A Very Big League Predictive Algorithm

Me

Washington University in St. Louis

November 04, 2016

Motivation

Motivation

Riegel's pace predictor formula: suppose your most recent run was D_1 units of length and took you T_1 unites of time, then based on this activity your next run of distance D_2 will take you

$$T_2 = T_1 \left(\frac{D_2}{D_1} \right)^{1.06}$$

Motivation

Riegel's pace predictor formula: suppose your most recent run was D_1 units of length and took you T_1 unites of time, then based on this activity your next run of distance D_2 will take you

$$T_2 = T_1 \left(\frac{D_2}{D_1} \right)^{1.06}$$

WHERE IS THE BIG DATA???

Motivation

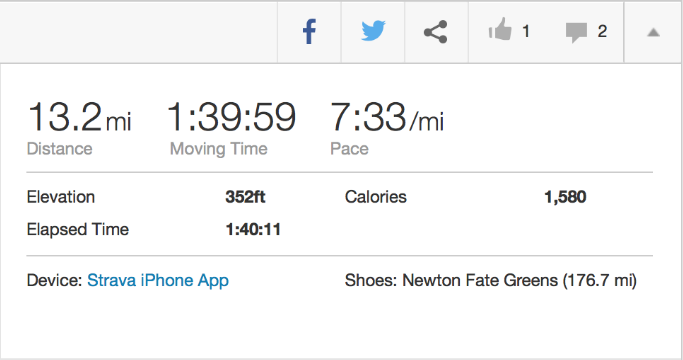
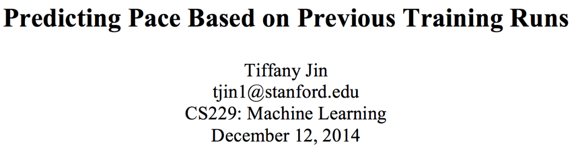


Figure: A summary of a Strava activity



Predicting Pace Based on Previous Training Runs

Tiffany Jin
tjin1@stanford.edu
CS229: Machine Learning
December 12, 2014

Figure: Recent literature

When ↓	Type	Gear	Name	City	State	Dist mi	Elv ft	Elapsed Time	Moving Time	Speed mph	Max Speed mph	Pace /mi
11/11/2014	Run	Saucony ProGrid OMNI 12	so much to do so little t	Shenzhen	Guangdong	2.17	0	00:21:00	00:21:00	6.2	0.0	09:39
11/10/2014	Run	Saucony ProGrid OMNI 12	Early morning treadmill			3.29	0	00:31:00	00:31:00	6.4	0.0	09:25
11/09/2014	Run	Saucony ProGrid OMNI 12	trying to stay awake in s			3.23	0	00:30:00	00:30:00	6.5	0.0	09:17
11/05/2014	Run	Saucony ProGrid OMNI 12	drowning in machine lea	San Jose	California	3.76	0	00:32:39	00:32:39	6.9	9.4	08:41
11/02/2014	Run	Saucony ProGrid OMNI 12	Maisie's peak	Cupertino	CA	6.47	791	01:05:17	01:02:35	6.2	8.5	09:40
10/31/2014	Run	Saucony Kinvara 4 (purple)	10/31/2014 Cupertino,	Cupertino	CA	5.37	581	00:48:50	00:48:22	6.7	11.2	09:01
10/30/2014	Run	Saucony ProGrid OMNI 12	yep, out of shape	Cupertino	CA	5.29	545	00:49:08	00:48:34	6.5	8.9	09:10
10/28/2014	Run	Saucony ProGrid OMNI 12	grumble grumble	Cupertino	CA	3.01	112	00:25:50	00:25:23	7.1	10.5	08:26
10/26/2014	Run	Saucony Kinvara 4 (purple)	break from the wonderl	Cupertino	CA	7.33	892	01:14:31	01:11:17	6.2	8.5	09:44
10/23/2014	Run	Saucony ProGrid OMNI 12	cabin fever	Cupertino	CA	3.01	75	00:24:40	00:24:40	7.3	10.7	08:12
10/18/2014	Run	Saucony ProGrid OMNI 12	sb run group	Campbell	California	8.16	128	01:12:45	01:10:50	6.9	9.2	08:41
10/16/2014	Run	Saucony ProGrid OMNI 12	Apple run club with Ari &	Cupertino	California	3.24	66	00:31:02	00:28:36	6.8	8.1	08:49
10/15/2014	Run	Saucony ProGrid OMNI 12	First lunch run since the	San Jose	California	6.01	0	00:53:41	00:52:40	6.8	8.5	08:46
10/13/2014	Run	Saucony ProGrid OMNI 12	working off food & boo	Cupertino	CA	3.03	82	00:25:13	00:25:08	7.2	10.7	08:17
10/09/2014	Run	Saucony ProGrid OMNI 12	still recovering, I guess	Cupertino	CA	3.52	177	00:31:59	00:30:48	6.9	10.1	08:45
10/06/2014	Run	Saucony ProGrid OMNI 12	unwillingly dragged alon	Cupertino	CA	3.01	75	00:28:45	00:28:45	6.3	8.1	09:34
10/05/2014	Run	Saucony Kinvara 4 (purple)	San Jose Rock n Roll Ha	San Jose	California	13.23	39	01:48:14	01:48:14	7.3	13.4	08:11
10/03/2014	Run	Saucony Kinvara 4 (purple)	chatting	Cupertino	CA	3.00	62	00:27:55	00:27:55	6.5	8.1	09:18
10/01/2014	Run	Saucony Kinvara 4 (purple)	Campus loop	Stanford	California	3.93	69	00:33:51	00:33:51	7.0	9.2	08:37

Motivation

Model	Training samples	Test samples	Training MSE (error in minutes)	Test MSE (error in minutes)
Basic linear regression	358	89	0.6403	0.5517
Locally weighted regression	358	89	0.4624	0.5521
Ridge regression	358	89	0.4615	0.5559
Lasso regression	358	89	0.5722	0.5682

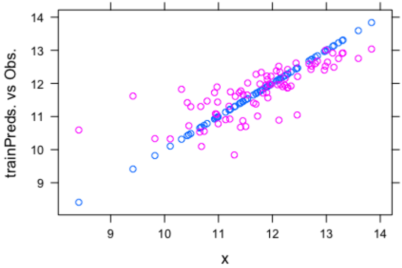
Figure: Tiffany Jin's results.

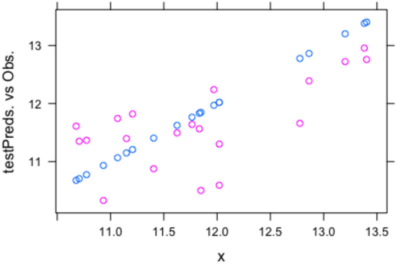
Features

1. Y, the average speed during the activity
2. $X_1, \ldots, X_5$ the distance ran, total elevation gain, temperature, humidity, and number of days from today.

```r
library("caret")

data <- read.csv(file = "dataFinal.csv")

speed <- data[,1]

split <- createDataPartition(speed, p = 0.8,
                  list = FALSE)

trainData <- data[split, ]
testData <- data[-split, ]

trainVals <- speed[split]
testVals <- speed[-split]

olsFit <- train(average_speed~., data = trainData,
         method = "lm")
summary(olsFit)
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.21236 -0.32095  0.07241  0.37806  1.32025 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)         13.9552955  0.5511496  25.320  < 2e-16 ***
distance            -0.1162594  0.0368043  -3.159  0.00229 ** 
total_elevation_gain  0.0098641  0.0047045   2.097  0.03943 *  
temp                -0.0013666  0.0092437  -0.148  0.88287    
hum                 -0.0134707  0.0088119  -1.529  0.13061    
days                -0.0061009  0.0008384  -7.277 2.98e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6534 on 74 degrees of freedom
Multiple R-squared:  0.5471,	Adjusted R-squared:  0.5165 
F-statistic: 17.88 on 5 and 74 DF,  p-value: 1.387e-11
```


trainPreds. vs Obs.
x
9
10
11
12
13
14


testPreds. vs Obs.
x
11.0
11.5
12.0
12.5
13.0
13.5
11
12
13

Results:

```
> reigelMSE <- c(s,t); reigelMSE
0.5633143 0.5858732
> c(olsTrainMSE, olsTestMSE)
0.4021139 0.4592827
```

Results:

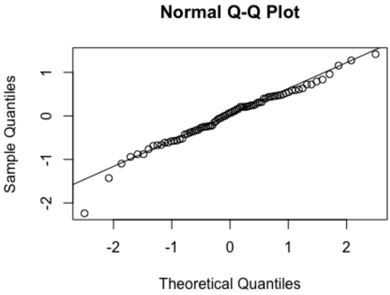
```
> reigelMSE <- c(s,t); reigelMSE
0.5633143 0.5858732
> c(olsTrainMSE, olsTestMSE)
0.4021139 0.4592827
```

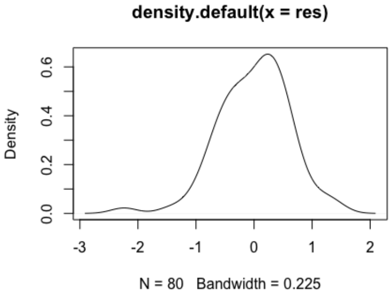
Model	Training samples	Test samples	Training MSE (error in minutes)	Test MSE (error in minutes)
Basic linear regression	358	89	0.6403	0.5517
Locally weighted regression	358	89	0.4624	0.5521
Ridge regression	358	89	0.4615	0.5559
Lasso regression	358	89	0.5722	0.5682

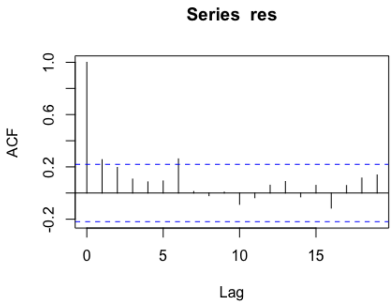
Figure: Tiffany Jin's results.

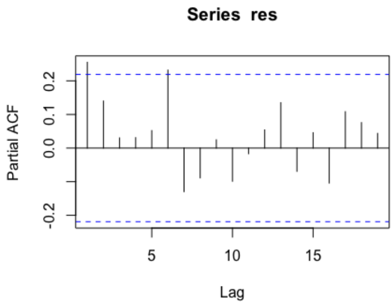
```
> var <- varImp(olsFit, scale = FALSE)
> var
lm variable importance

  Overall
days                    8.1433
distance                2.2196
hum                     2.1716
total_elevation_gain    2.0728
temp                    0.2918
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.21236 -0.32095  0.07241  0.37806  1.32025 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)          13.9552955  0.5511496  25.320  < 2e-16 ***
distance             -0.1162594  0.0368043  -3.159  0.00229 ** 
total_elevation_gain  0.0098641  0.0047045   2.097  0.03943 *  
temp                 -0.0013666  0.0092437  -0.148  0.88287    
hum                  -0.0134707  0.0088119  -1.529  0.13061    
days                 -0.0061009  0.0008384  -7.277 2.98e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6534 on 74 degrees of freedom
Multiple R-squared:  0.5471,	Adjusted R-squared:  0.5165 
F-statistic: 17.88 on 5 and 74 DF,  p-value: 1.387e-11
```


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
-2
-1
0
1
2


density.default(x = res)
Density
0.0
0.2
0.4
0.6
-3
-2
-1
0
1
2
N = 80 Bandwidth = 0.225


Series res
ACF
1.0
0.6
0.2
-0.2
0
5
10
15
Lag


Series res
Partial ACF
0.2
0.1
0.0
-0.2
5
10
15
Lag

```
> resModel = lm(res[-length(res)] ~ res[-1])
> summary(resModel)

Call:
lm(formula = res[-length(res)] ~ res[-1])

Residuals:
    Min      1Q  Median      3Q     Max 
-2.1772 -0.2717  0.0298  0.3196  1.5998 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)  
(Intercept) -0.008972   0.072344  -0.124   0.9016  
res[-1]      0.257506   0.109403   2.354   0.0211 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.643 on 77 degrees of freedom
Multiple R-squared:  0.06712,	Adjusted R-squared:  0.055 
F-statistic:  5.54 on 1 and 77 DF,  p-value: 0.02114

> # we can also use the Durbin-Watson Test
> # dwtest(olsFit)
> dwtest(lm(average_speed~., data = trainData))

	Durbin-Watson test

data:  lm(average_speed ~ ., data = trainData)
DW = 1.4594, p-value = 0.002963
alternative hypothesis: true autocorrelation is greater than 0
```

ARMA Time Series Models

ARMA Time Series Models

Y_t is time dependent, i.e. the values of Y_t are affected by past values.

ARMA Time Series Models

Y_t is time dependent, i.e. the values of Y_t are affected by past values.

1. Autoregressive (AR) models are models in which the value of a variable in one period is related to its values in previous periods.

ARMA Time Series Models

Y_t is time dependent, i.e. the values of Y_t are affected by past values.

1. Autoregressive (AR) models are models in which the value of a variable in one period is related to its values in previous periods. An AP(p) model is a model with p lags

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t$$

2. Moving average (MA) models account for the possibility of a relationship between Y_t and the residuals from previous periods. An MA(q) model is a moving average model with q lags

$$Y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}$$

ARAMA Time Series Models

An ARMA(p, q) model combines p autoregressive terms and q moving average terms

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}.$$

ARAMA Time Series Models

An ARMA(p, q) model combines p autoregressive terms and q moving average terms

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}.$$

We can also add other variables θ to get

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i} + F(\theta).$$

ARAMA Time Series Models

An ARMA(p, q) model combines p autoregressive terms and q moving average terms

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}.$$

We can also add other variables θ to get

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i} + F(\theta).$$

To fit an ARMA model we need to have Y_t be stationary

ARAMA Time Series Models

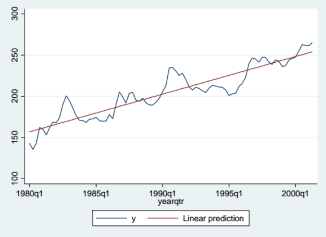
An ARMA(p, q) model combines p autoregressive terms and q moving average terms

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}.$$

We can also add other variables θ to get

$$Y_t = \mu + \sum_{i=1}^{p} \gamma_i Y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i} + F(\theta).$$

To fit an ARMA model we need to have Y_t be stationary



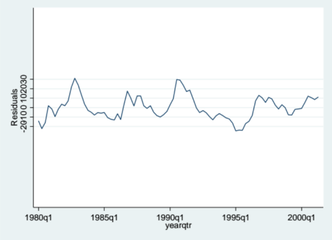


Figure: Left: Y_t. Right: $Y_t - Y_{t-1}$

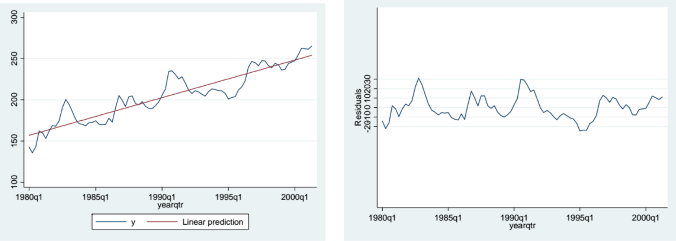


Figure: Left: Y_t. Right: $Y_t - Y_{t-1}$

An ARIMA(p, d, q) model accounts for non-stationary trends.

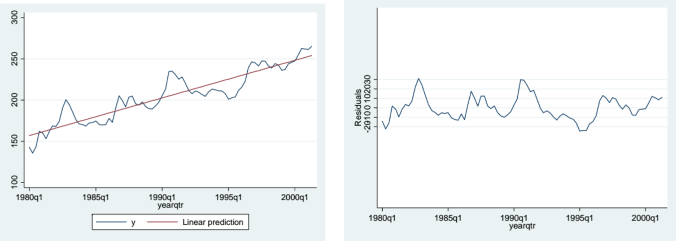


Figure: Left: Y_t. Right: $Y_t - Y_{t-1}$

An ARIMA(p, d, q) model accounts for non-stationary trends.Take d differences of Y_t.

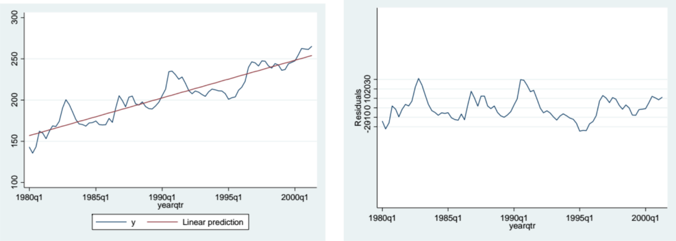
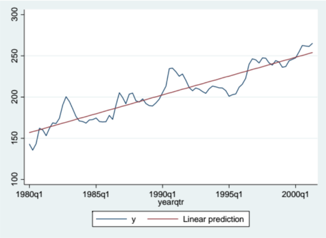


Figure: Left: Y_t. Right: $Y_t - Y_{t-1}$

An ARIMA(p, d, q) model accounts for non-stationary trends.Take d differences of Y_t. Model on $Z_t := (1 - L)^d Y_t$, where $L^i = Y_t - Y_{t-i}$ is the lag operator.



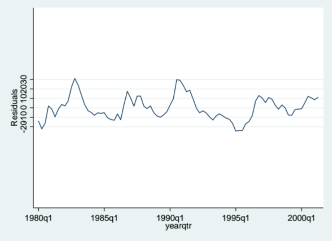


Figure: Left: Y_t. Right: $Y_t - Y_{t-1}$

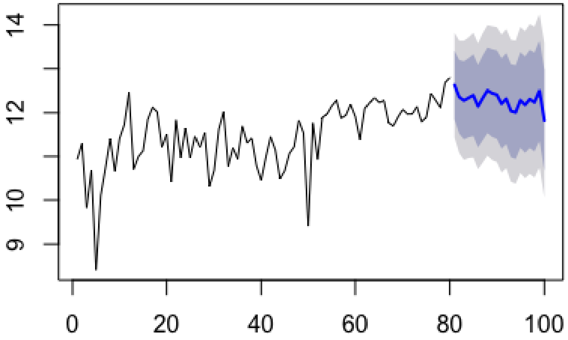
An ARIMA(p, d, q) model accounts for non-stationary trends. Take d differences of Y_t. Model on $Z_t := (1 - L)^d Y_t$, where $L^i = Y_t - Y_{t-i}$ is the lag operator. Now fit ARMA(p, q) to Z_t.

```r
library("forecast")

data <- read.csv(file = "dataFFF.csv")

trainData <- data[1:80,]
testData <- data[81:100,]

trainY <- trainData$average_speed;
testY <- testData$average_speed

# the outer regressors
trainReg <- trainData[,2:5]
testReg <- testData[,2:5]

m <- auto.arima(trainY, xreg = trainReg)
summary(m)
```

```
> summary(m)
Series: trainY
ARIMA(2,1,2)

Coefficients:
          ar1     ar2     ma1      ma2  distance
      -0.6665  0.3035  0.0855  -0.6853   -0.0425
s.e.   0.1830  0.1713  0.1418   0.1017    0.0339
      total_elevation_gain    temp      hum
                    0.0030  0.0079  -0.0026
s.e.                0.0039  0.0072   0.0076

sigma^2 estimated as 0.3354:  log likelihood=-74.99
AIC=167.98   AICc=170.23   BIC=190.47

Training set error measures:
                     ME      RMSE       MAE       MPE     MAPE
Training set 0.08231716 0.5497863 0.3934995 0.4753642 3.528439
                  MASE       ACF1
Training set 0.7769913 -0.0242225
```

```
f <- forecast(m, h = 20 ,xreg=testReg)

summary(f)

plot(f)

predictions <- f$mean

sum(residuals(m)^2)/80

RMSE(predictions, testY)^2
```

Forecasts from ARIMA(2,1,2)



```
> sum(residuals(m)^2)/80
[1] 0.308972
> RMSE(predictions, testY)^2
[1] 2.127791
```

LOOCV

1. Take $M_1, \ldots, M_k$ models.

LOOCV

1. Take $M_1, \ldots, M_k$ models. In this case, these correspond to different values of (p, d, q).

LOOCV

1. Take $M_1, \ldots, M_k$ models. In this case, these correspond to different values of (p, d, q).
2. Let $y_1, \ldots, y_t, \ldots y_N$ be the data. For each model M, fit to $y_1, \ldots, y_t$.

LOOCV

1. Take $M_1, \ldots, M_k$ models. In this case, these correspond to different values of (p, d, q).
2. Let $y_1, \ldots, y_t, \ldots y_N$ be the data. For each model M, fit to $y_1, \ldots, y_t$.Calculate the error $e_t^M := y_{t+1} - \hat{y}_{t+1}$.

LOOCV

1. Take $M_1, \ldots, M_k$ models. In this case, these correspond to different values of (p, d, q).
2. Let $y_1, \ldots, y_t, \ldots y_N$ be the data. For each model M, fit to $y_1, \ldots, y_t$.Calculate the error $e_t^M := y_{t+1} - \hat{y}_{t+1}$. Do this for $t+1, t+2, \ldots, t_{N-1}$. Estimate MSE from the errors
3. Choose model with smallest MSE estimate.

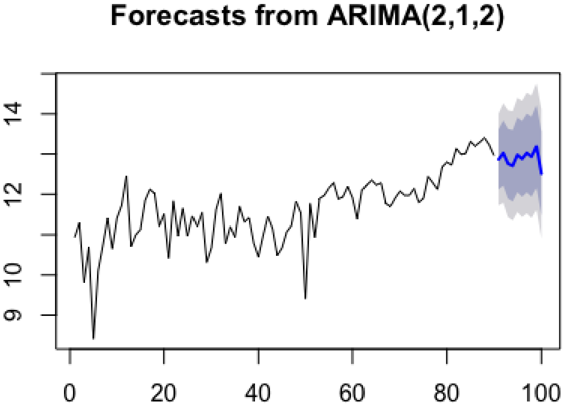
LOOCV

1. Take $M_1, \ldots, M_k$ models. In this case, these correspond to different values of (p, d, q).
2. Let $y_1, \ldots, y_t, \ldots y_N$ be the data. For each model M, fit to $y_1, \ldots, y_t$.Calculate the error $e_t^M := y_{t+1} - \hat{y}_{t+1}$. Do this for $t+1, t+2, \ldots, t_{N-1}$. Estimate MSE from the errors
3. Choose model with smallest MSE estimate.

Our models: $(1, 0, 1)$, $(2, 0, 1)$, $(2, 1, 1)$, $(2, 1, 2)$. Using 90 obs.

Winner: $(1, 0, 1)$.

Winner: $(1, 0, 1)$.



```
> predictions <- f$mean;
> #predictionsDiff <- forecast.modelDiff$mean
> sum(residuals(m)^2)/N
[1] 0.3054708
> RMSE(predictions, testY)^2
[1] 0.1609584
>
> df <- data.frame(testY, as.data.frame(f)); df
      testY Point.Forecast    Lo.80    Hi.80    Lo.95    Hi.95
 91 12.9816       12.86647 12.11985 13.61309 11.72461 14.00833
 92 13.1184       13.02471 12.21492 13.83451 11.78624 14.26318
 93 13.5936       12.76105 11.89154 13.63056 11.43124 14.09085
 94 12.4668       12.70845 11.82018 13.59673 11.34996 14.06695
 95 12.8340       12.97789 12.04889 13.90690 11.55711 14.39868
 96 12.8628       12.88023 11.93569 13.82477 11.43568 14.32478
 97 13.8384       13.02651 12.04533 14.00769 11.52592 14.52710
 98 12.7152       12.93342 11.93737 13.92948 11.41009 14.45676
 99 13.3812       13.18461 12.15439 14.21482 11.60903 14.76018
100 12.7764       12.51263 11.46792 13.55735 10.91488 14.11039
```

Final Predictions

On Sunday: 21km, elev gain $\approx 100ft$, forecast: 71 degrees, 55% hum.

Final Predictions

On Sunday: 21km, elev gain $\approx 100ft$, forecast: 71 degrees, 55% hum.

Predictor	Pace (min/km)
Riegel	5.17
OLS	5.34
ARIMA	4.73