e-01 0.699095
NumClass       dp1  -3.679e-03  4.363e-03 -8.432e-01 0.399147
NumRx          dp1   5.506e-03  6.476e-03  8.501e-01 0.395305
Pre_day        dp1   2.001e-01  2.537e-04  7.885e+02  < 2e-16 ***
BMI_pre        dp1  -8.985e-04  1.458e-03 -6.165e-01 0.537614
Pre_day_E      dp1   2.852e-04  3.196e-04  8.926e-01 0.372123
count_pre      dp1  -8.163e-03  5.260e-03 -1.552e+00 0.120766
Year:2011      dp1  -5.353e-02  4.507e-02 -1.188e+00 0.234975
Year:2012      dp1  -5.253e-02  4.468e-02 -1.176e+00 0.239675
Year:2013      dp1  -1.289e-02  4.481e-02 -2.876e-01 0.773649
Year:2014      dp1  -2.853e-02  4.387e-02 -6.503e-01 0.515554
Year:2015      dp1   7.481e-03  4.829e-02  1.549e-01 0.876906
Race:Race 1    dp1   1.010e+00  5.273e-02  1.916e+01  < 2e-16 ***
Race:Race 2    dp1   1.972e+00  4.951e-02  3.983e+01  < 2e-16 ***
Race:Race 3    dp1   3.060e+00  4.904e-02  6.239e+01  < 2e-16 ***
Race:Race 4    dp1   4.064e+00  4.939e-02  8.229e+01  < 2e-16 ***
Race:Race 5    dp1  -1.023e+00  4.995e-02 -2.048e+01  < 2e-16 ***
Sex:M          dp1  -2.041e+00  3.767e-02 -5.417e+01  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error:  0.9949 on 5694 degrees of freedom
Multiple R-squared:  0.9959 , Adjusted R-squared:  0.9959
F-statistic: 30625 on 45 and 5694 DF, p-value: < 2.22e-16
```

For comparison, we also provide the output for the following code:

```
> fit = lm(Change_BMI ~ ., vdra_data[, -(2:4)])
> summary(fit)
```

Output:

```
               Estimate Std. Error  t value Pr(>|t|)    
(Intercept)   -2.891e+01  1.407e-01 -205.471   <2e-16 ***
Exposure      -4.966e+00  2.637e-02 -188.370   <2e-16 ***
Age            2.006e-01  1.343e-03  149.359   <2e-16 ***
NumAV          9.980e-01  1.990e-03  501.416   <2e-16 ***
NumOA          1.994e+00  7.167e-03  278.274   <2e-16 ***
NumIP          3.022e+00  1.319e-02  229.101   <2e-16 ***
NumIS          3.996e+00  1.272e-02  314.176   <2e-16 ***
NumED          5.015e+00  1.303e-02  384.917   <2e-16 ***
ComorbidScore  2.943e-01  7.097e-03   41.463   <2e-16 ***
Covar1        -1.162e-01  6.190e-02   -1.878   0.0604 .  
Covar2        -2.570e-02  4.440e-02   -0.579   0.5627    
Covar3         4.126e-05  3.663e-02    0.001   0.9991    
Covar4         1.956e-02  3.281e-02    0.596   0.5511    
Covar5         4.494e-02  3.053e-02    1.472   0.1410    
Covar6        -4.832e-02  2.881e-02   -1.677   0.0936 .  
Covar7        -1.785e-02  5.994e-02   -0.298   0.7659    
Covar8         4.579e-02  4.462e-02    1.026   0.3048    
Covar9         6.121e-03  3.282e-02    0.187   0.8520    
Covar10       -3.879e-03  3.063e-02   -0.127   0.8992    
Covar11        1.778e-02  2.879e-02    0.618   0.5368    
Covar12        6.485e-02  6.267e-02    1.035   0.3009    
Covar13        2.399e-02  4.266e-02    0.562   0.5738    
Covar14        1.872e-02  3.687e-02    0.508   0.6117    
Covar15        3.553e-03  3.324e-02    0.107   0.9149    
Covar16        1.554e-02  3.075e-02    0.505   0.6133    
Covar17       -3.393e-02  2.862e-02   -1.186   0.2358    
Covar18       -7.879e-02  6.182e-02   -1.275   0.2025    
Covar19       -5.725e-02  4.466e-02   -1.282   0.2000    
NumGeneric    -8.675e-04  2.244e-03   -0.387   0.6991    
NumClass      -3.679e-03  4.363e-03   -0.843   0.3991    
NumRx          5.506e-03  6.476e-03    0.850   0.3953    
Pre_day        2.001e-01  2.538e-04  788.467   <2e-16 ***
BMI_pre       -8.986e-04  1.458e-03   -0.616   0.5376    
Pre_day_E      2.852e-04  3.195e-04    0.893   0.3721    
count_pre     -8.163e-03  5.260e-03   -1.552   0.1208    
Year2011      -5.353e-02  4.507e-02   -1.188   0.2350    
Year2012      -5.253e-02  4.468e-02   -1.176   0.2397    
Year2013      -1.289e-02  4.481e-02   -0.288   0.7736    
Year2014      -2.853e-02  4.387e-02   -0.650   0.5156    
Year2015       7.481e-03  4.829e-02    0.155   0.8769    
RaceRace 1     1.010e+00  5.273e-02   19.163   <2e-16 ***
RaceRace 2     1.972e+00  4.951e-02   39.830   <2e-16 ***
RaceRace 3     3.060e+00  4.904e-02   62.389   <2e-16 ***
RaceRace 4     4.064e+00  4.939e-02   82.289   <2e-16 ***
RaceRace 5    -1.023e+00  4.995e-02  -20.484   <2e-16 ***
SexM          -2.041e+00  3.767e-02  -54.171   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9949 on 5694 degrees of freedom
Multiple R-squared:  0.9959,
F-statistic: 3.062e+04 on 45 and 5694 DF,  p-value: < 2.2e-16
```

5.2 Logistic Regression

In order to perform logistic regression in a 2-party setting, open up a second instance of R, which will run simultaneously with `pmn()`, and run the following script. We first run DP0 which is also the analysis center. In reality, we can run either data partner first.

```
> library(vdra)
> fit = AnalysisCenter.2Party(regression    = "logistic",
                              data          = vdra_data[, c(2, 5:23)],
                              response      = "WtLost",
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

We now open up a third instance of R, which will also run simultaneously with the other two instances, and run the following script. This will be Data Partner 1.

```
> library(vdra)
> fit = DataPartner.2Party(regression    = "logistic",
                           data          = vdra_data[, 24:41],
                           monitorFolder = "~/vdra/dp1",
                           popmednet     = FALSE)
> summary(fit)
```

The output for the logistic regression is similar to

```
> fit = glm(WtLost ~ ., vdra_data[, -c(1, 3, 4)], family = binomial)
> summary(fit)
```

5.3 Cox Regression

In order to perform Cox regression in a 2-party setting, open up a second instance of R, which will run simultaneously with `pmn()`, and run the following script. We first run DP0 which is also the analysis center. In reality, we can run either data partner first.

```
> library(vdra)
> fit = AnalysisCenter.2Party(regression    = "Cox",
                              data          = vdra_data[, c(3:4, 5:23)],
                              response      = c("Time", "Status"),
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

We now open up a third instance of R, which will also run simultaneously with the other two instances, and run the following script. This will be Data Partner 1.

```
> library(vdra)
> fit = DataPartner.2Party(regression = "Cox",
                           data          = vdra_data[, 24:41],
                           monitorFolder = "~/vdra/dp1",
                           popmednet     = FALSE)
> summary(fit)
```

The output for Cox regression is similar to

```
> library(survival)
> fit = coxph(Surv(Time, Status) ~ ., data = vdra_data[, -(1:2)])
> summary(fit)
```

6 2^T-party Vertically Distributed Regression

This section is very similar to the section for 2-party Vertically Distributed Regression. As above, we assume that the directory `~/vdra` is empty. If it is not empty, such as having left over files from previous runs of the program, then unintended results may happen causing the program to terminate prematurely. We must first install and load the package `vdra`. Additionally, we will run `pmn()` telling it that there will be two data partners beyond the analysis center, and that the working directory is `~/vdra`.

```
> install.packages("vdra")
> library(vdra)
> if (!dir.exists("~/vdra")) dir.create("~/vdra")
> pmn(2, "~/vdra")
```

At this point, `pmn()` will have created directories `~/vdra/dp0`, `~/vdra/dp1`, `~/vdra/dp2` along with appropriate subdirectories used in the communication process. For the sake of this vignette, Data Partner 1 has the response variable(s) and covariates 5 through 23, while Data Partner 2 has covariates 24 through 41.

6.1 Linear Regression

In order to perform linear regression in a 2^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run either data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.3Party(regression    = "linear",
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner1.3Party(regression    = "linear",
                            data          = vdra_data[, c(1, 5:23)],
                            response      = "Change_BMI",
                            monitorFolder = "~/vdra/dp1",
                            popmednet     = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2.

```
> library(vdra)
> fit = DataPartner2.3Party(regression    = "linear",
                               data          = vdra_data[, 24:41],
                               monitorFolder = "~/vdra/dp2",
                               popmednet     = FALSE)
> summary(fit)
```

After a few minutes, you should see the same output as in the 2-party scenario

6.2 Logistic Regression

In order to perform logistic regression in a 2^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run either data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.3Party(regression    = "logistic",
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner1.3Party(regression    = "logistic",
                            data          = vdra_data[, c(2, 5:23)],
                            response      = "WtLost",
                            monitorFolder = "~/vdra/dp1",
                            popmednet     = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2.

```
> library(vdra)
> fit = DataPartner2.3Party(regression    = "logistic",
                               data          = vdra_data[, 24:41],
                               monitorFolder = "~/vdra/dp2",
                               popmednet     = FALSE)
> summary(fit)
```

After a few minutes, you should see the same output as in the 2-party scenario

6.3 Cox Regression

In order to perform Cox regression in a 2^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run either data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.2Party(regression    = "Cox",
                              monitorFolder = "~/vdra/dp0",
                              popmednet     = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner1.3Party(regression    = "Cox",
                            data          = vdra_data[, c(3:4, 5:23)],
                            response      = c("Time", "Status"),
                            monitorFolder = "~/vdra/dp1",
                            popmednet     = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2.

```
> library(vdra)
> fit = AnalysisCenter2.3Party(regression    = "Cox",
                               data          = vdra_data[, 24:41],
                               monitorFolder = "~/vdra/dp2",
                               popmednet     = FALSE)
> summary(fit)
```

After a few minutes, you should see the same output as in the 2-party scenario

7 k^T-party Vertically Distributed Regression

This section is very similar to the section for 2^T-party Vertically Distributed Regression. As above, we assume that the directory `~/vdra` is empty. If it is not empty, such as having left over files from previous runs of the program, then unintended results may happen causing the program to terminate prematurely. We must first install and load the package `vdra`. Additionally, we will run `pmn()` telling it that there will be two data partners beyond the analysis center data partner, and that the working directory is `~/vdra`. Note, for k^T-party, we are not limited to two data partners, but we can have any number. In fact, we have successfully run this program for $k = 10$ data partners. However, for the sake of this vignette, we will restrict ourselves to just two and note that would only have to change the value of `numDataPartners` for $k > 2$. For example, if we had $k = 10$ data partners, we would need to set `numDataPartners = 10` in each of the following code blocks.

```
> install.packages("vdra")
> library(vdra)
> if (!dir.exists("~/vdra")) dir.create("~/vdra")
> pmn(2, "~/vdra")
```

At this point, `pmn()` will have created directories `~/vdra/dp0`, `~/vdra/dp1`, `~/vdra/dp2` along with appropriate subdirectories used in the communication process. For the sake of this vignette, Data Partner 1 has the response variable(s) and covariates 5 through 23, while Data Partner 1 has the response variable and covariates 24 through 41

7.1 Linear Regression

In order to perform linear regression in a k^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run any data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.KParty(regression      = "linear",
                              numDataPartners = 2,
                              monitorFolder   = "~/vdra/dp0",
                              popmednet       = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "linear",
                           data            = vdra_data[, c(1, 5:23)],
                           response        = "Change_BMI",
                           numDataPartners = 2,
                           dataPartnerID   = 1,
                           monitorFolder   = "~/vdra/dp1",
                           popmednet       = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2. If we had more than two data partners, then we execute this same code in a new instance of R for each data partner, incrementing the value of `dataPartnerID` appropriately.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "linear",
                           data            = vdra_data[, 24:41],
                           numDataPartners = 2,
                           dataPartnerID   = 2,
                           monitorFolder   = "~/vdra/dp2",
                           popmednet       = FALSE)
> summary(fit)
```

7.2 Logistic Regression

In order to perform linear regression in a k^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run any data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.KParty(regression      = "logistic",
                              numDataPartners = 2,
                              monitorFolder   = "~/vdra/dp0",
                              popmednet       = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "logistic",
                           data            = vdra_data[, c(2, 5:23)],
                           response        = "WtLost",
                           numDataPartners = 2,
                           dataPartnerID   = 1,
                           monitorFolder   = "~/vdra/dp1",
                           popmednet       = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2. If we had more than two data partners, then we execute this same code in a new instance of R for each data partner, incrementing the value of `dataPartnerID` appropriately.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "logistic",
                           data            = vdra_data[, 24:41],
                           numDataPartners = 2,
                           dataPartnerID   = 2,
                           monitorFolder   = "~/vdra/dp2",
                           popmednet       = FALSE)
> summary(fit)
```

7.3 Cox Regression

In order to perform linear regression in a k^T-party setting, open up a second instance of R, which will run simultaneously with `pmn()`. We first run DP0 which is also the analysis center. In reality, we can run any data partner or the analysis center first.

```
> library(vdra)
> fit = AnalysisCenter.KParty(regression      = "Cox",
                             numDataPartners = 2,
                             monitorFolder   = "~/vdra/dp0",
                             popmednet       = FALSE)
> summary(fit)
```

Open up a third instance of R, which will run simultaneously with `pmn()` and run the following script. This will be Data Partner 1, which has the response variable.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "Cox",
                          data            = vdra_data[, c(3:4, 5:23)],
                          response        = c("Time", "Status"),
                          numDataPartners = 2,
                          dataPartnerID   = 1,
                          monitorFolder   = "~/vdra/dp1",
                          popmednet       = FALSE)
> summary(fit)
```

We now open up a fourth instance of R, which will also run simultaneously with the other three instances, and execute the following code. This will be Data Partner 2. If we had more than two data partners, then we execute this same code in a new instance of R for each data partner, incrementing the value of `dataPartnerID` appropriately.

```
> library(vdra)
> fit = DataPartner.KParty(regression      = "Cox",
                          data            = vdra_data[, 24:41],
                          numDataPartners = 2,
                          dataPartnerID   = 2,
                          monitorFolder   = "~/vdra/dp2",
                          popmednet       = FALSE)
> summary(fit)
```

8 Utilities

In order to facilitate the analysis of the data, several utilities have been provided which allow the user to asses the goodness of fit of the model. We present these for each of the three types of regression that we perform.

8.1 Linear Regression

For linear regression, we allow the creation of sub-models. If the the result of a distributed linear regression is stored in `vdra_fit_linear_A` (which comes with the package), then we can check different sub-models as shown in the following script. All data partners can use this function.

```
> library(vdra)
> fit1 = differentModel(Age ~ ., vdra_fit_linear_A)
> summary(fit1)
> fit2 = differentModel(Change_BMI ~ Exposure + Age, vdra_fit_linear_A)
> summary(fit2)
> fit3 = differentModel(Change_BMI ~ Exposure, vdra_fit_linear_A)
> summary(fit3)
> fit4 = differentModel(Change_BMI ~ Exposure + Age + `Sex:M` + `Race:Race 1`, vdra_fit_linear_A)
> summary(fit4)
```

8.2 Logistic Regression

We can perform both Hosmer-Lemeshow goodness of fit test for logistic regression and plot the ROC according to the following script. The results of a distributed logistic regression is stored in `vdra_fit_logistic_A` (which comes with the package). Only the data partner which holds the response can run these functions.

```
> HoslemTest(vdra_fit_logistic_A)
> HoslemTest(vdra_fit_logistic_A, 20)
> RocTest(vdra_fit_logistic_A)
> RocTest(vdra_fit_logistic_A, 20)
```

8.3 Cox Regression

We can compute the survival curve for a distributed Cox regression. This function is works similar to `survfit()` in the `survival` package. The results can be both printed and plotted. Only the data partner which holds the response can run this function.

```
> sf = survfitDistributed(vdra_fit_cox_A)
> print(sf)
> plot(sf)
```