APCC 2016 Statistical Downscaling Training Program
Basics of Weather Generator

August 2016

Dr. Moosup Kim

APCC
APEC CLIMATE CENTER

Contents

- Introduction to Weather Generator
- Temperature model
  - Normal distribution
  - Regression model
- Precipitation model
  - Gamma distribution
  - Semi-empirical model

Weather Generator

- Weather Generator
  - Generating *weather scenarios* based on *statistical model,*
  - TMAX, TMIN, Precipitation.
  - Observation data required
    - To be well quality-controlled (accuracy, no missing),
    - To be sufficiently long (30 years).

Weather Generator

- Weather Generator
  - Popularly used in agricultural and hydrological studies,
  - Observation data is limited to represent diverse inter/intra-seasonal variation of climate.
  - Weather Generator produces a great deal of scenarios to overcome such a limit.

Weather Generator

- Popular Weather Generators
  - WGEN (Richardson, 1981)
    - Gamma distribution: rainfall amount model,
    - Normal distribution: temperature anomaly model.
  - LARS-WG (Racsko et al., 1991)
    - Semi-empirical model: rainfall amount model.

Weather Generator

- Single-site Weather Generator
  - WGEN, LARS-WG: single site.

- Multisite Weather Generator
  - Covering a basin rather than a specific site.
  - Based on statistical model *describing weather/climate over multiple sites*.

Contents

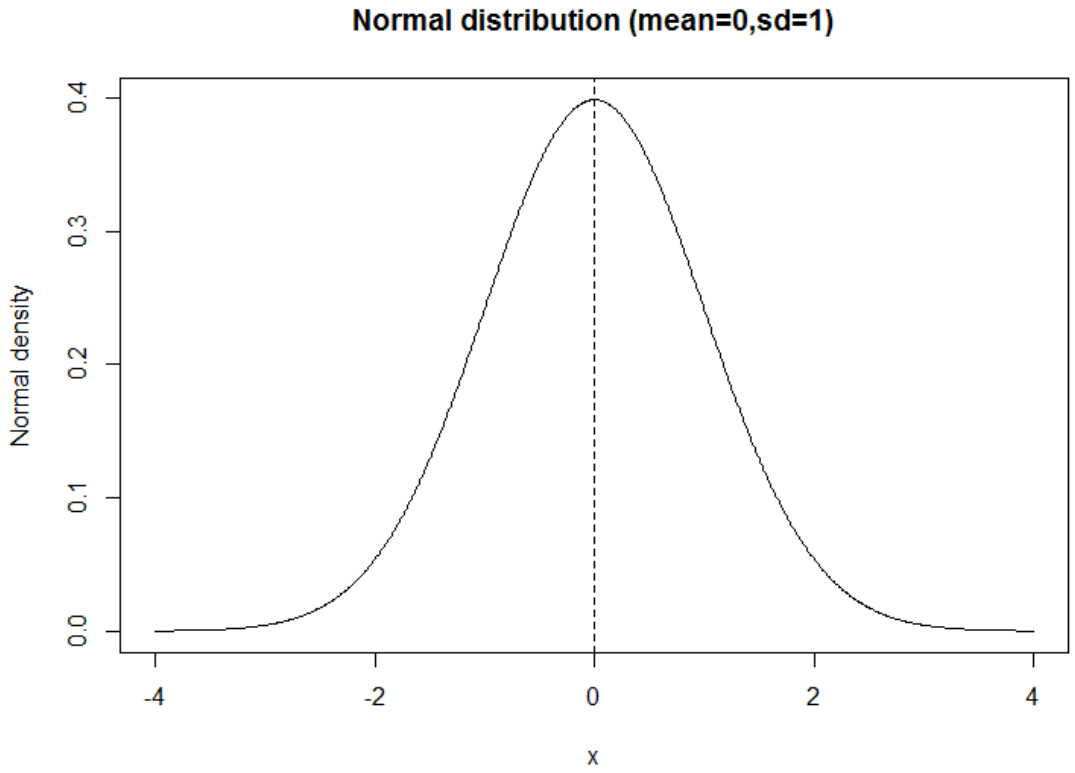
- Introduction to Weather Generator
- Temperature model
  - Normal distribution
  - Regression model
- Precipitation model
  - Gamma distribution
  - Semi-empirical model

Temperature Model

- Normal distribution
  - TMAX/TMIN *anomalies* follow Normal distribution.
  - Normal distribution Features:
    - Bell shape,
    - Parameter: mean, standard deviation,
    - Mean is equal to the center of distribution,
    - Standard deviation indicates the spread of distribution.

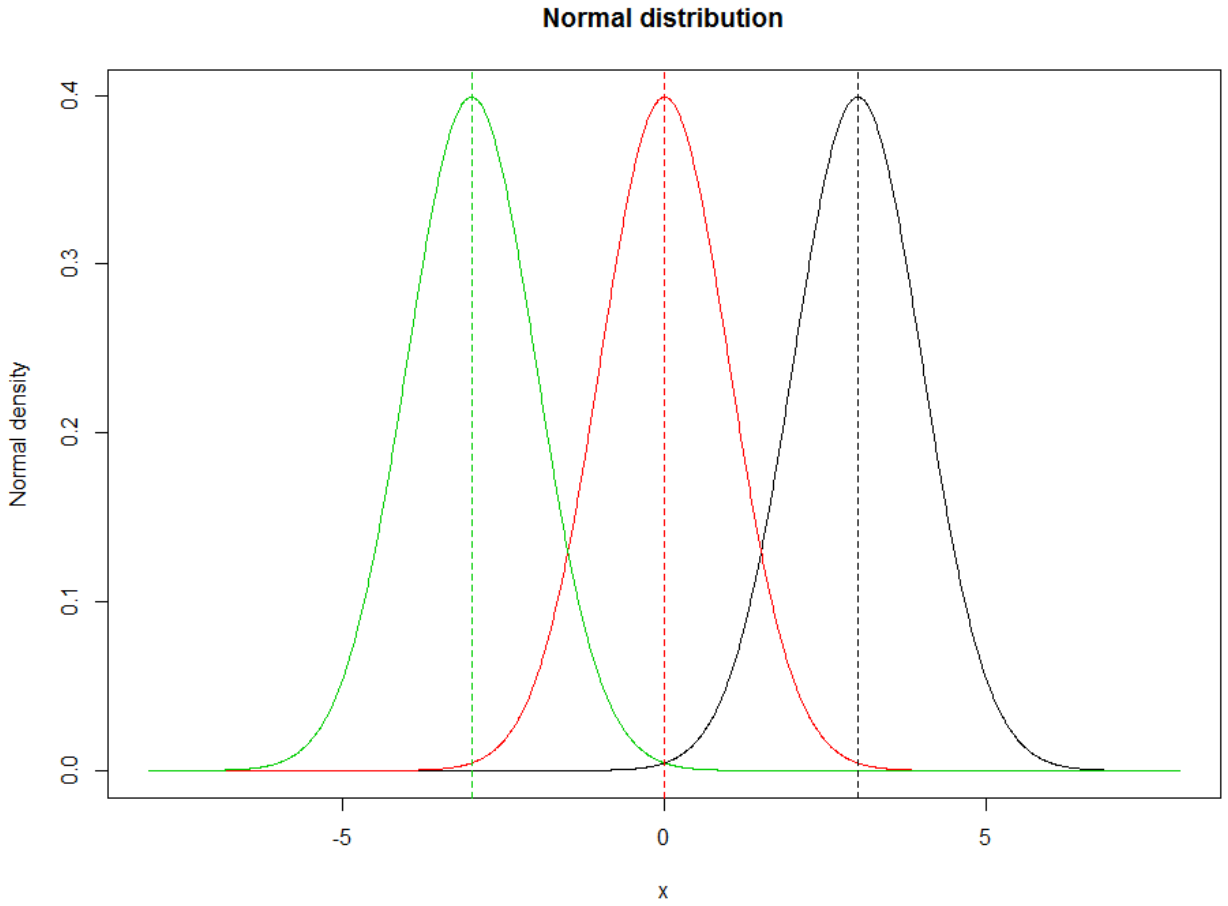
Temperature Model

- Normal distribution



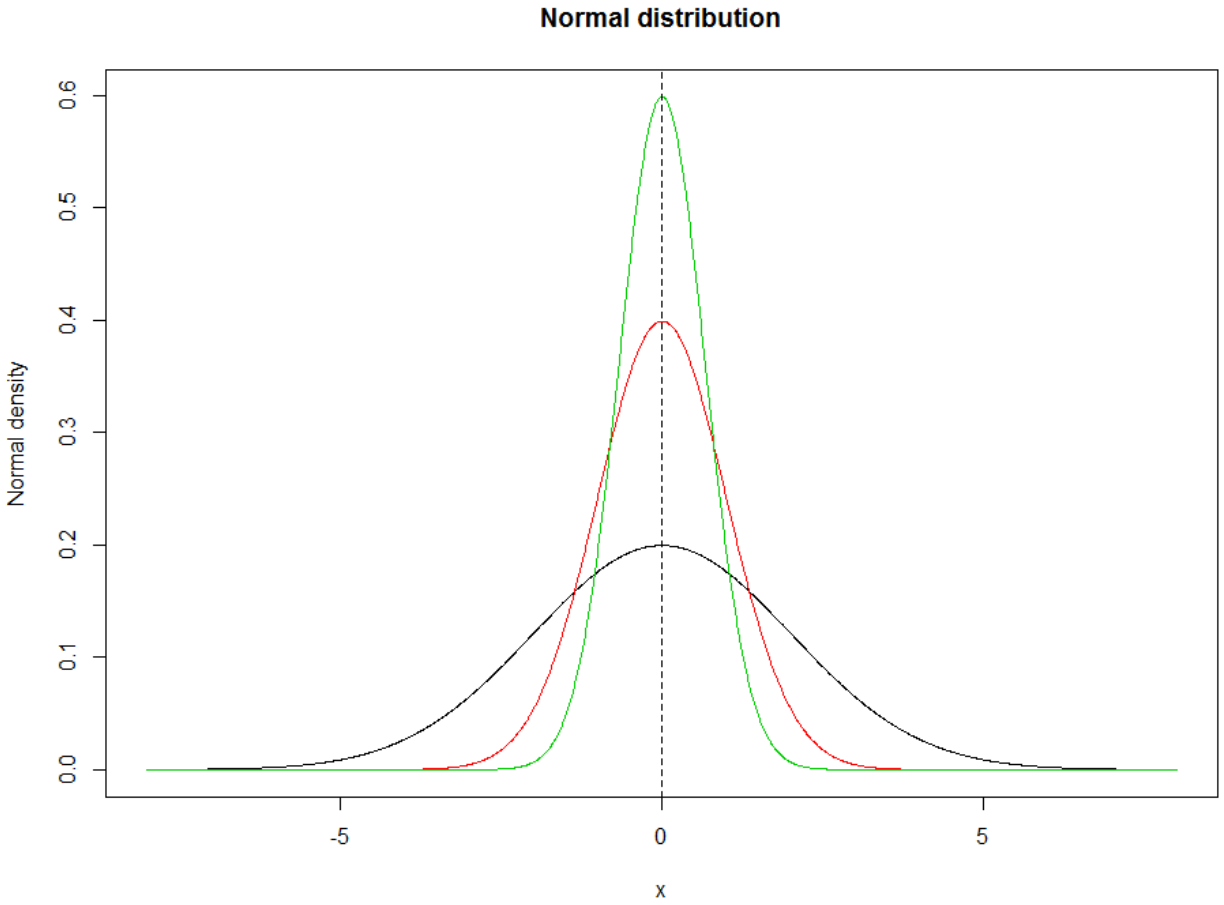
Temperature Model

- Normal distribution



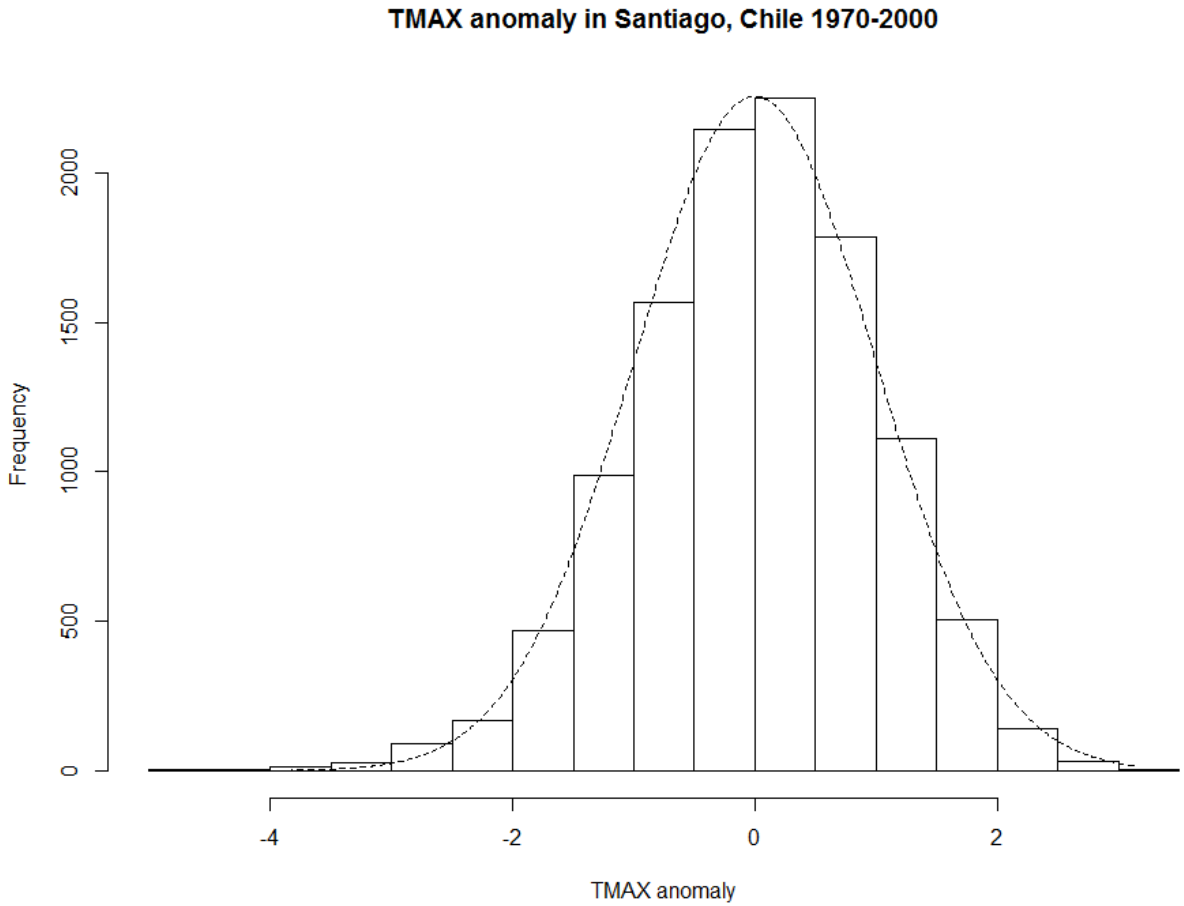
Temperature Model

- Normal distribution



Temperature Model

- Normal distribution for a real data



Temperature Model

- So what?
  - Normal distribution seems to be good!
  - Normal distribution serves as *mother distribution* for anomaly generation in Weather Generator.

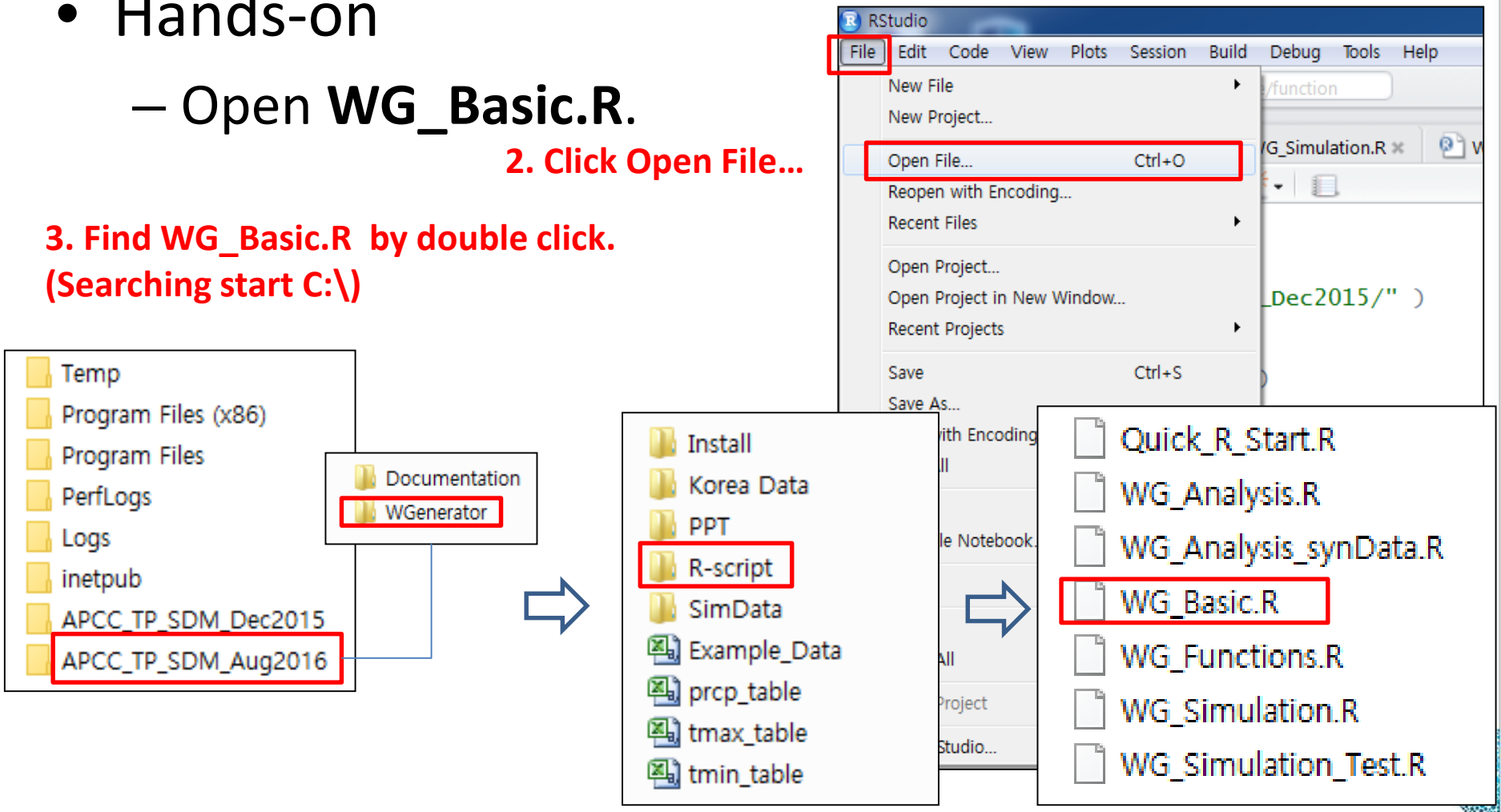
Temperature Model

- Hands-on
  - Open **WG_Basic.R**.

1. Click File

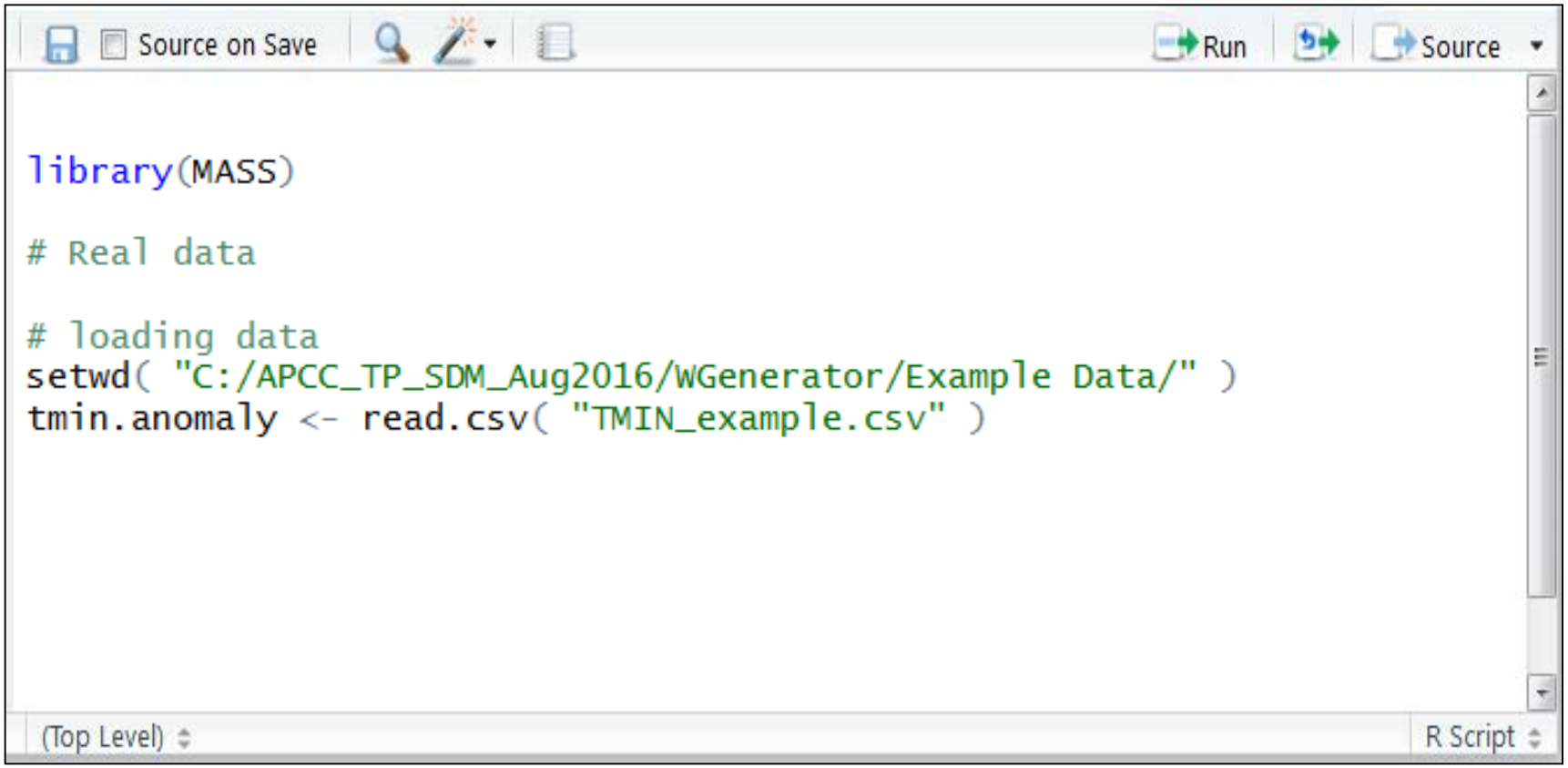
2. Click Open File…

3. Find WG_Basic.R by double click.
(Searching start C:\)



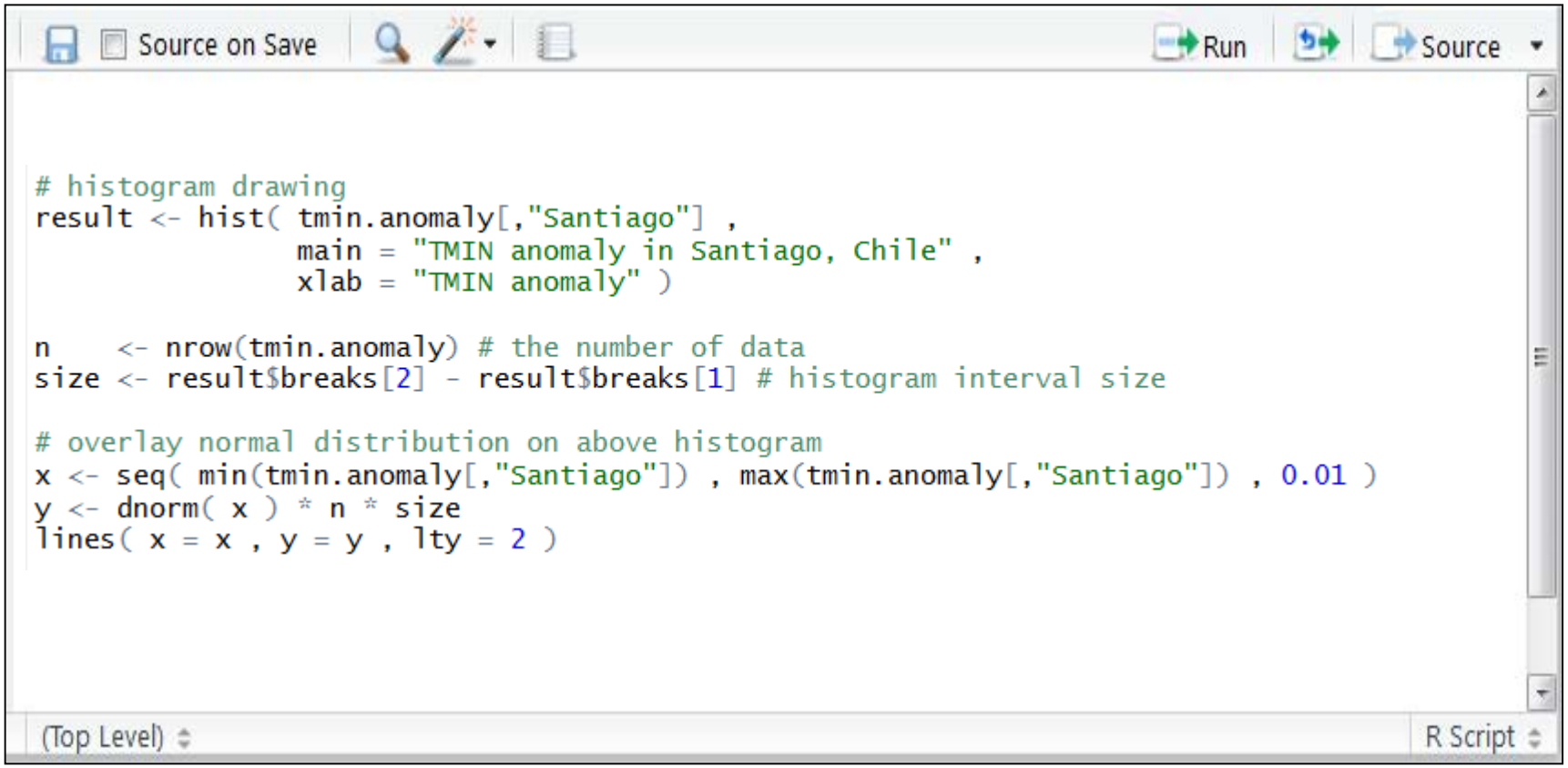
Temperature Model

- Hands-on



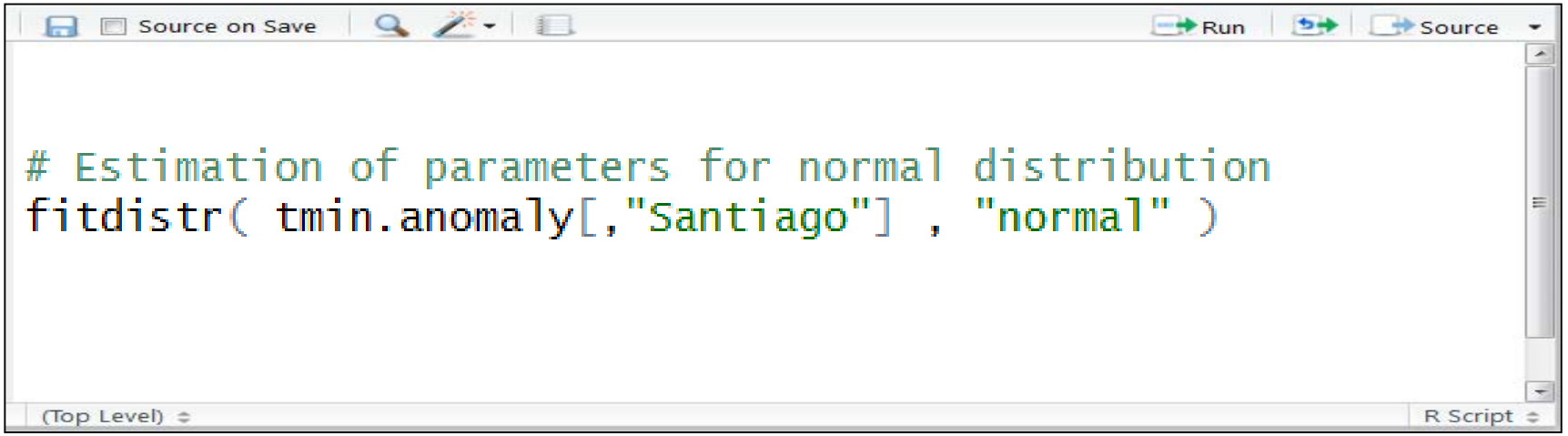
Temperature Model

- Hands-on



Temperature Model

- Hands-on
  - How to estimating parameters?
  - **fitdistr()** is used!



Temperature Model

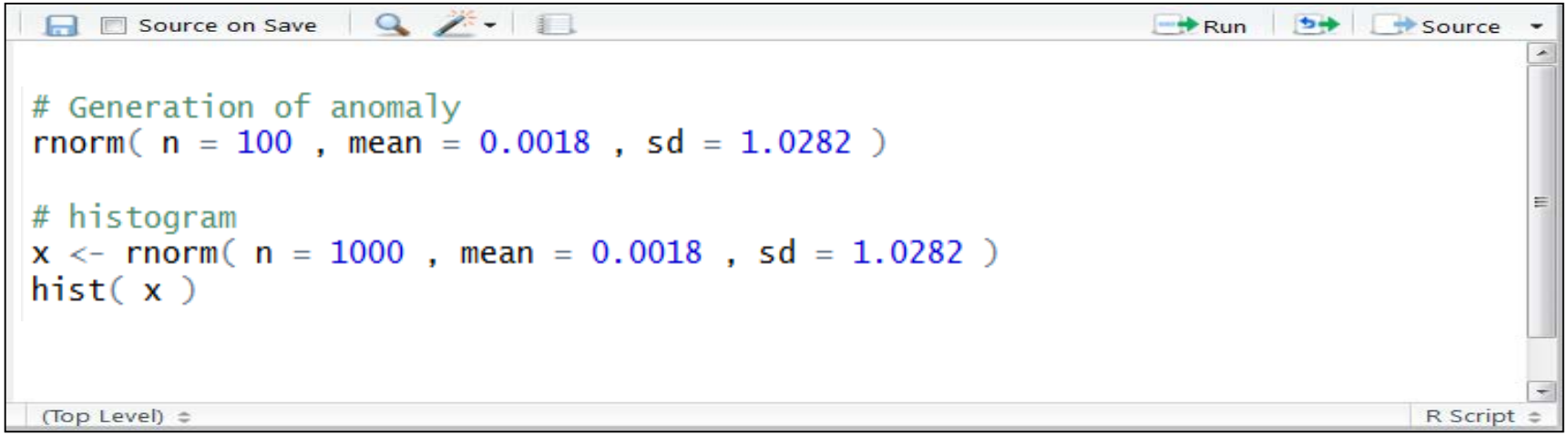
- Hands-on

```
Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/
> # Estimation of parameters for normal distribution
> fitdistr( tmin.anomaly[,"Santiago"] , "normal" )
      mean             sd
 0.001874483   1.028227524
(0.032515409) (0.022991866)
>
```

Note: Mean and standard deviation are almost 0 and 1, respectively, because anomaly is obtained by *standardizing* TMIN.

Temperature Model

- Hands-on
  - Using **rnorm()**, we generate anomaly!



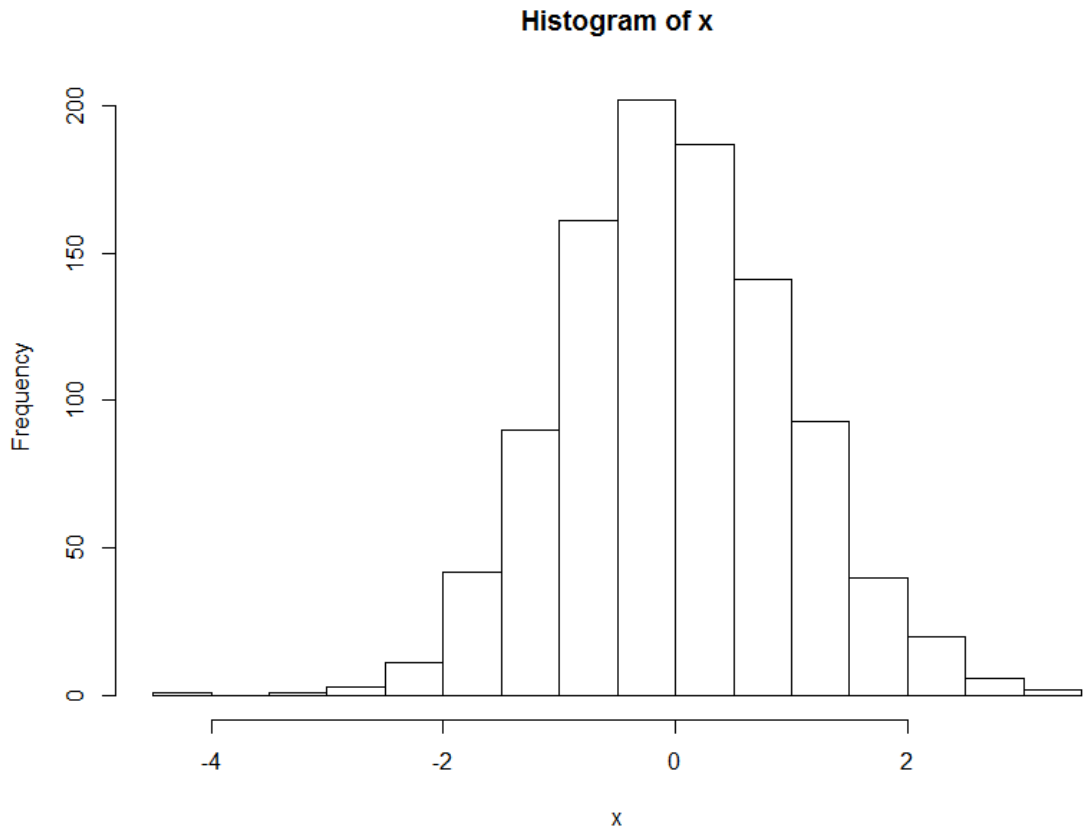
Temperature Model

- Hands-on
  - Using **rnorm()**, we generate anomaly!

```
Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/
> # Generation of anomaly
> rnorm( n = 100 , mean = 0.0018 , sd = 1.0282 )
  [1]  0.47500278 -0.87829780 -2.29084056 -0.71017919 -0.55587408  0.37255848 -1.68688370  0.84328022 -2.39830868 -0.89094406 -0.80494460
 [12] -0.56204142 -0.08467136  1.50457706  0.35686706  0.06370323  0.82422859 -0.59693756 -0.04544017  0.28637124  0.17550166 -0.37592209
 [23] -0.21164243 -0.09801094  0.02547301  0.23196749  1.24952318  0.39961456 -0.12246393 -0.25321039 -0.78331429  1.10814774 -0.68987234
 [34]  0.09170438  0.88271511 -0.27730836 -1.11094017 -0.46533027  0.21751153  1.60787906 -0.08595035 -1.28691387  1.87838346  1.70565195
 [45]  1.63619364  0.82102148  0.41344235 -0.18890839 -0.41609134  0.30355270  1.92547410 -1.62642977  0.97065773  0.18995292  0.15297201
 [56] -0.81289920 -0.91861636  0.27409177  0.59998381 -0.61125382  0.21809135  0.95279854 -0.28679113 -1.49370468  0.89313619  0.72065316
 [67] -1.24554859  0.48778109  0.41075628 -0.11539813 -1.72800017 -0.09478735 -0.45177810  0.17191641  0.23430957 -0.87544763 -1.93884048
 [78]  0.01031942 -1.83410137  1.39168276  1.27657189 -1.05020215 -0.33124318 -0.95643441  1.00895085  2.56563934  1.06910019  1.66298341
 [89] -1.20169099  1.59892974 -0.22367666  1.83100330 -0.44793955 -0.58240439 -0.18574037 -0.99913037 -0.95147432  0.29363568 -1.52771435
[100]  0.96207582
>
> # histogram
> x <- rnorm( n = 1000 , mean = 0.0018 , sd = 1.0282 )
> hist( x )
>
```

Temperature Model

- Hands-on

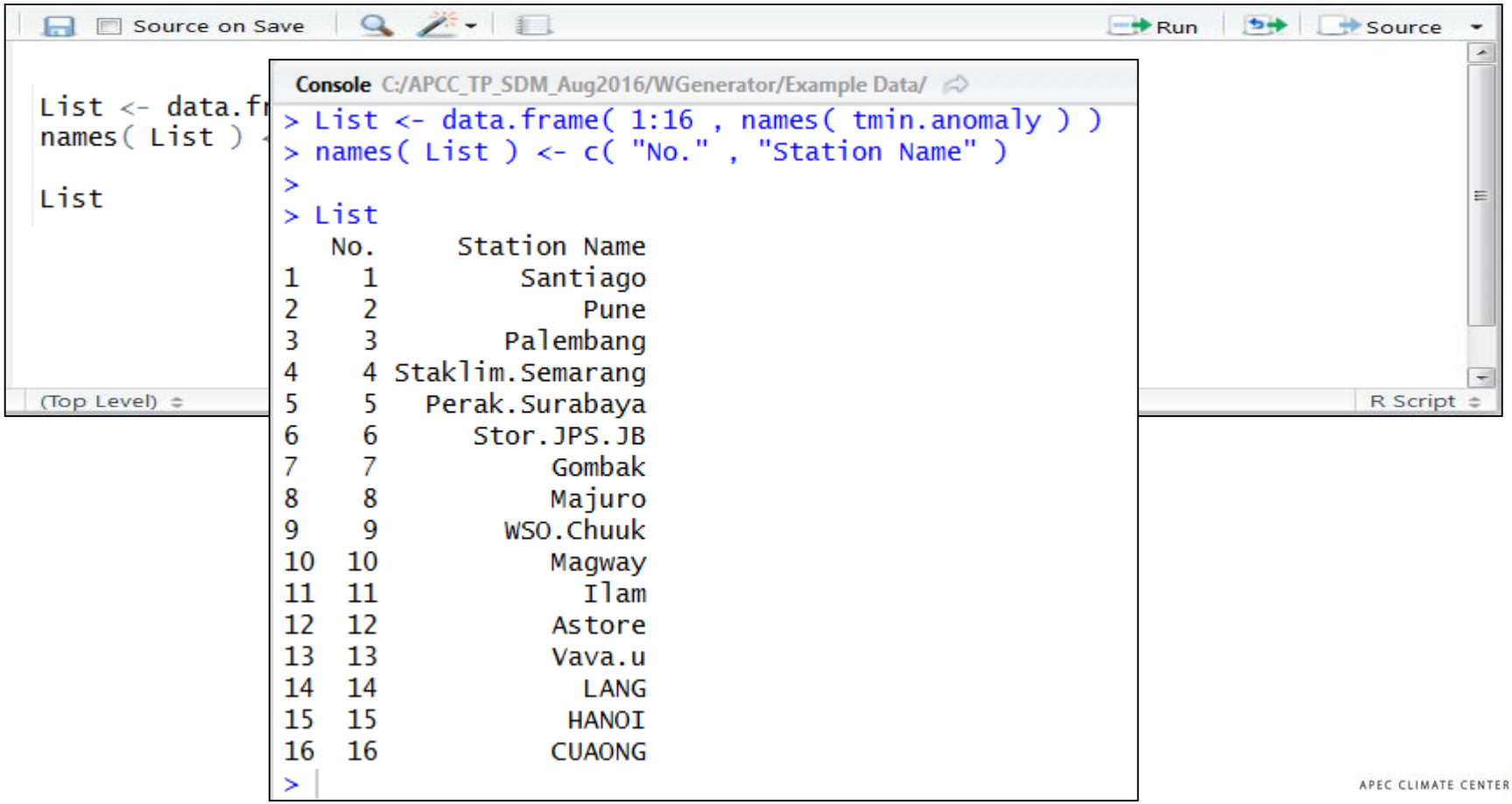


Temperature Model

- Practice: Check Normality for another station!
  - Loaded **tmin.anomaly** has many other anomaly data.
  - Choose one of them and test normality!

Temperature Model

- Practice: Check Normality for another station!



Temperature Model

- Practice: Check Normality for another station!

Source on Save | Run | Source

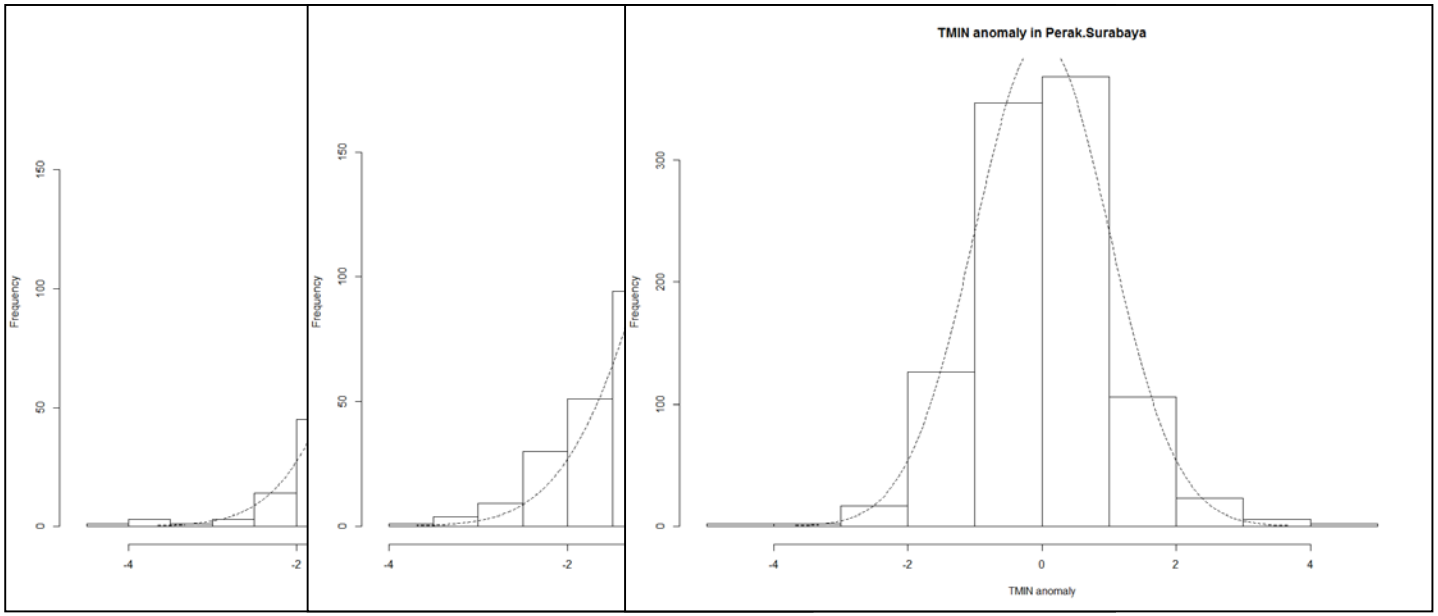
```
# Choose a station
j <- 16 # j is No. in List


# histogram drawing
result <- hist( tmin.anomaly[,j] ,
                main = paste( "TMIN anomaly in" , List[j,2] ) ,
                xlab = "TMIN anomaly" )

n    <- sum( !is.na( tmin.anomaly[,j] ) ) # the number of data
size <- result$breaks[2] - result$breaks[1] # histogram interval size

# overlay normal distribution on above histogram
x <- seq( min( tmin.anomaly[,j] , na.rm = T ) , max( tmin.anomaly[,j] , na.rm = T ) , 0.01 )
y <- dnorm( x ) * n * size
lines( x = x , y = y , lty = 2 )
```

Change 16 to another number you want!

(Top Level) | R Script

Temperature Model

- Practice: Check Normality for another station!



Temperature Model

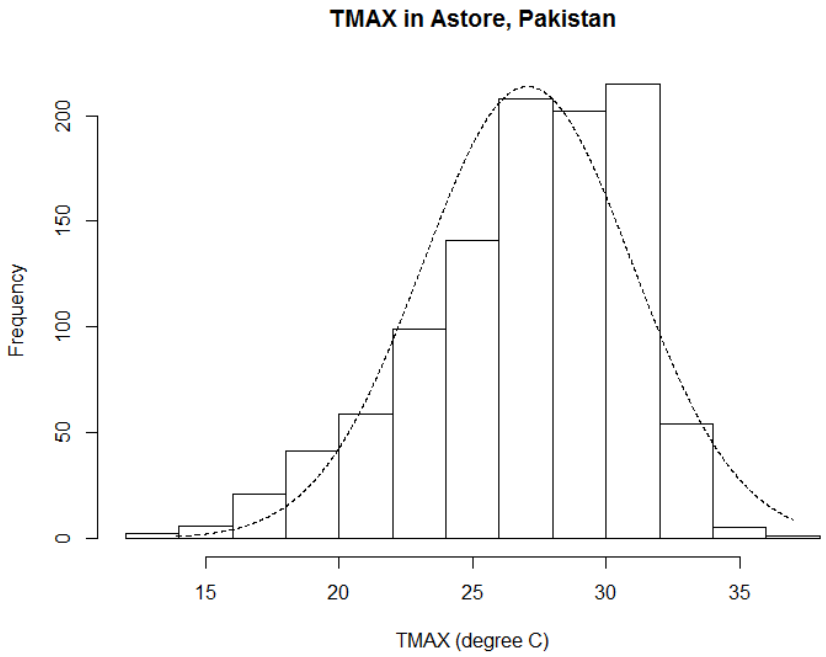
- Regression Model
  - Temperature is affected by precipitation.
  - Precipitation entails:
    - Cloudy weather,
    - Short or no sunshine duration time, low solar radiation,
    - High humidity, slow diurnal variation of temperature.
    - TMAX tends to decrease. In dry cold season, TMIN tends to increase.

Temperature Model

- Regression Model
  - Simple model: estimating parameters for wet/dry day, separately. (WGEN, LARS-WG)
  - Regression model
    - Precipitation effect depends on rainfall intensity,
    - Regression model provides quantitative relation,
    - Regression model is appropriate for multisite cases.

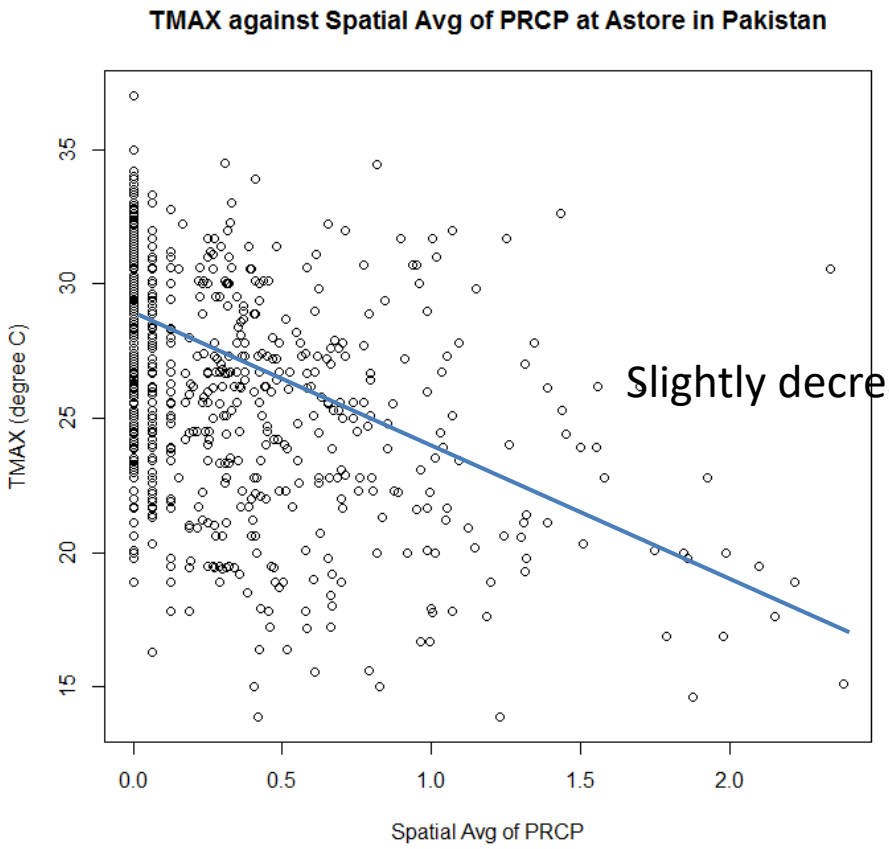
Temperature Model

- Regression Model
  - TMAX distribution, (not its anomaly)
  - Far from Normal! Why?



Temperature Model

- Regression Model



Slightly decreasing tendency

Temperature Model

- Regression Model
  - Assume $\mathrm{TMAX} = a + b \times \mathrm{AVG} + (\mathrm{error})$.
  - How to estimate intercept(a) and coefficient(b)?
  - Note: coefficient is expected to be negative.
  - **lm()** is used for estimating regression model.

Temperature Model

- Hands on

Source on Save | Run | Source

```
# Regression model

# data loading
setwd( "C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/" )
data.table <- read.csv( "PRCP_TMAX_Pakistan.csv" )

# scatter plot TMAX against spatial average of precipitation
plot( data.table[,c("AVG","Astore")] ,
      xlab = "Spatial Avg of PRCP" , ylab = "TMAX (degree C)" ,
      main = "TMAX against Spatial Avg of PRCP at Astore in Pakistan" )

# regression model fitting
names( data.table ) # colnames checking
lm( Astore ~ AVG , data = data.table )
# checking the regression coefficient!
# TMAX = a + b * AVG + (error)
```

Regression model estimation!

Astore ~ AVG*: Regression formula*

(Top Level) | R Script

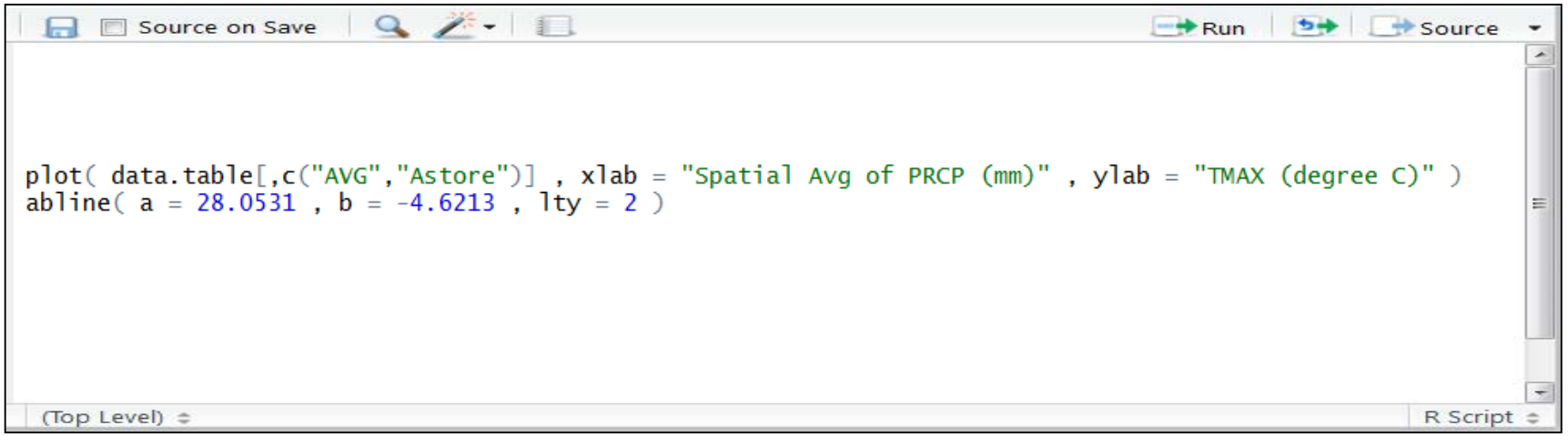
Temperature Model

- Hands on

```
Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/
> # data loading
> setwd( "C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/" )
> data.table <- read.csv( "PRCP_TMAX_Pakistan.csv" )
>
> # scatter plot TMAX against spatial average of precipitation
> plot( data.table[,c("AVG","Astore")] ,
+       xlab = "Spatial Avg of PRCP" , ylab = "TMAX (degree C)" ,
+       main = "TMAX against Spatial Avg of PRCP at Astore in Pakistan" )
>
> # regression model fitting
> names( data.table ) # colnames checking
[1] "Astore" "Bunji"  "Gupis"  "Giglit" "AVG"
> lm( Astore ~ AVG , data = data.table )

Call:
lm(formula = Astore ~ AVG, data = data.table)

Coefficients:
(Intercept)          AVG
     28.053       -4.621

> # checking the regression coefficient!
> # TMAX = a + b * AVG + (error)
>
```

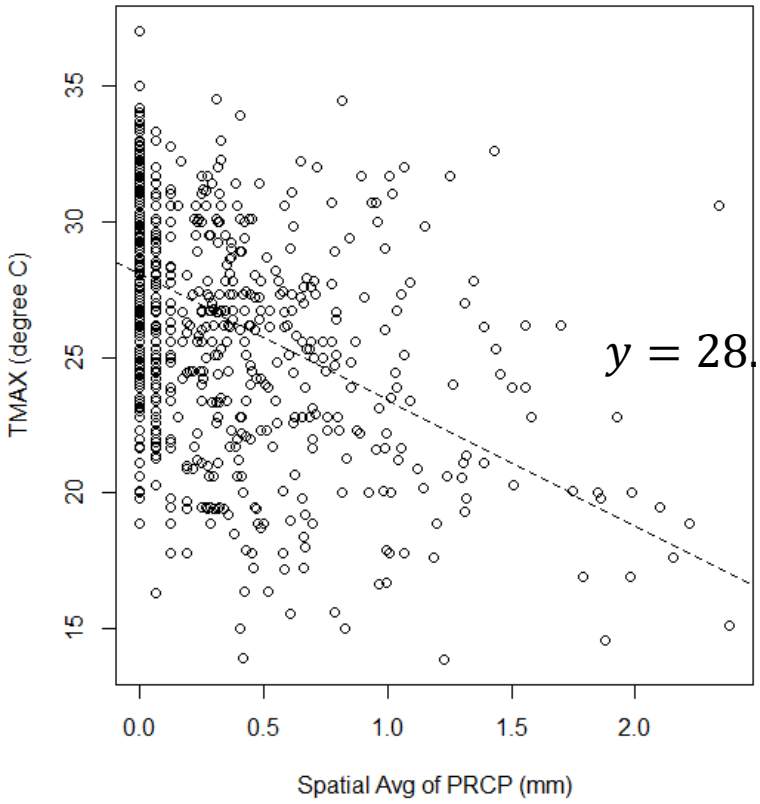
Temperature Model

- Hands on



Temperature Model

- Hands on



$$y = 28.05331 - 4.6213x$$

Temperature Model

- Practice: Regression modeling
  - Loaded **data.table** contains the sites in the basin.
  - Choose one of the sites and fit regression model
    - Plotting AVG vs. TMAX of the chosen site,
    - Estimating intercept and coefficient!

Temperature Model

- Practice: Regression modeling

Source on Save | Run | Source

```
# Practice: Regression model fitting
# Note: data.table contain TMAX data at the other sites in Pakistan Basin.
# Choose another site and fitting regression model

site.names <- names( data.table )[-5]
List <- data.frame( 1:4 , site.names )
names( List ) <- c( "No." , "Station Name" )

List
```

(Top Level) | R Script

Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/

```
> site.names <- names( data.table )[-5]
> List <- data.frame( 1:4 , site.names )
> names( List ) <- c( "No." , "Station Name" )
>
> List
  No. Station Name
1   1       Astore
2   2        Bunji
3   3        Gupis
4   4       Giglit
>
```

Temperature Model

- Practice: Regression modeling

Source on Save | Run | Source

```
# Choose a station
j <- 2 # j is No. in List

# plotting
plot( data.table[,c("AVG",site.names[j])] ,
      xlab = "Spatial Avg of PRCP (mm)" , ylab = "TMAX (degree C)" )

# Regression model fitting
f <- as.formula( paste( site.names[j] , "AVG" , sep = "~" ) )
lm( f , data.table )
```

Change 2 to another number you want!

(Top Level) | R Script

Contents

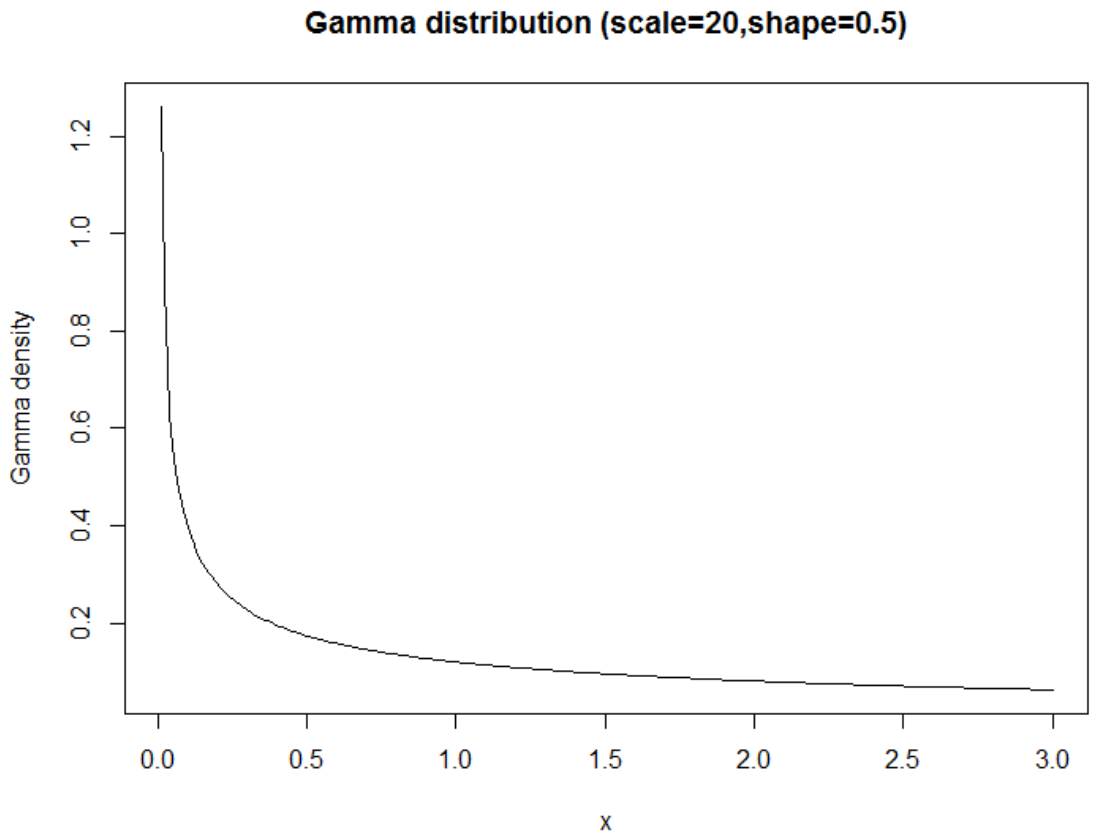
- Introduction to Weather Generator
- Temperature model
  - Normal distribution
  - Regression model
- Precipitation model
  - Gamma distribution
  - Semi-empirical model

Precipitation Model

- Gamma Distribution
  - Precipitation distribution is skewed.
  - Gamma distribution is popularly employed.
  - Parameter: scale, shape.

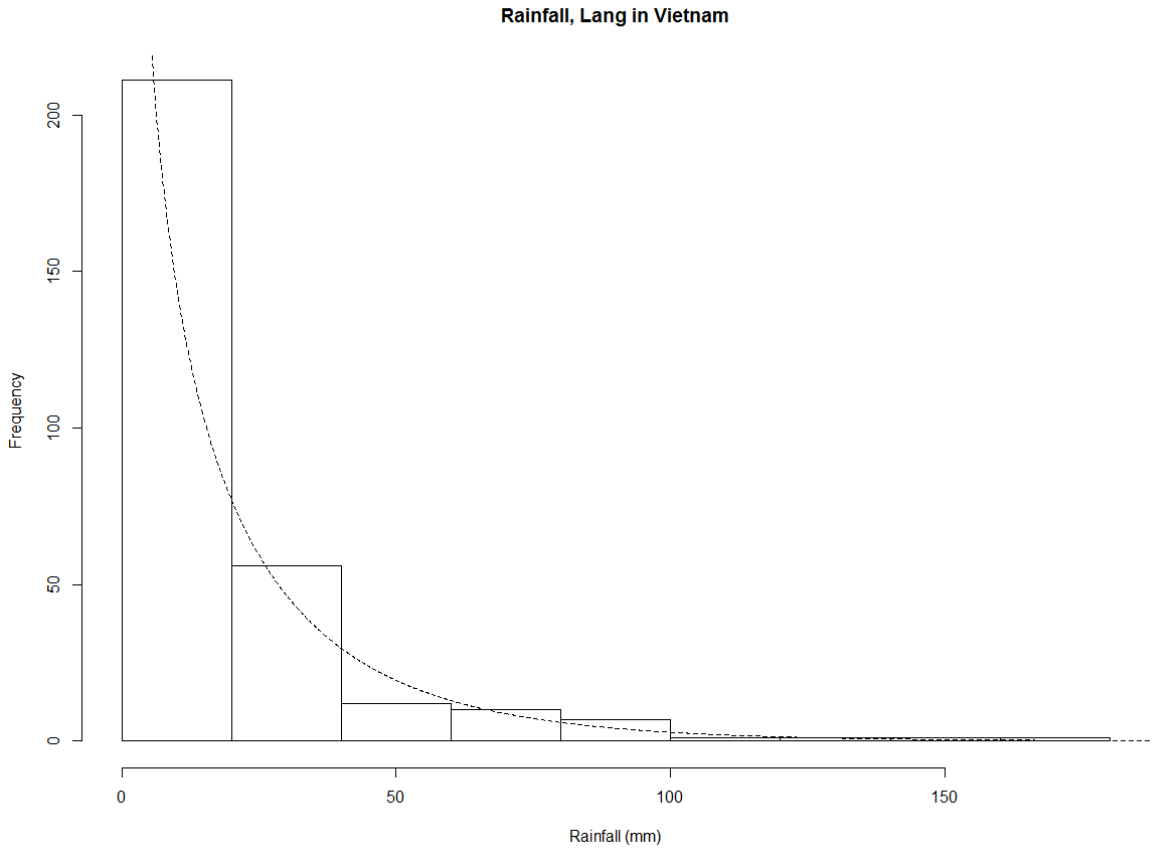
Precipitation Model

- Gamma Distribution



Precipitation Model

- Gamma Distribution



Precipitation Model

- Hands on

Source on Save | Run | Source

```
# Real data
setwd( "C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/" )
rainfall.data <- read.csv( "PRCP_examples.csv" )

# Gamma distribution fitting
fitdistr( rainfall.data[,"LANG"] ,
          densfun = dgamma ,
          start   = list( shape = 1 , scale = 1 ),
          lower = 0.001 )

# histogram and Gamma distribution
result <- hist( rainfall.data[,"LANG"] ,
                main = "Rainfall, Lang in Vietnam" ,
                xlab = "Rainfall (mm)" )

n    <- length(rainfall.data[,"LANG"])
size <- result$breaks[2] - result$breaks[1]

x <- seq(0,200,0.1)
y <- dgamma( x , shape = 0.5570 , scale = 30.774 )
lines( x , y * n * size , lty = 2 )
```

(Top Level) | R Script

Estimation of parameters of Gamma distribution

Precipitation Model

- Hands on

Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/

```
> setwd( "C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/" )
> rainfall.data <- read.csv( "PRCP_examples.csv" )
>
> # Gamma distribution fitting
> fitdistr( rainfall.data[,"LANG"] ,
+           densfun = dgamma ,
+           start   = list( shape = 1 , scale = 1 ),
+           lower = 0.001 )
      shape          scale
  0.55707684   30.77432936
( 0.03792981) ( 3.17132181)
>
> # histogram and Gamma distribution
> result <- hist( rainfall.data[,"LANG"] ,
+                 main = "Rainfall, Lang in Vietnam" ,
+                 xlab = "Rainfall (mm)" )
>
> n    <- length(rainfall.data[,"LANG"])
> size <- result$breaks[2] - result$breaks[1]
>
> x <- seq(0,200,0.1)
> y <- dgamma( x , shape = 0.5570 , scale = 30.774 )
> lines( x , y * n * size , lty = 2 )
>
```

Estimation result!

Precipitation Model

- Practice: Gamma modeling
  - Loaded **rainfall.data** has many other precipitation data.
  - Choose one of them and test whether or not Gamma distribution is appropriate!

Precipitation Model

- Practice: Gamma modeling

Source on Save | Run | Source

```
# Prcatice: Gamma modeling

List <- data.frame( 1:18 , names( rainfall.data ) )
names( List ) <- c( "No." , "Station Name" )

List
```

```
> List
   No.      Station Name
1    1          Santiago
2    2          Agartala
3    3              Pune
4    4          Palembang
5    5 Staklim.Semarang
6    6   Perak.Surabaya
7    7      Stor.JPS.JB
8    8            Gombak
9    9            Majuro
10  10         WSO.Chuuk
11  11            Magway
12  12              Ilam
13  13            Astore
14  14       Chiang.Mai
15  15            Vava.u
16  16              LANG
17  17             HANOI
18  18            CUAONG
>
```

(Top Level) | R Script

Precipitation Model

- Practice: Gamma modeling

Source on Save | Run | Source

```
# Choose a station
j <- 4 # j is No. in List
```

Change 4 to another number you want!

```
fit.result <- fitdistr( rainfall.data[,j] ,
          densfun = dgamma ,
          start   = list( shape = 1 , scale = 1 ),
          lower = 0.001 )

fit.result

# histogram and Gamma distribution
result <- hist( rainfall.data[,j] ,
                main = paste("Rainfall, " , names(rainfall.data)[j] , sep = "" ),
                xlab = "Rainfall (mm)" )

n    <- length(rainfall.data[,j])
size <- result$breaks[2] - result$breaks[1]

x <- seq(0,200,0.1)
y <- dgamma( x , shape = fit.result$estimate[1] , scale = fit.result$estimate[2] )
lines( x , y * n * size , lty = 2 )
```

(Top Level) | R Script

Precipitation Model

- Semi-empirical modeling
  - What if Gamma is not good?
  - Semi-empirical model as alternative (LARS-WG).

Precipitation Model

- Semi-empirical modeling
  - Divide the whole range into several intervals
    - break points: 0,10,20,…,90, 100% *quantiles*,
    - 10 intervals.
  - Choose one of them randomly.
  - Randomly choose a value from the selected interval (uniform distribution).

Precipitation Model

- Hands on

Source on Save | Run | Source

```
# select a site
List <- data.frame( 1:18 , names( rainfall.data ) )
names( List ) <- c( "No." , "Station Name" )

List

j <- 1
```

Change 1 to another number you want!

```
> List
   No.      Station Name
1    1          Santiago
2    2          Agartala
3    3              Pune
4    4         Palembang
5    5 Staklim.Semarang
6    6   Perak.Surabaya
7    7       Stor.JPS.JB
8    8            Gombak
9    9            Majuro
10  10         WSO.Chuuk
11  11            Magway
12  12              Ilam
13  13            Astore
14  14       Chiang.Mai
15  15            Vava.u
16  16              LANG
17  17             HANOI
18  18            CUAONG
>
```

(Top Level) | R Script

Precipitation Model

- Hands on

Source on Save | Run | Source

```
# 1. Divide the whole range into several intervals
# 2. Choose one of the intervals randomly.
# 3. Randomly choose a value from the selected interval (uniform distribution).

# 1. Divide the whole range into several intervals:
# break points: 0,10,20,...,90, 100 % quantiles
# 10 intervals
( break.points <- quantile( rainfall.data[,j] , seq( 0 , 1 , 0.1 ) ) )

# 2. Choose one of 10 intervals randomly.
( i <- sample( 10 , 1 ) )
# i = 1 : the first interval
# i = 2 : the second interval
# ...
# i = 10 : the last interval

# 3. Randomly choose a value from the selected interval (uniform distribution).
runif( 1 , min = break.points[i] , max = break.points[i+1] )
```

(Top Level) | R Script

Precipitation Model

- Hands on

```
Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/
> # 1. Divide the whole range into several intervals
> # 2. Choose one of the intervals randomly.
> # 3. Randomly choose a value from the selected interval (uniform distribution).
>
> # 1. Divide the whole range into several intervals:
> # break points: 0,10,20,...,90, 100 % quantiles
> # 10 intervals
> ( break.points <- quantile( rainfall.data[,j] , seq( 0 , 1 , 0.1 ) ) )
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
 0.10  0.20  0.50  1.40  3.18  5.85  8.80 13.28 19.54 29.34 68.70
>
> # 2. Choose one of 10 intervals randomly.
> ( i <- sample( 10 , 1 ) )
[1] 3
> # i = 1 : the first interval
> # i = 2 : the second interval
> # ...
> # i = 10 : the last interval
>
> # 3. Randomly choose a value from the selected interval (uniform distribution).
> runif( 1 , min = break.points[i] , max = break.points[i+1] )
[1] 1.323441
>
```

Break points

Choose one of 1,2,…,10.

Since i = 3*, we choose a value from interval* (0.50 , 1.40).

Precipitation Model

- Gamma vs. Semi-empirical Modeling

Source on Save | Run | Source

```
# function
semi.empirical.generation <- function( n )
{
    # 1. Divide the whole range into several intervals:
    # We use break.points calculated above.

    # 2. Choose one of 10 intervals, repeatedly n times
    s <- sample( 10 , n , replace = T )

    # 3. Randomly choose a value from the selected intervals (uniform distribution).
    x <- c()
    for( k in 1:n )
    {
        x[k] <- runif( 1 , min = break.points[s[k]] , max = break.points[s[k]+1] )
    }

    return(x)
}
```

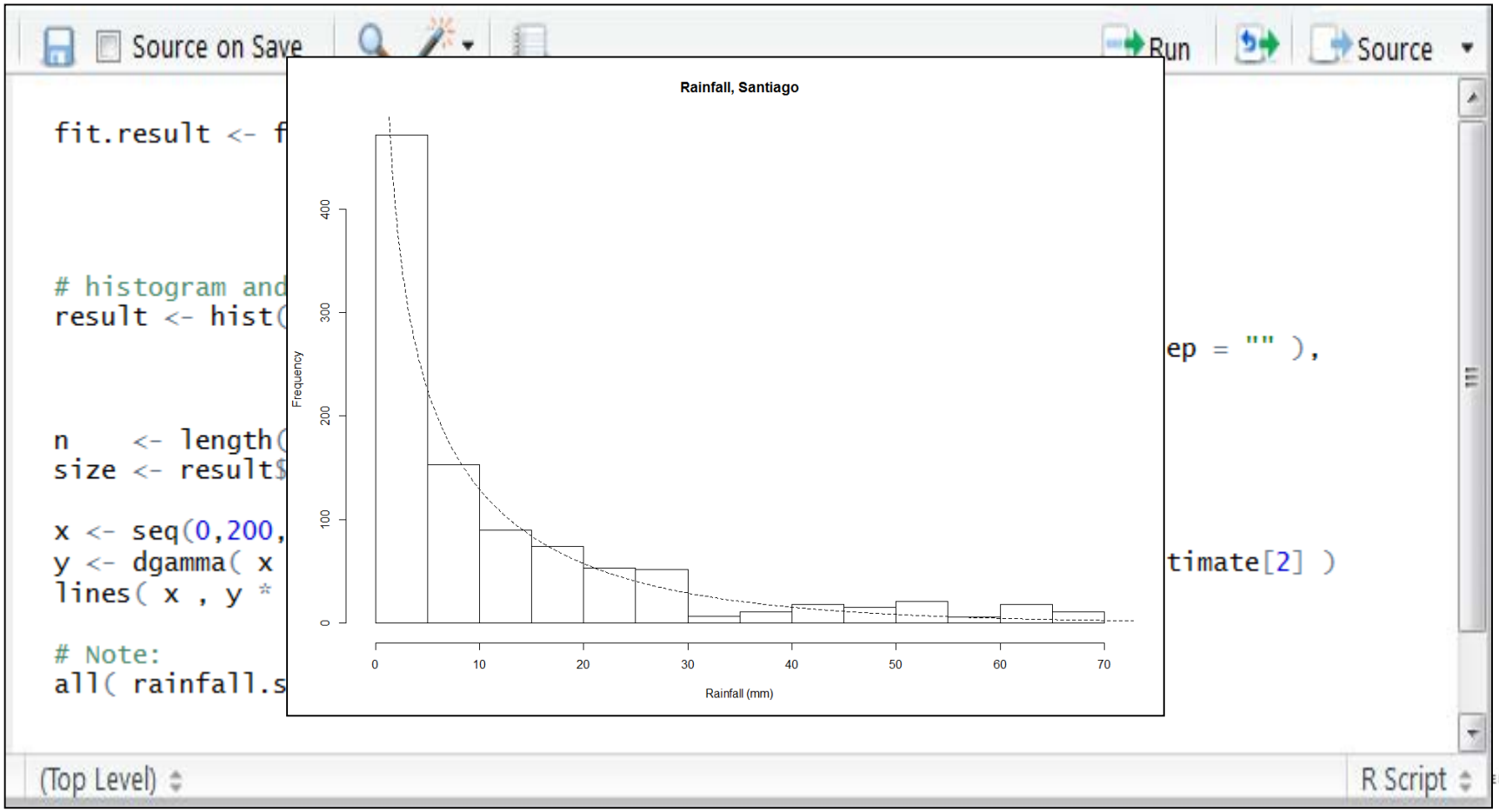
1000 rainfall numbers generated!

```
rainfall.sim <- semi.empirical.generation( 1000 )
```

(Top Level) | R Script

Precipitation Model

- Gamma vs. Semi-empirical Modeling



Precipitation Model

- Gamma vs. Semi-empirical Modeling

```
Console C:/APCC_TP_SDM_Aug2016/WGenerator/Example Data/
> fit.result <- fitdistr( rainfall.data[,j] ,
+                         densfun = dgamma ,
+                         start   = list( shape = 1 , scale = 1 ),
+                         lower = 0.001 )
>
> # histogram and Gamma distribution
> result <- hist( rainfall.sim ,
+                 main = paste("Rainfall, " , names(rainfall.data)[j] , sep = "" ),
+                 xlab = "Rainfall (mm)" )
>
> n    <- length(x)
> size <- result$breaks[2] - result$breaks[1]
>
> x <- seq(0,200,0.1)
> y <- dgamma( x , shape = fit.result$estimate[1] , scale = fit.result$estimate[2] )
> lines( x , y * n * size , lty = 2 )
>
> # Note:
> all( rainfall.sim <= max(rainfall.data[,j]) )
[1] TRUE
>
```

All simulation values are less than maximum of observations!

Precipitation Model

- Comment on Semi-empirical Modeling
  - Distributional distortion in tail part,
  - Generation scheme can not produce *values greater than maximum in observation period*.
  - *Extreme value modeling* needed.

The End of Session 2

Welcome any Questions.

APCC
APEC CLIMATE CENTER