Demo-3

Variograms

Jeffrey Yarus

08 juillet, 2025

Contents

1 Demo Summary

In this demo, the objective is to become familiar with how to construct both omnidirectional and directional variograms.

Here, the code begins as usual by reading in the data set, creating the assignment variables, identifying outliers, and removing the outliers. We are now ready to create variograms. **Please be sure to read through the text I have included before each code chunk. .**

First, run the script chunk by chunk. After the initial data frames are calculated, I create a data base called db as we did in the previous demo. This data base looks a lot like the data frame df and, as you recall, it contains the outliers. You will see that I correct for the outliers and create a data base without the outliers called **db__noout.db**. You will need to walk through the chunks one-at-a-time and change **db** in each chunk to **db__noout.db** in order to see the impact without the outlier. Take your time and feel free to swap the data bases back and forth to see the impact with the outlier and without.

2 Terminology, packages, and key functions from this Demo

1. **Variogram Map** : A Variogram map is a graph that that displays the change in variance in all directions from the origin located at its center. It is actually a polar plot and requires a “lot” of data

to produce, so data sets that are sparse will not be very useful. Grid data, like satellite images, gravity, magnetics, geophysical surveys etc. produce excellent variogram maps.

3 Loading Packages

Run the code chunk below…

```
setwd('/mnt/vstor/CSE_MSE_RXF131/cradle-members/sdle/jmy41/GIT/jmy-research/topics/RPS_N58')

getwd()
```

As in the previous demos, we:

- Read in our data
- Create the tibble
- Ensure our categorical variables are designated as factors
- Create a backup data frame
- Remove the unnecessary variables from the tibble
- Create our assigned symbolic names
- Identify the outliers. all of this is being done in one code chunk.

Run the chunk below…

```
file.name <- "../../data/WT-2D-all-outlier.csv"
df <- read_csv(file.name)
```

```
## Rows: 262 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): F_Top
## dbl (10): N_Well, C_X_ft, C_Y_ft, L_3_FACIES, P_Delta_t, P_KH_md, P_PHI, P_P...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df %<>%
  mutate(across(matches("N_|L_"), factor))
head(df, n = 20)
```

```
## # A tibble: 20 x 11
##    F_Top       N_Well C_X_ft C_Y_ft L_3_FACIES P_Delta_t P_KH_md P_PHI P_PHI_pct
##    <chr>       <fct>   <dbl>  <dbl> <fct>          <dbl>   <dbl> <dbl>     <dbl>
##  1 S_Siltstone 5001    11564   5691 2               60.8    3.51 0.094       9.4
##  2 S_Siltstone 5002    10679  13706 2               59.1    5.81 0.093       9.3
##  3 S_Siltstone 5003     6311  36307 1               55.9    0.77 0.059       5.9
##  4 S_Siltstone 5004     2754  11437 1               52.5    0.55 0.044       4.4
##  5 S_Siltstone 5005     9386  24799 3               63.3    6.29 0.099       9.9
##  6 S_Siltstone 5006    10761  24755 3               63.1    2.9  0.099       9.9
##  7 S_Siltstone 5007    10769  23463 3               65.9   12.3  0.113      11.3
##  8 S_Siltstone 5008     9145  23413 1               57.9    0.79 0.074       7.4
##  9 S_Siltstone 5009     8201  23557 3               64.4    8.19 0.104      10.4
## 10 S_Siltstone 5010     6727  23472 1               55.8    1.45 0.071       7.1
## 11 S_Siltstone 5011     3834  23446 1               53.9    1.6  0.062       6.2
## 12 S_Siltstone 5012     2780  23494 1               54.8    1.57 0.063       6.3
## 13 S_Siltstone 5013     3994  22098 1               53.5    1.51 0.06        6
## 14 S_Siltstone 5014     5482  22214 1               53.9    1.79 0.065       6.5
## 15 S_Siltstone 5015     6692  22138 2               60.8    8.21 0.091       9.1
```

```
## 16 S_Siltstone 5016     9324  22199 3                63.5    4.48 0.1        10  
## 17 S_Siltstone 5017    10817  22305 2                61.7    2.73 0.093       9.3
## 18 S_Siltstone 5018    10761  20774 1                57.4    2.68 0.073       7.3
## 19 S_Siltstone 5019     9350  20761 3                62.5    3.42 0.096       9.6
## 20 S_Siltstone 5020     8237  20748 1                57.2    3.82 0.079       7.9
## # i 2 more variables: P_Thickness <dbl>, P_Top_ft <dbl>
```

```r
# Creating a duplicate tibble for later use, but eliminate unnecessary variables

df2 <- df %>%
  dplyr::select(L_3_FACIES:P_Top_ft)

# Creating Symbolic Inputs

xlon <- "C_X_ft"
ylat <- "C_Y_ft"
out_analysis <- "Raw (outlier analysis performed on raw data)"
property <-
  "P_PHI_pct" # The property you are selecting for analysis

## Data Analytics - Managing outliers

df <-
  df %>%
  mutate(
    iqr_val = IQR(!!sym(property)),
    # calculates the IQR value
    iqr_val_adj = ((iqr_val) * 1.5),
    third_q = quantile(!!sym(property), prob = 0.75, na.rm = TRUE),
    first_q = quantile(!!sym(property), prob = .25, na.rm = TRUE),
    # Creating a column of True and False, True = an outlier, False = not an
    #outlier
    outlier =
      (!!sym(property)) > (third_q + iqr_val_adj) |
      (!!sym(property) < (first_q - iqr_val_adj))
  )
df
```

```
## # A tibble: 262 x 16
##    F_Top       N_Well C_X_ft C_Y_ft L_3_FACIES P_Delta_t P_KH_md P_PHI P_PHI_pct
##    <chr>       <fct>   <dbl>  <dbl> <fct>          <dbl>   <dbl> <dbl>     <dbl>
##  1 S_Siltstone 5001    11564   5691 2               60.8    3.51 0.094       9.4
##  2 S_Siltstone 5002    10679  13706 2               59.1    5.81 0.093       9.3
##  3 S_Siltstone 5003     6311  36307 1               55.9    0.77 0.059       5.9
##  4 S_Siltstone 5004     2754  11437 1               52.5    0.55 0.044       4.4
##  5 S_Siltstone 5005     9386  24799 3               63.3    6.29 0.099       9.9
##  6 S_Siltstone 5006    10761  24755 3               63.1    2.9  0.099       9.9
##  7 S_Siltstone 5007    10769  23463 3               65.9   12.3  0.113      11.3
##  8 S_Siltstone 5008     9145  23413 1               57.9    0.79 0.074       7.4
##  9 S_Siltstone 5009     8201  23557 3               64.4    8.19 0.104      10.4
## 10 S_Siltstone 5010     6727  23472 1               55.8    1.45 0.071       7.1
## # i 252 more rows
## # i 7 more variables: P_Thickness <dbl>, P_Top_ft <dbl>, iqr_val <dbl>,
## #   iqr_val_adj <dbl>, third_q <dbl>, first_q <dbl>, outlier <lgl>
```

4 Create a database from the dataframe and select properties for analysis

Run the chunk below...

```
# Remove the first variable
df1 = df %>% dplyr::select(-F_Top)

# creating a database with the outlier from the Data Frame
db = Db_fromTL(df1)

# Define the different variables
err = db$setLocators(c(xlon, ylat), ELoc_X())
err = db$setLocator(property, ELoc_Z())
db
```

```
##
## Data Base Characteristics
## =========================
##
## Data Base Summary
## -----------------
## File is organized as a set of isolated points
## Space dimension              = 2
## Number of Columns            = 15
## Total number of samples      = 262
##
## Variables
## ---------
## Column = 0 - Name = N_Well - Locator = NA
## Column = 1 - Name = C_X_ft - Locator = x1
## Column = 2 - Name = C_Y_ft - Locator = x2
## Column = 3 - Name = L_3_FACIES - Locator = NA
## Column = 4 - Name = P_Delta_t - Locator = NA
## Column = 5 - Name = P_KH_md - Locator = NA
## Column = 6 - Name = P_PHI - Locator = NA
## Column = 7 - Name = P_PHI_pct - Locator = z1
## Column = 8 - Name = P_Thickness - Locator = NA
## Column = 9 - Name = P_Top_ft - Locator = NA
## Column = 10 - Name = iqr_val - Locator = NA
## Column = 11 - Name = iqr_val_adj - Locator = NA
## Column = 12 - Name = third_q - Locator = NA
## Column = 13 - Name = first_q - Locator = NA
## Column = 14 - Name = outlier - Locator = NA
```

```
# Creating databases with no outliers
# Remove the first variable
df1 = df %>% dplyr::select(-F_Top) %>% filter(outlier == FALSE)

# creating a database with the outlier from the Data Frame
db_noout.db = Db_fromTL(df1)

# Define the different variables
err = db_noout.db$setLocators(c(xlon, ylat), ELoc_X())
err = db_noout.db$setLocator(property, ELoc_Z())
```

```
db_noout.db
```

```
##
## Data Base Characteristics
## =========================
##
## Data Base Summary
## -----------------
## File is organized as a set of isolated points
## Space dimension              = 2
## Number of Columns            = 15
## Total number of samples      = 261
##
## Variables
## ---------
## Column = 0 - Name = N_Well - Locator = NA
## Column = 1 - Name = C_X_ft - Locator = x1
## Column = 2 - Name = C_Y_ft - Locator = x2
## Column = 3 - Name = L_3_FACIES - Locator = NA
## Column = 4 - Name = P_Delta_t - Locator = NA
## Column = 5 - Name = P_KH_md - Locator = NA
## Column = 6 - Name = P_PHI - Locator = NA
## Column = 7 - Name = P_PHI_pct - Locator = z1
## Column = 8 - Name = P_Thickness - Locator = NA
## Column = 9 - Name = P_Top_ft - Locator = NA
## Column = 10 - Name = iqr_val - Locator = NA
## Column = 11 - Name = iqr_val_adj - Locator = NA
## Column = 12 - Name = third_q - Locator = NA
## Column = 13 - Name = first_q - Locator = NA
## Column = 14 - Name = outlier - Locator = NA
```

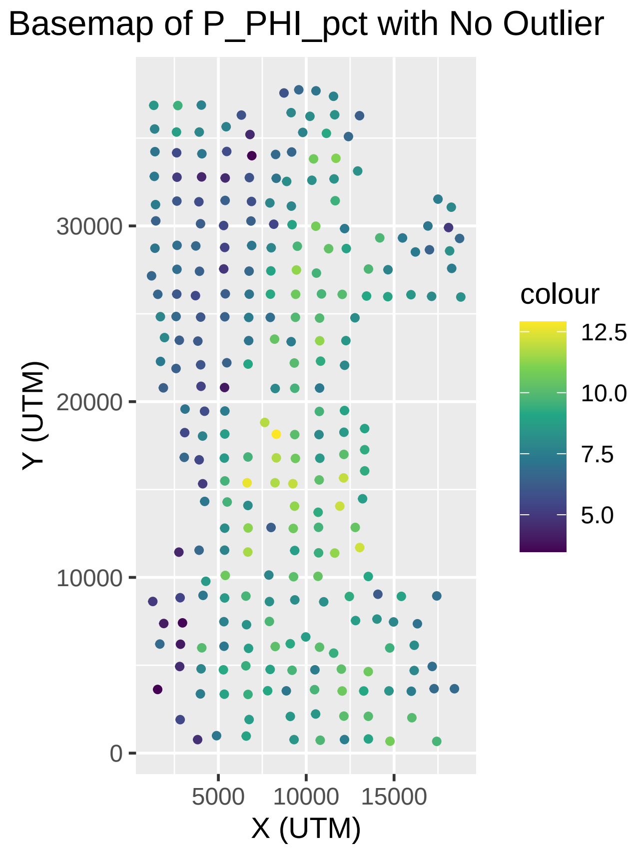
5 Create the Basemap without the outlier

In this code chunk the basemap without the outlier is reproduced.

(note, you will need to replace the database db__noout.db if you want to see the impact of the outlier)

Run the chunk below…

```
p = plot.init(asp=1)
p = p + plot.symbol(db_noout.db, nameColor = property, pch = 19, cex = 1, flagLegend = TRUE)
p = p + plot.decoration(title=paste("Basemap of", property, "with No Outlier"), xlab = "X (UTM)", ylab =
plot.end(p)
```



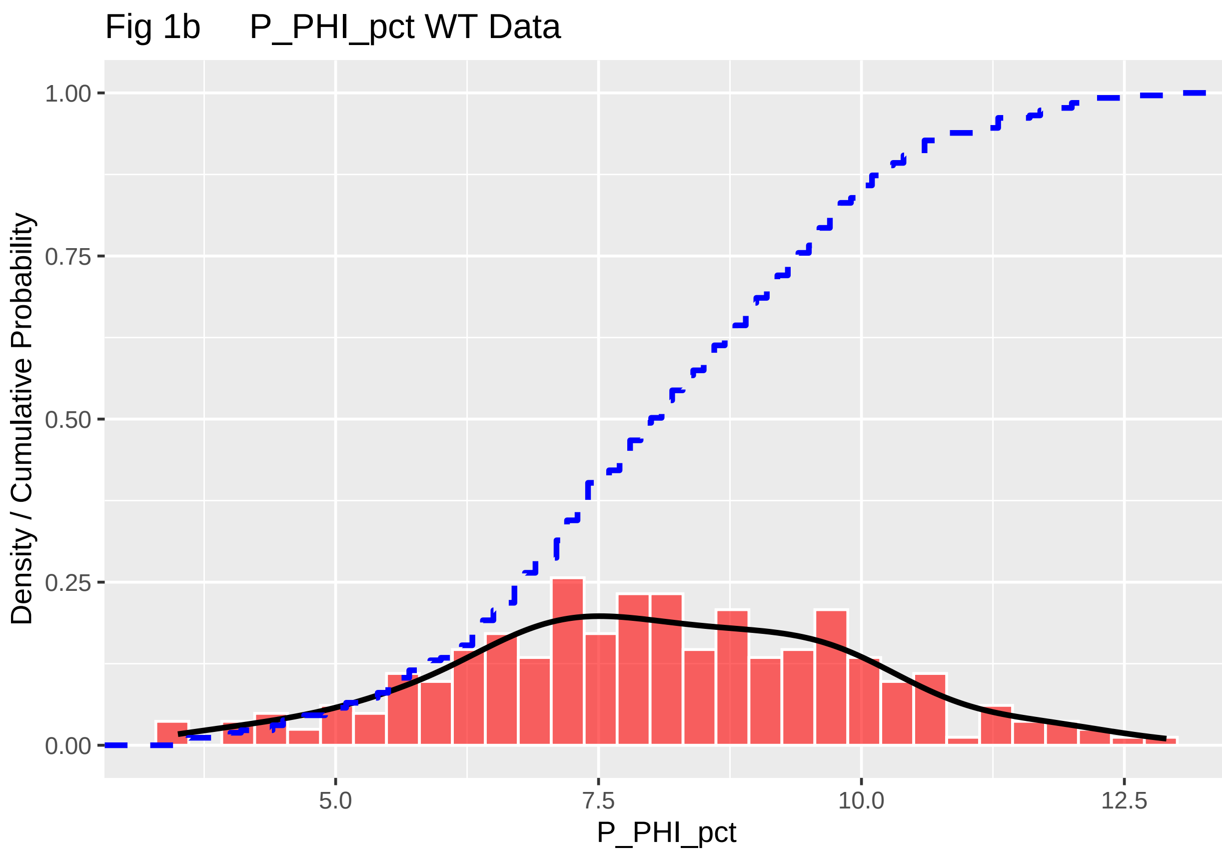
6 Outliers

The code chunck below displays the histogram from the selected variable. Note the outlier on the far right of the histogram

Run the chunk below...

```
hist1 <- df %>%
  filter(outlier == FALSE) %>%
  ggplot(aes(x = !!sym(property))) +
  geom_histogram(           #   Plots the histogram
    aes(y = ..density..),
    fill = "red",
    color = "white",
    bins = 31,              #   Smooths out the histogram and ensures center bin
    alpha = 0.6             #   Provides transparency to histogram see CDF better
  ) +
  geom_density(color = "black", size = 1) +      #   Plots the Density Curve
  stat_ecdf(
    aes(y = ..y..),
    color = "blue",
    size = 1,
    linetype = "dashed"
  ) +                       #   Plots the CPDF overlay
  ggtitle(sprintf("Fig 1b     %s WT Data", property)) +
  xlab(property) +
```

```
  ylab("Density / Cumulative Probability")

hist1
```



Note the "Density" curve in black is a "Probability Distribution Function". It shows a smoothed, continuous estimate of the data distribution — helpful for spotting skewness, multi-modality, or how normal (Gaussian) the data are. The blue curve is the "Cumulative Probability distribution function" or CPDF.

7 Construction of the Variograms

The code chunks below calculates and plots the variograms. Initially, they do not correct for outliers. Replace the database (db) with the database that removes the outliers (db_noout.db). Observe the impact of the outlier.

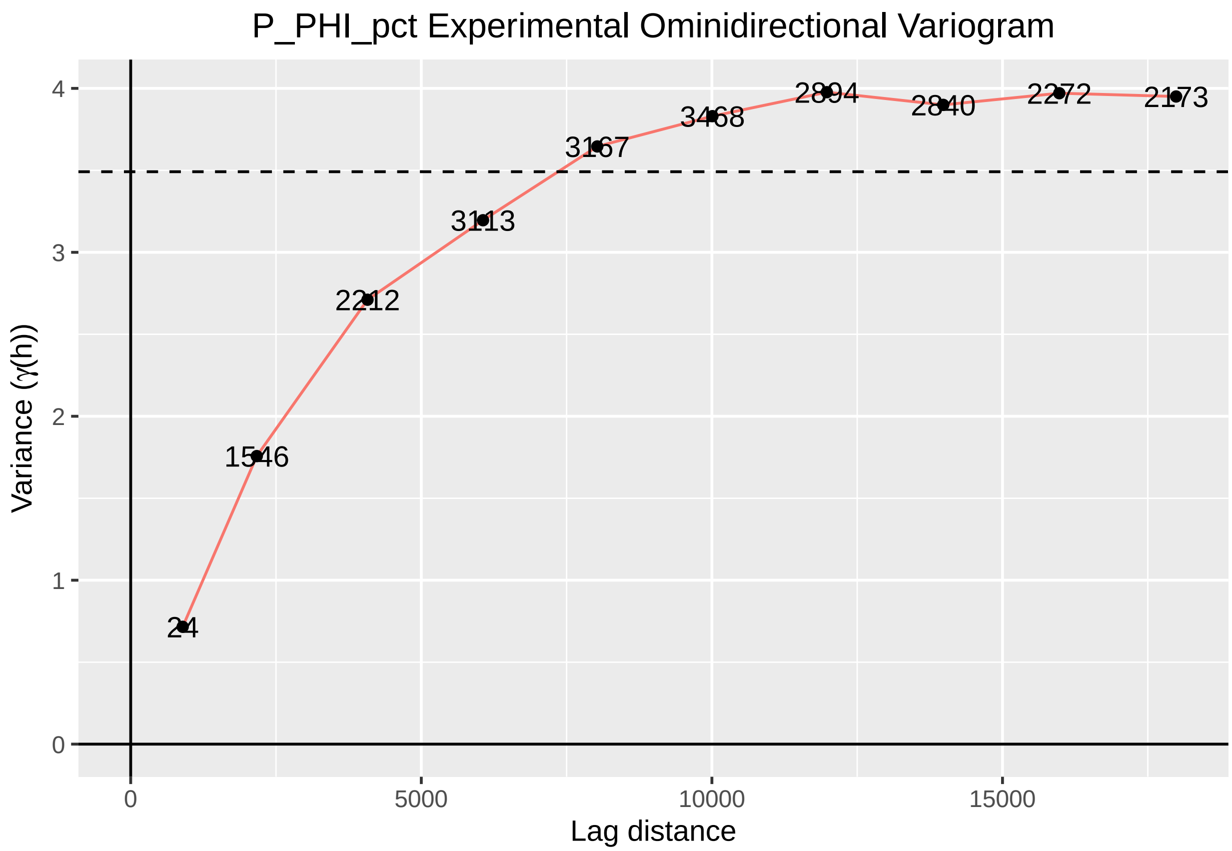
7.1 The Omnidirectional Experimental Semivariogram

Look what happens in the variogram when you do not remove the outlier! To see the impact, be sure the data base is, "db". To see the impact without the outlier, change the data base name to, "db_noout".

Run the code chunk below...

```
varioparam <- VarioParam_createOmniDirection(nlag = 10, dlag=2000)
vario_omni <- Vario_computeFromDb(varioparam, db_noout.db)

p = plot.init()
p = p + plot.vario(vario_omni, drawPlabel = TRUE)
p = p + plot.decoration(title = paste(property, "Experimental Ominidirectional Variogram"),
```

```r
                              xlab = "Lag distance",
                              ylab = expression(paste("Variance (", gamma, "(h))", sep="")))
plot.end(p)
```



8 Directional Variograms

8.1 Directional Experimental Semi- Variograms

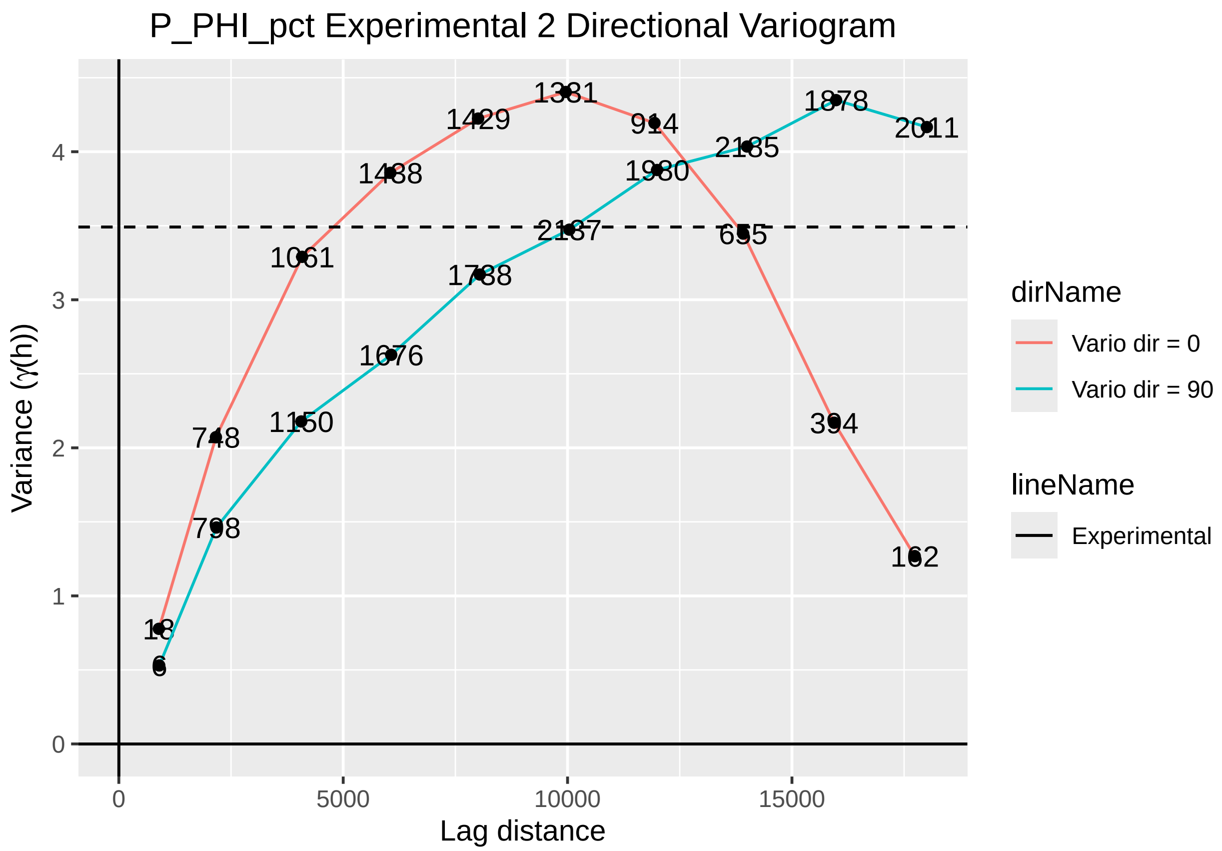
The same process is followed for directional variograms. Note, however, that the number of arguments increases and we include the multiple directions we want to assess. When entering the directions, begin with the most northerly direction. **If you enter only 2 directions, like c(90, 0), then gstlearn assumes you explicitly know the maximum and minimum directions of continuity.** If you enter 3 or more directions, like, C(90, 45, 0) or c(90, 60, 30, 0), gstlearn will automatically calculate the directions of continuity if you are using the function "Vario_computeFromDb" along with "Model_fit". This is very handy and makes variogram modeling much easier.

In the code chunk below, we calculate the directional variograms for 2 directions, 3 directions, and 4 directions.
.

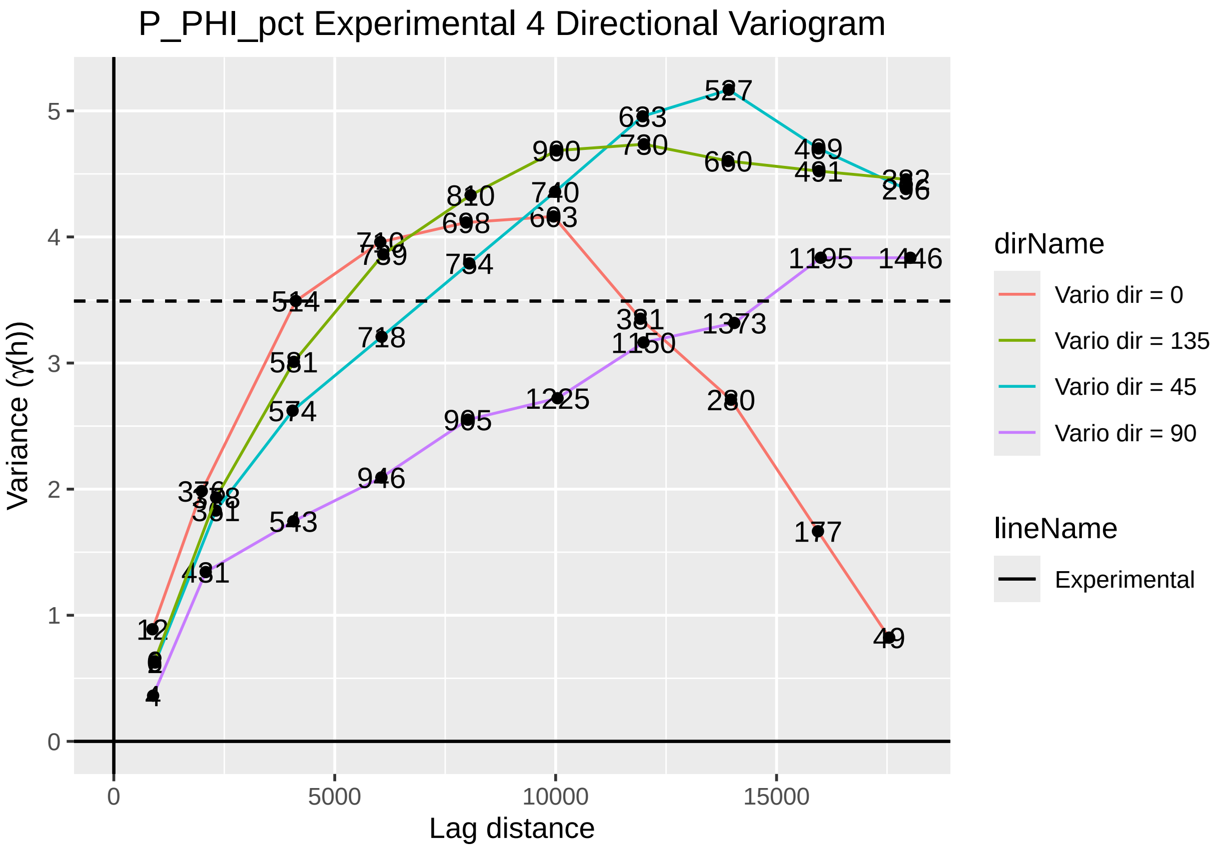
Run the chunk below…

```r
#This time to calculate multiple specific directions , such as 0, 90 degrees
varioparam <- VarioParam_createMultiple(ndir = 2, nlag = 10, dlag=2000)
data.2dir.vario <- Vario_computeFromDb(varioparam, db_noout.db)

p = plot.init()
p = p + plot.vario(data.2dir.vario, drawPlabel = TRUE, flagLegend=TRUE)
```

```r
p = p + plot.decoration(title = paste(property, "Experimental 2 Directional Variogram"),
                        xlab = "Lag distance",
                        ylab = expression(paste("Variance (", gamma, "(h))", sep = "")))
plot.end(p)
```




```r
#This time to calculate multiple specific directions , such as 0, 45, 90 and 135 degrees
varioparam <- VarioParam_createMultiple(ndir = 4, nlag = 10, dlag=2000)
data.4dir.vario <- Vario_computeFromDb(varioparam, db_noout.db)

p = plot.init()
p = p + plot.vario(data.4dir.vario, drawPlabel = TRUE, flagLegend=TRUE)
p = p + plot.decoration(title = paste(property, "Experimental 4 Directional Variogram"),
                        xlab = "Lag distance",
                        ylab = expression(paste("Variance (", gamma, "(h))", sep = "")))
plot.end(p)
```



Recall that each point represents the average squared difference value for each lag. The small numbers printed above each point is the number of pair-points that went into that lag calculation.

End of Demo#3a-2025-SM