Kriging Corn Yield Data

Peter Claussen

June 11, 2020

Introduction

We are given a data file containing yield monitor data. Our assignment is to determine if kriging can be used to impute missing data. More specifically, our data consists of harvester yield monitor data from a single field. The field is harvested in a series of *passes*. Ideally, a pass consists of one swath starting at one end of the field and ending at the opposite end.

We want to determine if yield for a missing pass can be reasonably interpolated from the adjacent corn harvest data points.

This analysis is in contrast to more a commonly used method. In [**?**, **?**] the field is analyzed as a lattice of 9.8 $\times$ 9.8 meter squares, although [**?**] argue that a 20m length is minimum for robust yield prediction, while [**?**] used 35m quadrats. The lattice may be rotated from true north to match the direction of planting. In [**?**] this reduces a set 8288 point estimates of yield to 1738 polygons. Anselin [**?**] suggest this reduces errors associated with GPS yield monitor data. Bongiovanni [**?**] present a more detailed discussion of the processes and errors associated with yield monitor data.

Kriging has been used to generate whole field visualizations [**?**]. One drawback to kriging is the imprecision in both yield measurement and within-field point location. We will consider methods for reducing this uncertainty.

Harvest monitor yield data is inherently messy. [**?**] identify six types of erroneous data, which they screen before analysis. We will try to keep as much original data as possible, and interpolate expected values for outliers.

For this exercise, load and examine the contents of our data file.

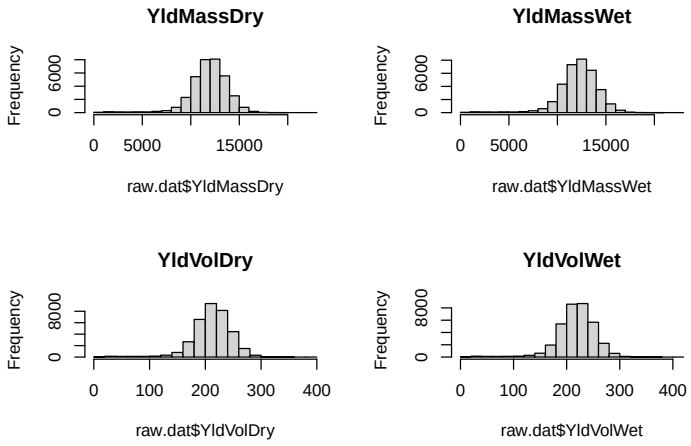
```
> raw.dat <- read.csv("Swest_Corn2013-Clean.csv", header = TRUE)
> summary(raw.dat)
```

```
       ID             OID          Longitude        Latitude
 Min.   :    1   Min.   :    1   Min.   :-96.9   Min.   :43.23
 1st Qu.: 2905   1st Qu.: 8715   1st Qu.:-96.9   1st Qu.:43.23
 Median : 7054   Median :17428   Median :-96.9   Median :43.23
 Mean   : 8303   Mean   :17428   Mean   :-96.9   Mean   :43.23
 3rd Qu.:12993   3rd Qu.:26142   3rd Qu.:-96.9   3rd Qu.:43.23
 Max.   :21707   Max.   :34856   Max.   :-96.9   Max.   :43.24
    Field              Dataset          Product              Swath
 Length:34856       Min.   :-113892   Length:34856       Min.   : 4.99
 Class :character   1st Qu.:-113892   Class :character   1st Qu.:14.99
 Mode  :character   Median :-113892   Mode  :character   Median :14.99
                    Mean   :-113892                      Mean   :14.92
                    3rd Qu.:-113892                      3rd Qu.:14.99
                    Max.   :-113892                      Max.   :14.99
    Distance        Duration     Track          Elevation
 Min.   :0.820   Min.   :1   Min.   :  0.0   Min.   :1241
 1st Qu.:4.530   1st Qu.:1   1st Qu.:  0.0   1st Qu.:1252
 Median :4.950   Median :1   Median :148.3   Median :1255
```

```
Mean   :4.875   Mean   :1   Mean   :100.6   Mean   :1255
3rd Qu.:5.250   3rd Qu.:1   3rd Qu.:180.0   3rd Qu.:1258
Max.   :7.190   Max.   :1   Max.   :359.5   Max.   :1266
 Areacount             Time              Yoffset     PassNum
Length:34856       Length:34856       Min.   :0   Min.   : 1.00
Class :character   Class :character   1st Qu.:0   1st Qu.:12.00
Mode  :character   Mode  :character   Median :0   Median :31.00
                                      Mean   :0   Mean   :34.24
                                      3rd Qu.:0   3rd Qu.:55.00
                                      Max.   :0   Max.   :73.00
   CropFlw         Moisture       YldMassDry        YldMassWet
Min.   : 0.40   Min.   : 0.00   Min.   :  299.9   Min.   :  307.1
1st Qu.:18.10   1st Qu.:16.87   1st Qu.:10866.3   1st Qu.:11173.1
Median :20.60   Median :17.25   Median :11974.6   Median :12305.4
Mean   :20.34   Mean   :17.28   Mean   :11837.6   Mean   :12165.9
3rd Qu.:23.00   3rd Qu.:17.63   3rd Qu.:13059.2   3rd Qu.:13424.0
Max.   :35.60   Max.   :24.00   Max.   :22241.7   Max.   :22736.6
  YldVolDry        YldVolWet          Speed            Prod
Min.   :  5.35   Min.   :  5.48   Min.   :0.560   Min.   :1.020
1st Qu.:194.04   1st Qu.:199.52   1st Qu.:3.090   1st Qu.:5.570
Median :213.83   Median :219.74   Median :3.380   Median :6.140
Mean   :211.39   Mean   :217.25   Mean   :3.324   Mean   :6.005
3rd Qu.:233.20   3rd Qu.:239.71   3rd Qu.:3.580   3rd Qu.:6.460
Max.   :397.17   Max.   :406.01   Max.   :4.900   Max.   :8.900
   CropFlwV           Date
Min.   :  25.71   Length:34856
1st Qu.:1163.57   Class :character
Median :1324.28   Mode  :character
Mean   :1307.62
3rd Qu.:1478.57
Max.   :2288.57
```

Our primary response will be yield, reported in dry volume. How is this distributed?

```
> par(mfrow=c(2,2))
> hist(raw.dat$YldMassDry,main="YldMassDry")
> hist(raw.dat$YldMassWet,main="YldMassWet")
> hist(raw.dat$YldVolDry,main="YldVolDry")
> hist(raw.dat$YldVolWet,main="YldVolWet")
> par(mfrow=c(1,1))
```

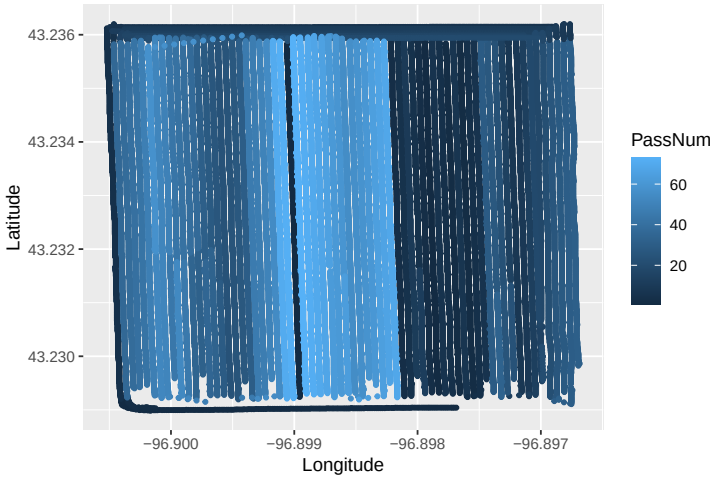


We could use any of the four yield measures, they appear to be roughly equivalent.

Cleaning the data

Our task is to impute yield data for single passes in the field. Figure **??** shows the original data colored by pass number. This gives us an idea about the order which data were collected. We note that passes were not made in an orderly progression across the field. This will complicate our analysis later if we want to determine relationships among neighboring passes.

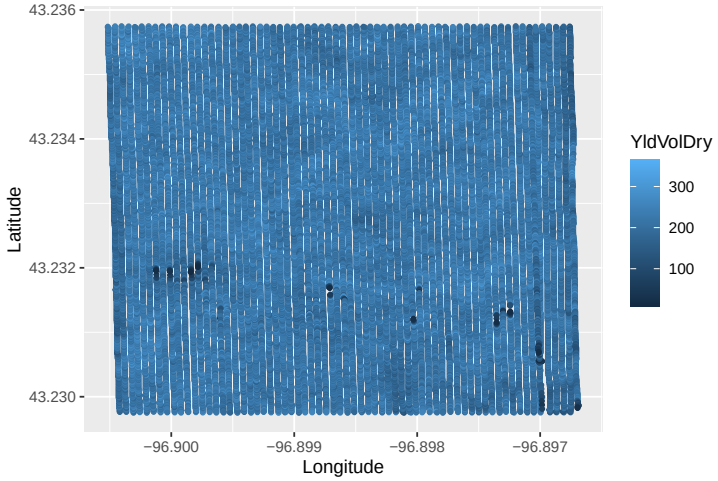
```
> ggplot(raw.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = PassNum),size = 1)
```



We want to discard the endrows. They generally won't be representative of the field in general, since during planting the soil is passed over more frequently.

Examining **??**, we can assign upper and lower bounds for analysis. We'll refer to this as the trimmed data set.

```
> northBorder <- 43.23575
> southBorder <- 43.22975
> trimmed.dat <- subset(raw.dat,raw.dat$Latitude>=southBorder)
> trimmed.dat <- subset(trimmed.dat,trimmed.dat$Latitude<=northBorder)

> ggplot(trimmed.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = YldVolDry),size =
```



Subset data set

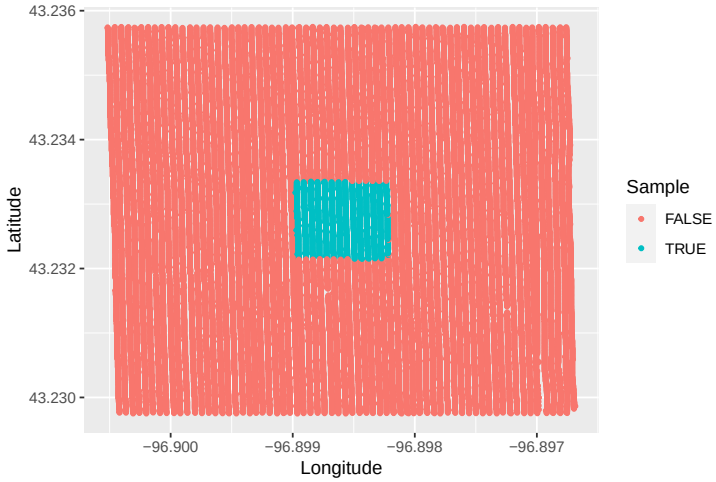
The `kriging` function slows my system greatly, so we need to start with a subset of the full data set to explore variagrams for this type of data.

We'll start with the middle 1/25 of the trimmed field. I'm labelling the subset to simplify plotting. We'll refer to this data set as the sample data set.

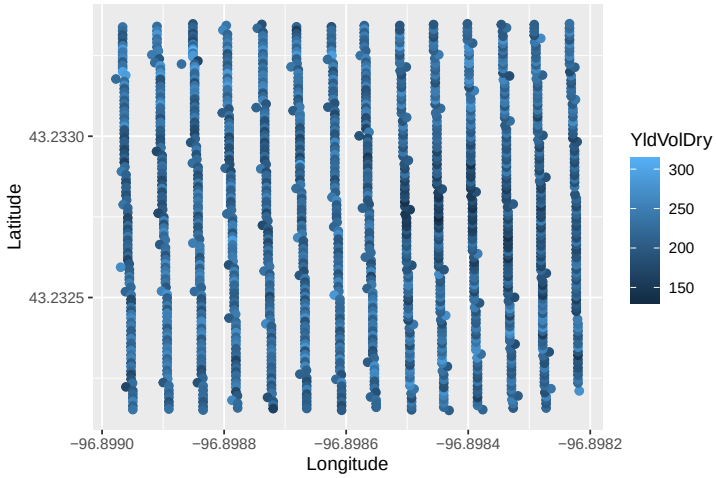
```
> min.lat <- min(trimmed.dat$Latitude)
> max.lat <- max(trimmed.dat$Latitude)
> min.lon <- min(trimmed.dat$Longitude)
> max.lon <- max(trimmed.dat$Longitude)
> lat.rng <- max.lat-min.lat
> lon.rng <- max.lon-min.lon
> trimmed.dat$Sample <- trimmed.dat$Latitude>(min.lat + 2*(lat.rng/5)) &
+                       trimmed.dat$Latitude<(max.lat - 2*(lat.rng/5)) &
+                       trimmed.dat$Longitude>(min.lon + 2*(lon.rng/5)) &
+                       trimmed.dat$Longitude<(max.lon - 2*(lon.rng/5))
> sample.dat <- subset(trimmed.dat,trimmed.dat$Sample)
```

Examine the trimmed data and sample.

```
> ggplot(trimmed.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = Sample),size = 1)
```




```
> ggplot(sample.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = YldVolDry),size = 2
```



`PassNum` tells us the order which the combine moves up or down the field. This is not in longitudinal order, so we need to convert. First, we average the longitude of each pass, the rank values to get a spatial order. We add these to the sample data set. Note that I'm creating factors for pass number and order so we can visualize these. We'll use the pass order number in Figure **??** to remove a single pass from the middle of the sample for interpolation.

```
> sample.dat$Pass <- as.factor(sample.dat$PassNum)
> pass.long <- tapply(sample.dat$Longitude,list(sample.dat$Pass),mean)
> sample.dat$Order <- rank(pass.long)[sample.dat$Pass]
> sample.dat$PassOrder <- as.factor(sample.dat$Order)
```

```
> ggplot(sample.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = PassOrder),size = 2
```

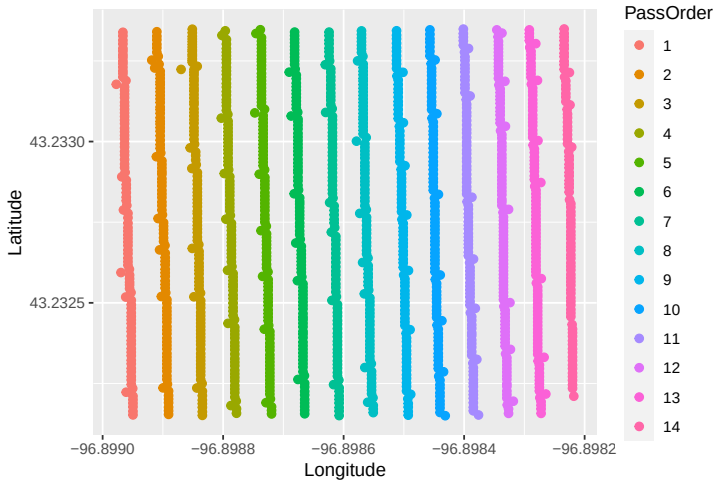
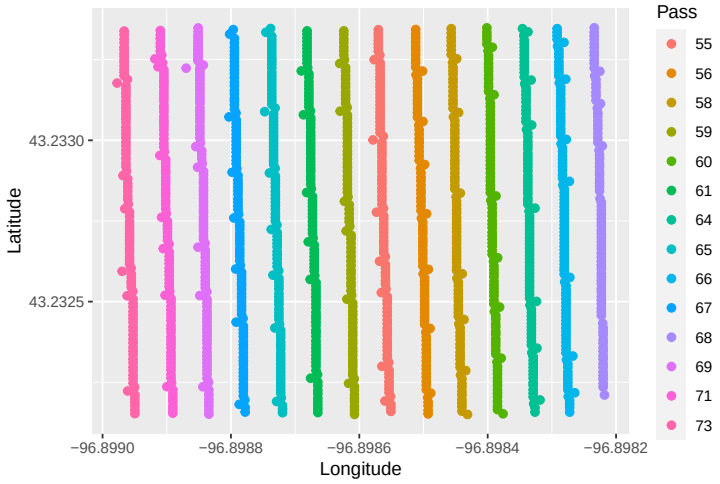


Figure 1: Sample region, colored by Order (east to west)

```
> ggplot(sample.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = Pass),size = 2)
```



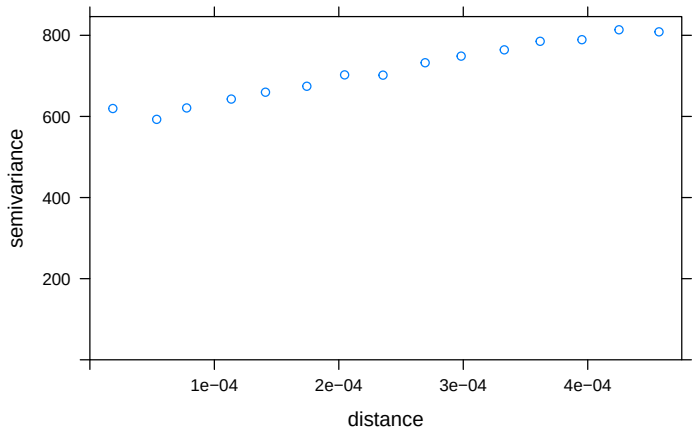
Kriging

There are several options for kriging using R. I prefer the `gstat` library because it doesn't require creating spatial objects; it can be used with a simple data frame. But we will compare a couple other packages.

gstat

Generating a variogram with `gstat` is straightforward; response and spatial coordinates can be specified with standard R formula, no need to create spatial instances. I have some concern that spatial objects are memory inefficient; but I've not yet taken the time to confirm that suspicion. Maybe a different exercise.

```
> library(gstat)
> sample.var <- variogram(YldVolDry~1,
+                         locations=~Longitude+Latitude,
+                         data=sample.dat)
> plot(sample.var)
```



We next fit a model the variogram, using `fit.variogram`. We give an initial guess using the `vgm` function. I tried several combinations, but none work well; we get a message, `Warning: singular model in variogram fit`.

```
> sample.vgm <- fit.variogram(sample.var, vgm(800,"Exp",0.0004,500))
```

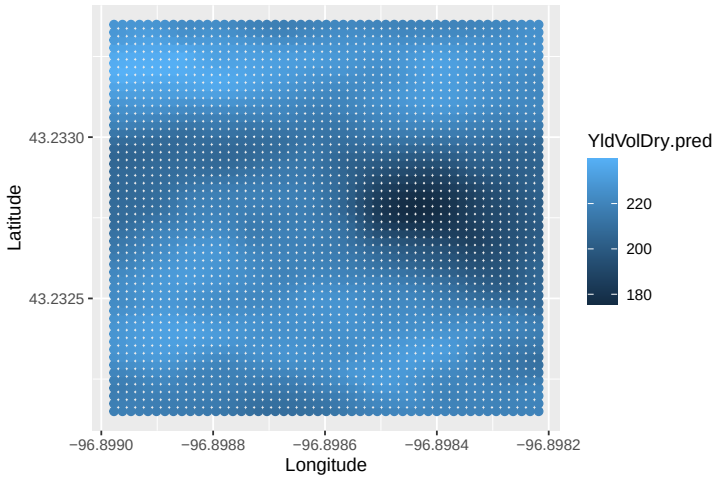
We don't get a warning with the larger data set, so for now we'll just proceed as if the fit were satisfactory. The next step is to define grid for interpolation - these will be points where we don't have measured yields.

```
> smin.lat <- min(sample.dat$Latitude)
> smax.lat <- max(sample.dat$Latitude)
> smin.lon <- min(sample.dat$Longitude)
```

```
> smax.lon <- max(sample.dat$Longitude)
> slat.rng <- smax.lat-smin.lat
> slon.rng <- smax.lon-smin.lon
> sample.grd <- expand.grid(Longitude=seq(from=smin.lon, to=smax.lon, by=slon.rng/50),
+                           Latitude=seq(from=smin.lat, to=smax.lat, by=slat.rng/50))
```

The actual kriging step is to interpolate yield values for our arbitrary points in `sample.grid`, base on the original observations and using the parameters estimated from our variogram.

```
> sample.krig <- krige(id="YldVolDry",
+                      formula=YldVolDry~1,
+                      data = sample.dat,
+                      newdata = sample.grd,
+                      model = sample.vgm,
+   locations=~Longitude+Latitude)

[using ordinary kriging]

> ggplot(sample.krig, aes(Longitude, Latitude)) + geom_point(aes(colour = YldVolDry.pred),si
```



`gstat` allows for directional variograms. This is useful to examine the isotropy of the data. I've included 0 and 90 to estimate variograms in the main directions, plus slight offset angles to allow for a slight deviation from true north in planting direction.

Note that there is a clear difference in directions in Figure **??**, suggesting anisotropy in the data. There also appears to be a periodic effect in the north-south (90) direction. This may be a result of the mixing as the harvester moves along the pass.

```
> sample90.var <- variogram(YldVolDry~1,
+                           locations=~Longitude+Latitude,
```

```
> sample90.var <- variogram(YldVolDry~1,
+                           locations=~Longitude+Latitude,
+                           alpha=c(0,30,60,90),
+                           data=sample.dat)
> plot(sample90.var)
```

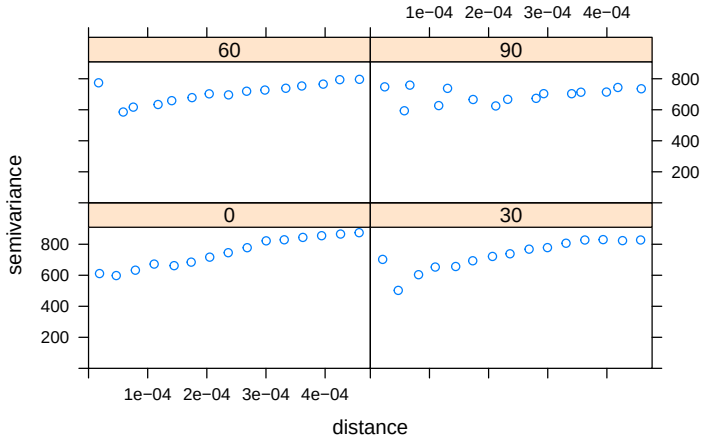
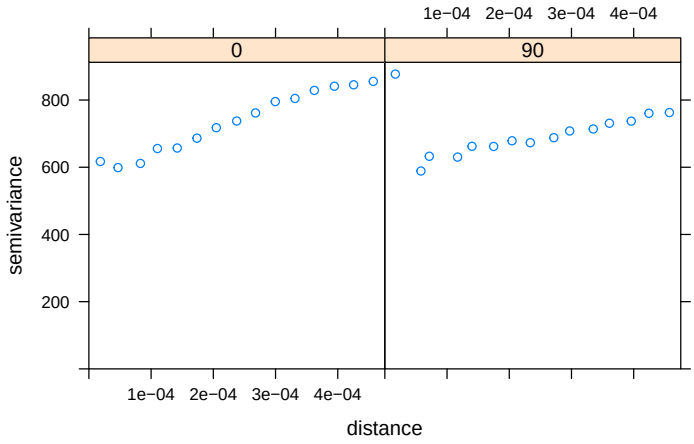


Figure 2: Sample one-direction variograms

```
+                                   alpha=c(0,90),
+                                   data=sample.dat)
> plot(sample90.var)
```

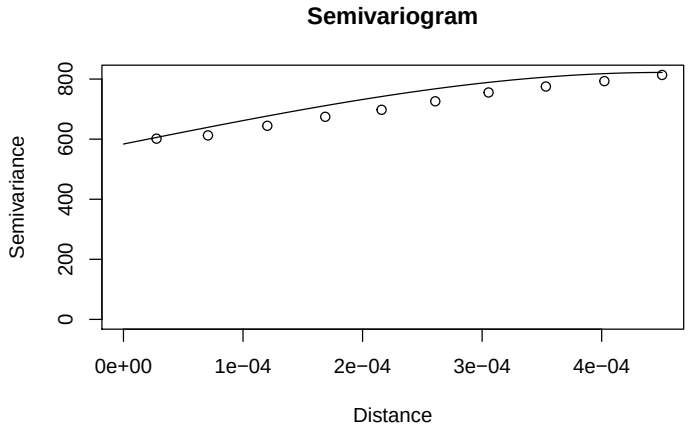


Note that including more angles alters the variograms.

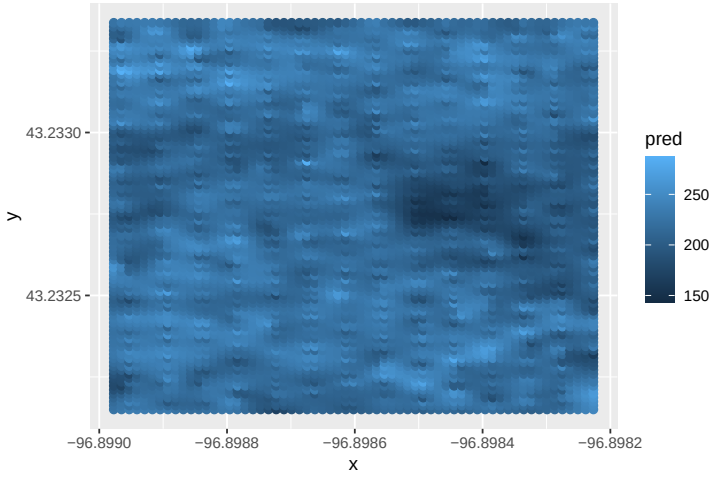
kriging

The `kriging` package provide a kriging function, but this is extremely slow. It produces a kriged data set in one pass, which does not allow for control of the points to be interpolated. With both `gdata` and `geoR`, the kriging over the original data area is the slow step. Later, we'll be interpolating for a select pass in our data, which will be a smaller area and should be faster. `kriging` doesn't allow for this; the slowest step is performed by default.

```
> library(kriging)
> sample.kriging <- kriging(sample.dat$Longitude, sample.dat$Latitude, sample.dat$YldVolDry)
> plot(sample.kriging)
```




```
> ggplot(sample.kriging$map, aes(x, y)) +
+    geom_point(aes(colour = pred),size = 2)
```



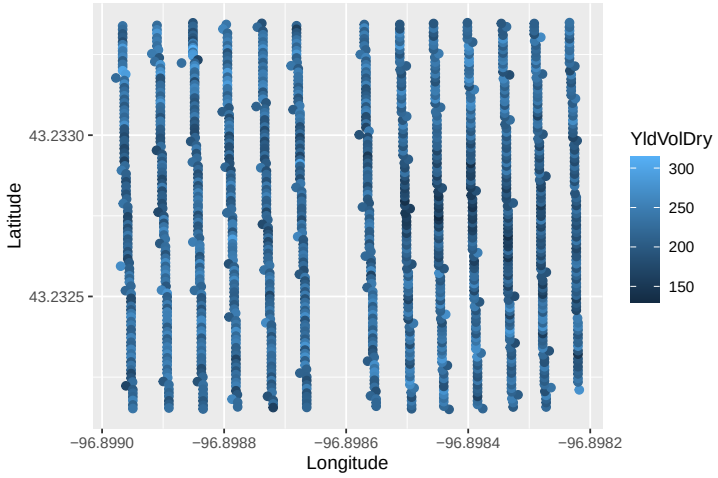
It does appear that with the default settings, `kriging` produces a finer interpolated field.

Interpolating a missing pass

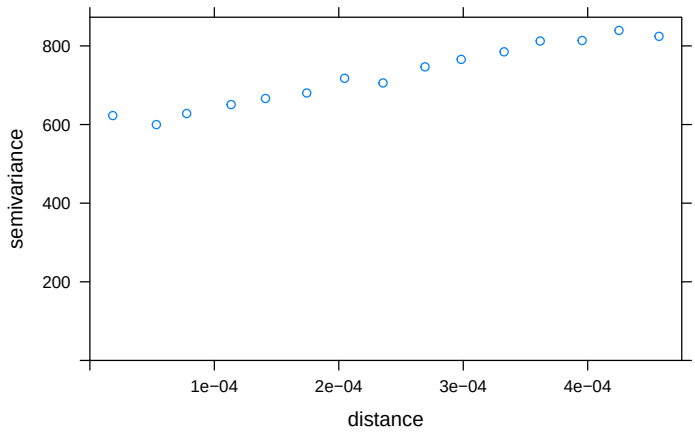
Next, we consider how well kriging can be used to interpolate a single missing pass. We consider this in the context of testing the effectiveness of site-specific management, and that we've applied some treatment to a single pass. We wish to remove the pass from analysis, estimate an expected yield for that pass and compare to the measured yield. See [**?**] for a review of other experimental considerations.

First, we consider the sample data. We want to drop the pass closest to the center. We'll drop this pass from the sample data set, then perform kriging on the dropped pass.

```
> min.order <- min(sample.dat$Order)
> max.order <- max(sample.dat$Order)
> drop.order <- floor(min.order + (max.order-min.order)/2)
> sample.drop1.dat <- subset(sample.dat,sample.dat$Order!=drop.order)
> sample.dropped.dat <- subset(sample.dat,sample.dat$Order==drop.order)

> ggplot(sample.drop1.dat, aes(Longitude, Latitude)) +
+    geom_point(aes(colour = YldVolDry),size = 2)
```




```
> sample.dropped.var <- variogram(sample.drop1.dat$YldVolDry~1,
+                                 locations=~Longitude+Latitude,
+                                 data=sample.drop1.dat)
> plot(sample.dropped.var)
```



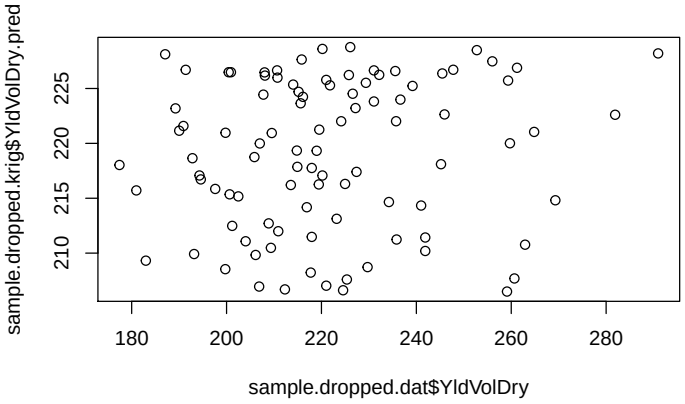
Now we krige a missing pass, interpolated at the points in the dropped data set using measured values from the sample (minus one pass) data set.

```
> sample.dropped.vgm <- fit.variogram(sample.dropped.var,
+                                     vgm(800,"Exp",0.0004,500))
> sample.dropped.krig <- krige(id="YldVolDry", formula=YldVolDry~1,
+                              data = sample.drop1.dat,
+                              newdata = sample.dropped.dat,
+                              model = sample.dropped.vgm,
+                              locations=~Longitude+Latitude)
```

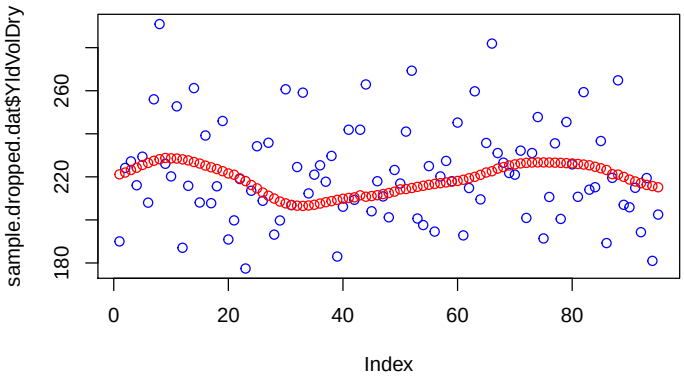
```
[using ordinary kriging]
```

Compare the imputed and actual values.

```
> plot(sample.dropped.dat$YldVolDry, sample.dropped.krig$YldVolDry.pred)
```

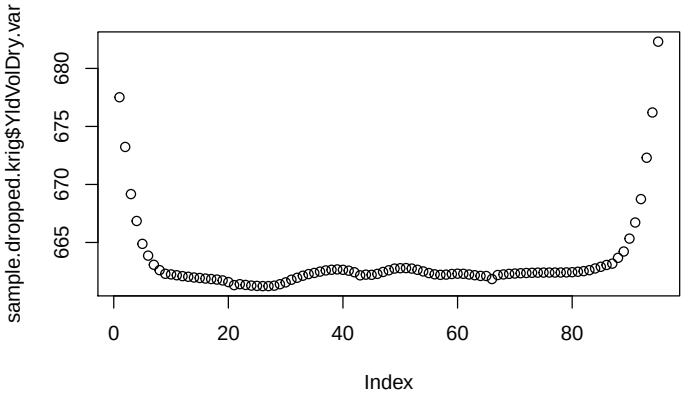


```
> plot(sample.dropped.dat$YldVolDry,col="blue")
> points(sample.dropped.krig$YldVolDry.pred,col="red")
```



We are also provided a variance on the predicted value; as we might expect, variance is higher at the ends.

```
> plot(sample.dropped.krig$YldVolDry.var)
```



We might be tempted to simply compare values predicted by kriging and raw harvest values. If we do this, we would see that kriged values do not accurately predict actual harvest.

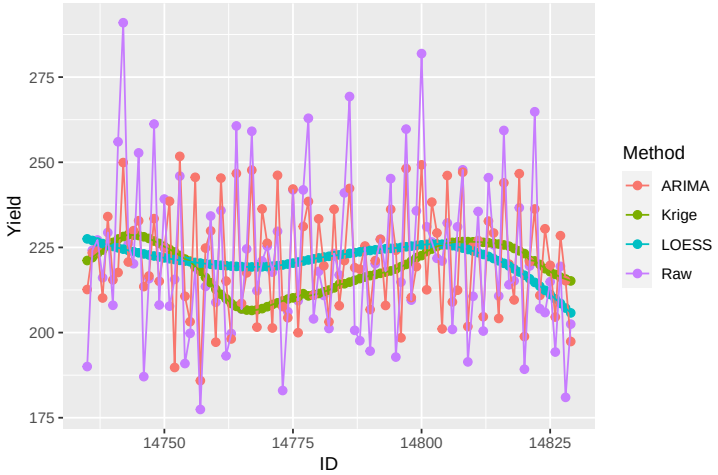
```
> sample.mean <- mean(sample.dropped.dat$YldVolDry)
> df <- length(sample.dropped.dat$YldVolDry)-1
> total.sums <- sum((sample.dropped.dat$YldVolDry-sample.mean )^2)
> model.sums <- sum((sample.dropped.krig$YldVolDry.pred-sample.mean)^2)
```

```
> resid.sums <- total.sums-model.sums
> 1 - resid.sums/total.sums

[1] 0.1010406
```

However, we should note that kriging is a interpolating process, and tends to produce inherently "smoothed" values. We should compare instead kriging values with other interpolations. For example, compare the values interpolated from the two-dimensions space outside the dropped pass with values interpolated from within the dropped pass only. We can't simply repeat kriging; this will produce our original values. Instead, we'll smooth using `loess` and `arima`

```
> library(forecast)
> sample.dropped.arima <- auto.arima(sample.dropped.dat$YldVolDry)
> sample.dropped.arima

Series: sample.dropped.dat$YldVolDry
ARIMA(3,0,2) with non-zero mean

Coefficients:
          ar1      ar2     ar3     ma1     ma2      mean
      -0.9745  -0.5510  0.3056  1.4368  0.8572  221.7122
s.e.   0.1116   0.1398  0.1126  0.0829  0.0592    2.3838

sigma^2 estimated as 263.9:  log likelihood=-398.46
AIC=810.91   AICc=812.2   BIC=828.79

> sample.dropped.loess <- loess(YldVolDry ~ ID,
+                               data=sample.dropped.dat)
> sample.dropped.loess

Call:
loess(formula = YldVolDry ~ ID, data = sample.dropped.dat)

Number of Observations: 95
Equivalent Number of Parameters: 4.37
Residual Standard Error: 22.91

> points <- length(sample.dropped.dat$YldVolDry)
> sample.dropped.plot <- data.frame(
+    Yield = c(sample.dropped.dat$YldVolDry,
+              sample.dropped.krig$YldVolDry.pred,
+              sample.dropped.dat$YldVolDr - sample.dropped.arima$residuals,
+              predict(sample.dropped.loess)
+    ),
+    Method = c(rep("Raw",points),
+               rep("Krige",points),
```

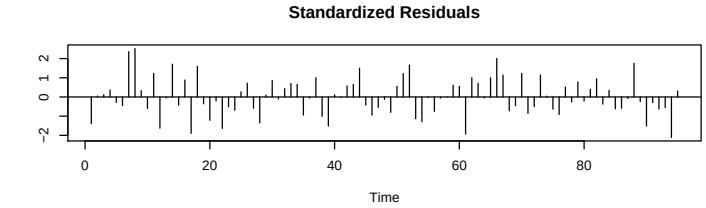
```
+                rep("ARIMA",points),
+                rep("LOESS",points)
+    ),
+    ID = rep(sample.dropped.dat$ID,4)
+ )
> sample.dropped.plot$Method <- as.factor(sample.dropped.plot$Method)

> ggplot(sample.dropped.plot, aes(ID, Yield)) +
+       geom_point(aes(colour = Method),size = 2) +
+       geom_line(aes(group=Method,color=Method))
```

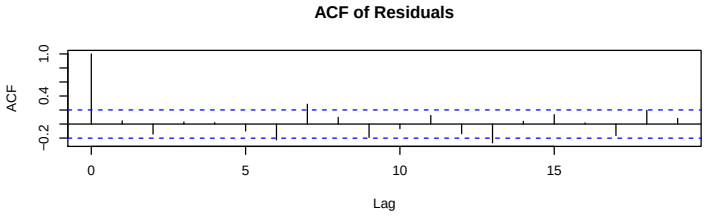


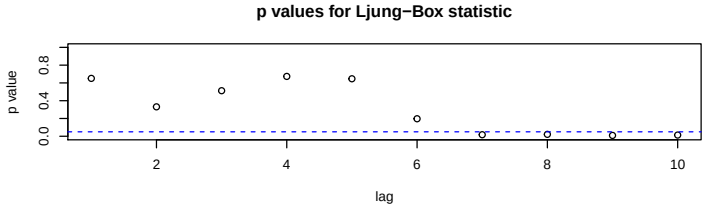
The `loess` smoothing captures the general trend of the original yield while removing much of the noise. We'll examine this further with the larger data set.

The `arima` smoothed data is still very noisy. The ARIMA process can be expected to remove time based autocorrelation with some kind of frequency component; I've included this mostly to see if there is a periodic mixing in yield data.

```
> tsdiag(sample.dropped.arima)
```


Standardized Residuals
−2
0
1
2
0
20
40
60
80
Time


ACF of Residuals
ACF
−0.2
0.4
1.0
0
5
10
15
Lag

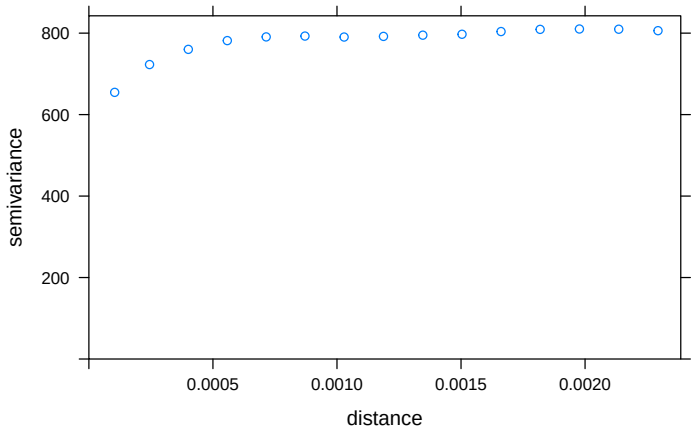

p values for Ljung−Box statistic
p value
0.0
0.4
0.8
2
4
6
8
10
lag

Full Data

Next, consider the variogram of the full data set. Remember that there were some problems with the sample data - the variograms do not appear to be exponential.

gstat

```
> trimmed.var <- variogram(trimmed.dat$YldVolDry~1,
+                          locations=~Longitude+Latitude,
+                          data=trimmed.dat)
> plot(trimmed.var)
```

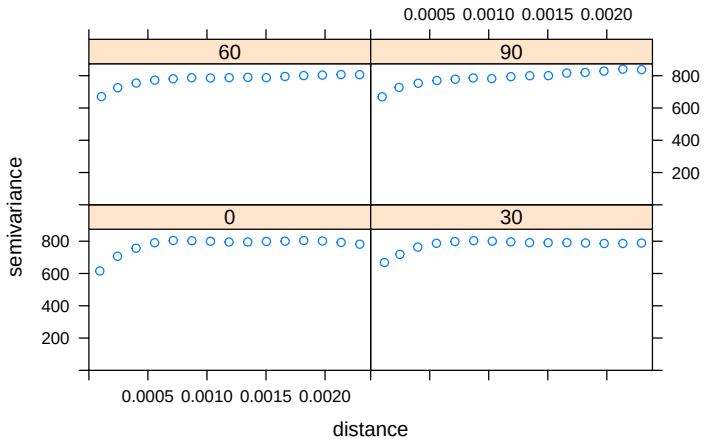


The variogram for the trimmed data set looks more like we'd expect, and the variogram fits well to an exponential function. We get consistent results with different starting values

```
> trimmed.vgm <- fit.variogram(trimmed.var, vgm(800,"Exp",0.0004,500))
> trimmed.vgm

  model    psill        range
1   Nug 569.0225 0.0000000000
2   Exp 230.4380 0.0002236035

> fit.variogram(trimmed.var, vgm(100,"Exp",0.0001,100))

  model    psill        range
1   Nug 569.0226 0.0000000000
2   Exp 230.4379 0.0002236037
```

Again, we repeat the directional variograms.

```
> trimmed90.var <- variogram(trimmed.dat$YldVolDry~1,
+                            locations=~Longitude+Latitude,
+                            alpha=c(0,30,60,90),
+                            data=trimmed.dat)
> plot(trimmed90.var)
```



Again, there are differences by direction, with the relative nugget effect greater at right angles to the main axis of the pass.

geoR

We similarly repeat the `geoR` analysis with the trimmed data. The first time I ran this, my computer froze several times. So, I'll skip this part

```
trimmed.gdat <- NA
trimmed.gvar <- NA
trimmed0.gvar <- NA
trimmed90.gvar <- NA
if(!file.exists("trimmed.gdat.Rda")) {
   trimmed.gdat <- as.geodata(trimmed.dat, coords.col = 3:4, data.col = 21)
   save(trimmed.gdat,file="trimmed.gdat.Rda")
} else {
   load("trimmed.gdat.Rda")
}

if(!file.exists("gvars.Rda")) {
   trimmed.gvar <- variog(trimmed.gdat)
   trimmed0.gvar <- variog(trimmed.gdat,direction=0)
   trimmed90.gvar <- variog(trimmed.gdat,direction=pi/2)
   gvars <- list(trimmed.gvar,trimmed0.gvar,trimmed90.gvar)
   save(gvars,file="gvars.Rda")
} else {
   load("gvars.Rda")
   trimmed.gvar <- gvars[[1]]
   trimmed0.gvar <- gvars[[2]]
   trimmed90.gvar <- gvars[[3]]
}

plot(trimmed.gvar)
plot(trimmed0.gvar)
plot(trimmed90.gvar)
trimmed.fit <- variofit(trimmed.gvar, cov.model="exp",
                        ini.cov.pars=c(800, 400),
                        fix.nugget=FALSE, nugget=500)
```

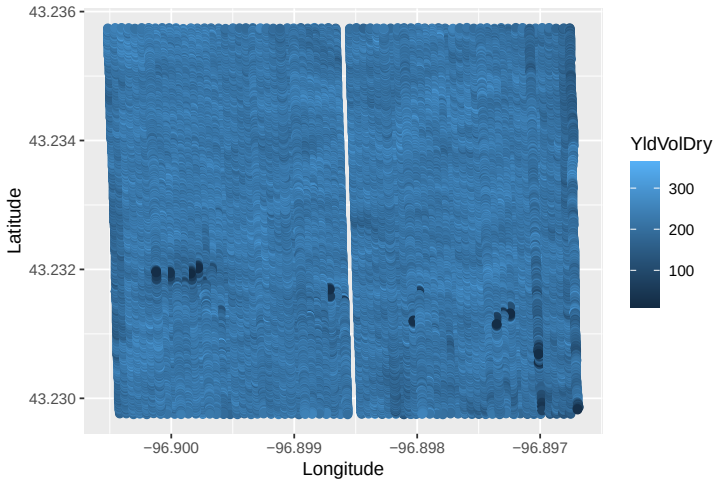
Interpolating a missing pass

Repeat for the full data set, dropping one pass

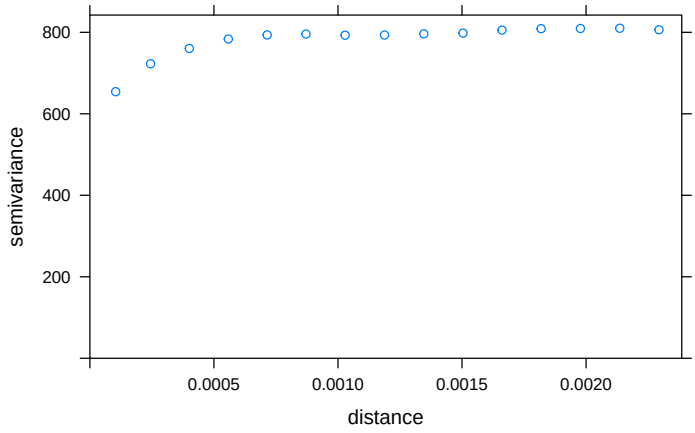
```
> dropped.pass <- sample.drop1.dat$PassNum[1]
> dropped.pass

[1] 55

> trimmed.drop1.dat <- subset(trimmed.dat,trimmed.dat$PassNum!=dropped.pass)
> trimmed.dropped.dat <- subset(trimmed.dat,trimmed.dat$PassNum==dropped.pass)
```

Examine the missing pass. This is a full pass from the middle of the field.

```
> ggplot(trimmed.drop1.dat, aes(Longitude, Latitude)) +
+     geom_point(aes(colour = YldVolDry),size = 2)
```




```
> trimmed.dropped.var <- variogram(trimmed.drop1.dat$YldVolDry~1, locations=~Longitude+Latit
> plot(trimmed.dropped.var)
```



Kriging the excluded pass from the whole field data is very slow - on my machine, this step took about 2 hours. I'm caching the results.

```
> trimmed.dropped.vgm <- fit.variogram(trimmed.dropped.var,
+                                      vgm(800,"Exp",0.0004,500))
> if(!file.exists("trimmed.dropped.krig.Rda")) {
+    trimmed.dropped.krig <- krige(id="YldVolDry", formula=YldVolDry~1,
```

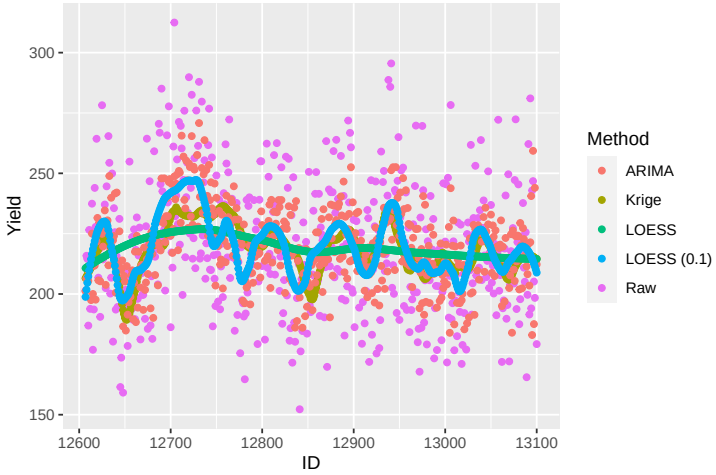
```
+                                    data = trimmed.drop1.dat,
+                                    newdata = trimmed.dropped.dat,
+                                    model = trimmed.dropped.vgm,
+                                    locations=~Longitude+Latitude)
+    save(trimmed.dropped.krig,file="trimmed.dropped.krig.Rda")
+ } else {
+    load("trimmed.dropped.krig.Rda")
+ }
```

```
[using ordinary kriging]
```

Since this is larger data set, we'll use both the default loess span and a reduced span.

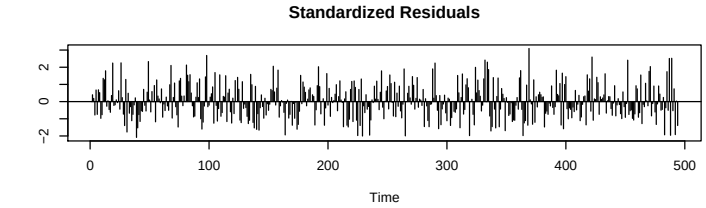
```
> trimmed.dropped.arima <- auto.arima(trimmed.dropped.dat$YldVolDry)
> trimmed.dropped.arima

Series: trimmed.dropped.dat$YldVolDry
ARIMA(4,1,2)

Coefficients:
          ar1      ar2     ar3      ar4      ma1      ma2
      -0.2685  -0.1278  0.2014  -0.2795  -0.5318  -0.2684
s.e.   0.0837   0.0685  0.0640   0.0593   0.0812   0.0785

sigma^2 estimated as 564.6:  log likelihood=-2259.46
AIC=4532.93   AICc=4533.16   BIC=4562.33

> trimmed.dropped.loess <- loess(YldVolDry ~ ID, data=trimmed.dropped.dat)
> trimmed.dropped.loess

Call:
loess(formula = YldVolDry ~ ID, data = trimmed.dropped.dat)

Number of Observations: 494
Equivalent Number of Parameters: 4.34
Residual Standard Error: 26.74

> trimmed.dropped.loess10 <- loess(YldVolDry ~ ID, data=trimmed.dropped.dat,span=0.1)
> trimmed.dropped.loess10

Call:
loess(formula = YldVolDry ~ ID, data = trimmed.dropped.dat, span = 0.1)

Number of Observations: 494
Equivalent Number of Parameters: 29.34
Residual Standard Error: 25.04
```

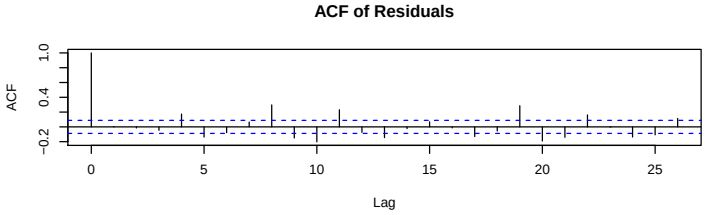
```
> points <- length(trimmed.dropped.dat$YldVolDry)
> trimmed.dropped.plot <- data.frame(
+    Yield = c(trimmed.dropped.dat$YldVolDry,
+              trimmed.dropped.krig$YldVolDry.pred,
+              trimmed.dropped.dat$YldVolDr - trimmed.dropped.arima$residuals,
+              predict(trimmed.dropped.loess),
+              predict(trimmed.dropped.loess10)
+    ),
+    Method = c(rep("Raw",points),
+                 rep("Krige",points),
+                 rep("ARIMA",points),
+                 rep("LOESS",points),
+                 rep("LOESS (0.1)",points)
+    ),
+    ID = rep(trimmed.dropped.dat$ID,5)
+ )
> trimmed.dropped.plot$Method <- as.factor(trimmed.dropped.plot$Method)

> ggplot(trimmed.dropped.plot, aes(ID, Yield)) +
+   geom_point(aes(colour = Method),size = 1.5)
```

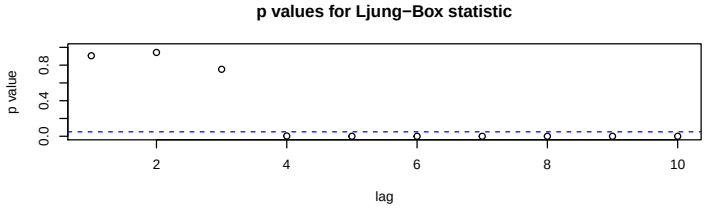


Note the difference in loess smooth with different spans. We may consider how to determine an appropriate span based on the 2D variogram. What span corresponds to 95% of the variogram sill? I've also repeated the ARIMA fit to see if there might be a temporal autocorrelation.

```
> tsdiag(trimmed.dropped.arima)
```


Standardized Residuals
2
0
−2
0
100
200
300
400
500
Time


ACF of Residuals
ACF
1.0
0.4
−0.2
0
5
10
15
20
25
Lag

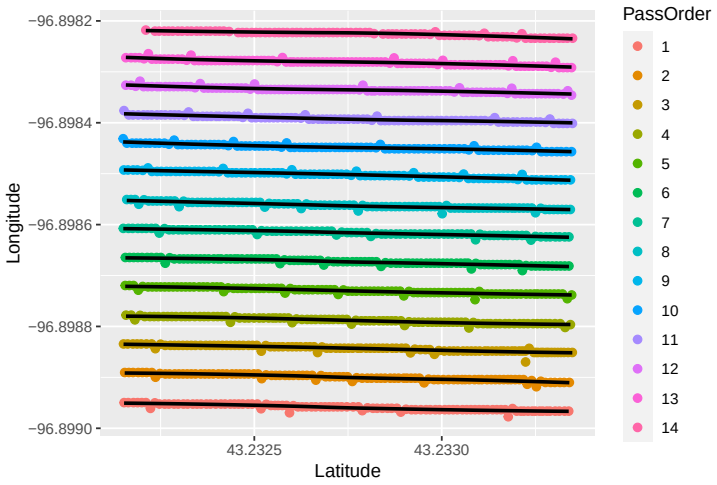

p values for Ljung−Box statistic
p value
0.8
0.4
0.0
2
4
6
8
10
lag

0.1 Correcting GPS Position

From the previous plots, it should be obvious that there are errors in GPS positioning. First, we consider horizontal displacements along the main pass axis. To illustrate this, we plot the field with passes moving horizontally; this allows us to use `ggplot`s smoothing options.

Horizontal Displacement

```
> ggplot(sample.dat, aes(Latitude,Longitude)) +
+    geom_point(aes(colour = PassOrder),size = 2) +
+    geom_smooth(method = "loess",aes(group = PassOrder),color="black")
```



Here we see that loess smoothing approximates a straight line. We should prefer loess, since planting may not be strictly linear along the length of the field.

```
> sample.dat$LonAdj <- sample.dat$Longitude
> for (pass in sample.dat$PassOrder) {
+    pass.dat <- subset(sample.dat,sample.dat$PassOrder==pass)
+    pass.loess <- loess(Longitude ~ Latitude,data=pass.dat)
+    sample.dat$LonAdj[sample.dat$PassOrder==pass] <- pass.loess$fitted
+ }
```

We can use the adjusted longitude to estimate a more linear path for the harvester.

```
> ggplot(sample.dat, aes(Latitude,LonAdj)) +
+    geom_point(aes(colour = PassOrder),size = 2) +
+    geom_smooth(method = "loess",aes(group = PassOrder),color="black")
```

```
> lon.sample90.var <- variogram(sample.dat$YldVolDry~1,
+                               locations=~LonAdj+Latitude,
+                               alpha=c(0,30,60,90),
+                               data=sample.dat)
> plot(lon.sample90.var)
```

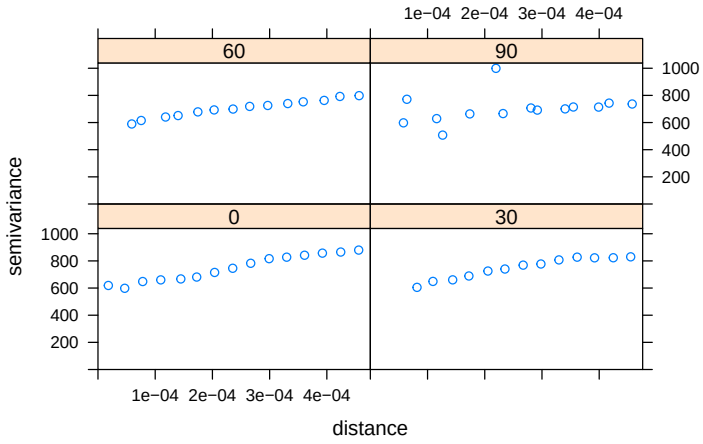


Figure 3: Sample one-direction variograms, using adjusted longitude

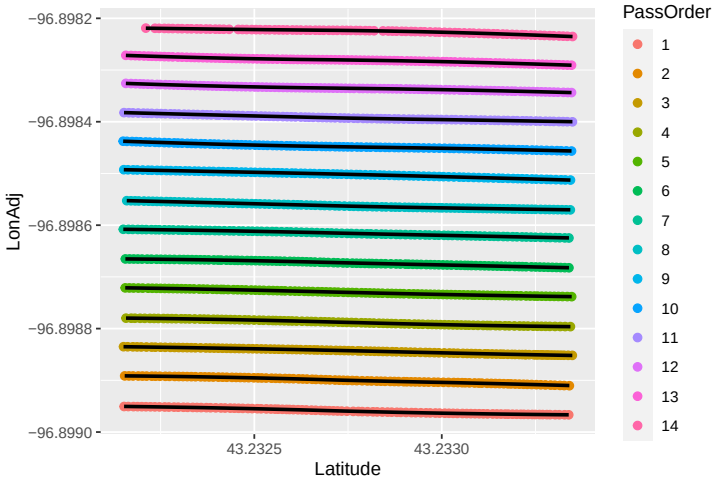
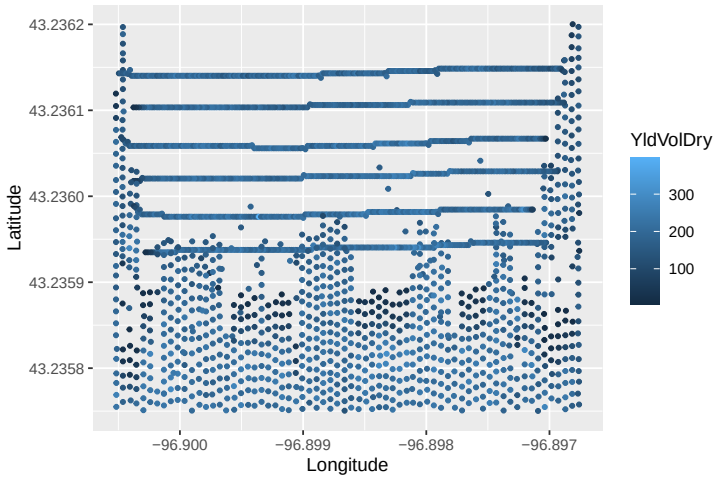


Figure **??** suggests that adjusting path has an effect on the measurable anistropy in the data. Compare this with Figure **??**

Vertical Displacement

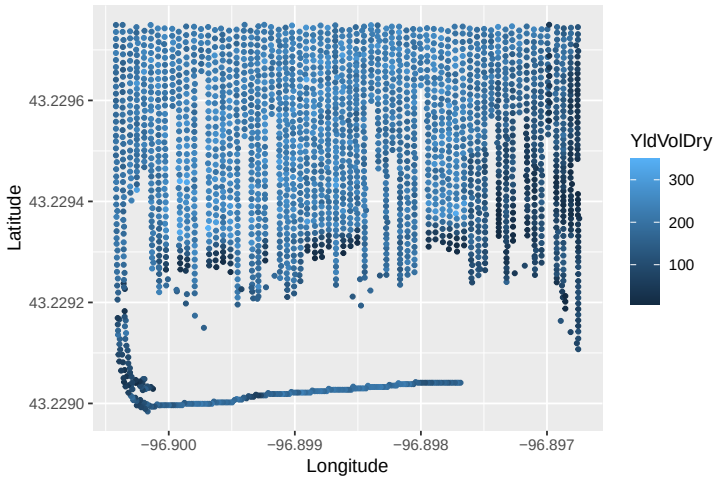
Consider the points at either end of the field.

```
> endrowsN.dat <- subset(raw.dat,raw.dat$Latitude>northBorder)
> ggplot(endrowsN.dat, aes(Longitude, Latitude)) +
+         geom_point(aes(colour = YldVolDry),size = 1)
```



Looking at the detail of the north end, we might, later, consider that the horizontal passes represent endrows. Where points appear north of the lowest horizontal pass, we may assume lag between combine position and sensor recording.

```
> endrowsS.dat <- subset(raw.dat,raw.dat$Latitude<southBorder)
> ggplot(endrowsS.dat, aes(Longitude, Latitude)) +
+     geom_point(aes(colour = YldVolDry),size = 1)
```



There appears to be a similar displacement at the south end of the field. Many references ([?, ?, ?, ?]) state that commercial yield sensors adjust for discrepancies between the point at which grain is harvested and the point at which grain yield is recorded, based on expected transit times through the harvester. However, this may not work as expected; there are point estimates for yield where there should be no grain in the field. Sudduth and Drummond [?] describe a method for editing passes using a time-based delay. We consider adusting points spatially.

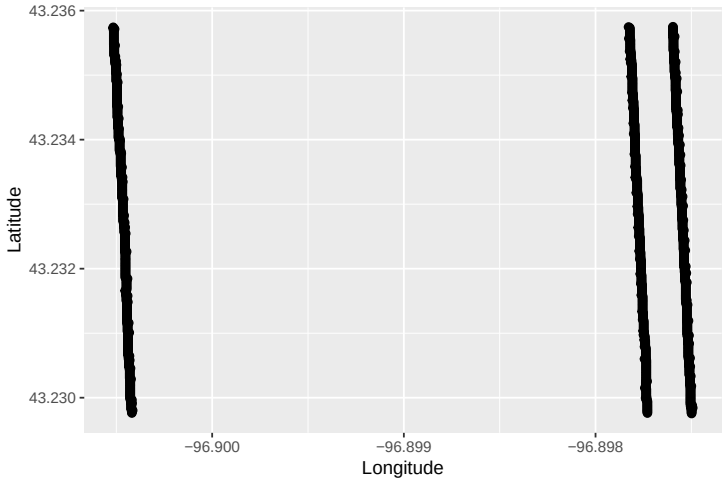
Can we improve yield estimates if we adjust for vertical displacement? To explore this further, we need to determine direction of travel. This will not be as easy as we might hope. If pass number were recorded strictly for each single pass in one direction, we might adjust passes based only on `PassNum`. We can determine which passes are unusual by counting the number of points per each pass.

```
> tapply(trimmed.dat$PassNum, list(trimmed.dat$Pass),length)
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
856	857	864	1288	760	864	439	390	421	300	103	132	421	304	69
16	17	18	19	21	22	23	24	25	26	27	28	29	30	31
3	8	355	302	7	417	290	540	541	729	757	706	575	873	719
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
154	263	144	142	403	259	420	250	442	281	433	270	155	142	468
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
431	424	271	460	440	148	149	495	494	487	308	496	337	507	512
62	63	64	65	66	67	68	69	70	71	72	73			
154	156	528	352	521	486	498	518	472	486	473	478			

Most passes have about 450 points. Pass 4 has nearly 1300.

```
> ggplot(subset(trimmed.dat,trimmed.dat$PassNum==4), aes(Longitude, Latitude)) + geom_point(
```



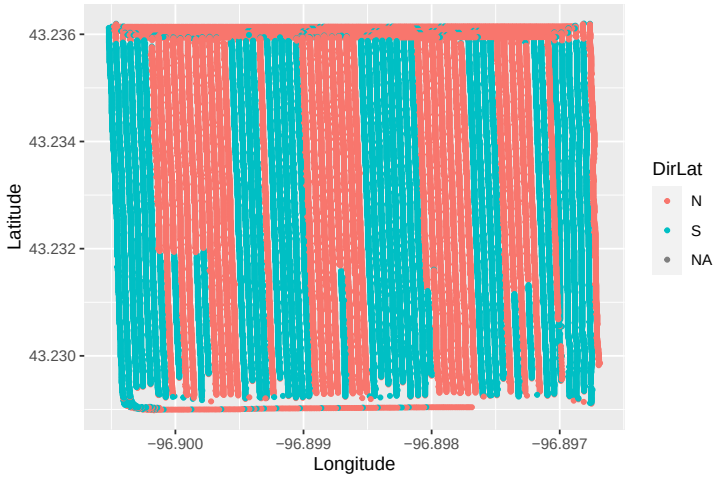
This pass number corresponds to three distinct passes, separated by several other passes horizontally. We may try to estimate passes, instead.

Estimating Pass Number

First attempt, we realize that when moving north to south (or vice-versa), latitude will change much more rapidly than longitude. For 45 degree path, longitude and latitude will change at the same rate, so if we use this as a cutoff, we can allow for some jumps in GPS - see **??**. So, we'll iterate over the data set, and increment pass when $\Delta lon > 2 \times \Delta lat$. I've found this works better identifying north-south passes than strict equality. This does overestimate the number of east-west passes, but we can remove these later.

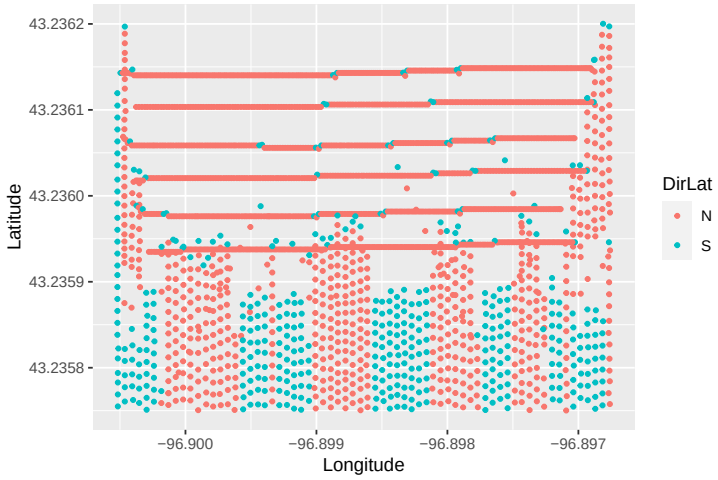
```
> raw.dat$ImpPass <- 0
> impPass <- 1
> i=1
> raw.dat$ImpPass[i] <- impPass
> raw.dat$DirLon <- NA
> raw.dat$DirLat <- NA
> while(i<length(raw.dat$ImpPass)) {
+    delta.Lon <- raw.dat$Longitude[i+1] - raw.dat$Longitude[i]
+    delta.Lat <- raw.dat$Latitude[i+1] - raw.dat$Latitude[i]
+    if(delta.Lon > 0 ) {
+       #moving west
+       raw.dat$DirLon[i+1] <- "W"
+    } else {
+       raw.dat$DirLon[i+1] <- "E"
+    }
+
+    if(delta.Lat > 0 ) {
+       #moving south
+       raw.dat$DirLat[i+1] <- "S"
+    } else {
+       raw.dat$DirLat[i+1] <- "N"
+    }
+
+    delta.Lon <- abs(delta.Lon)
+    delta.Lat <- abs(delta.Lat)
+
+    if(delta.Lon > delta.Lat) {
+       impPass = impPass+1
+    }
+    raw.dat$ImpPass[i+1] <- impPass
+    i <- i+1
+ }
> raw.dat$DirLon <- as.factor(raw.dat$DirLon)
```

```
> raw.dat$DirLat <- as.factor(raw.dat$DirLat)

> ggplot(raw.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = DirLat),size = 1)
```



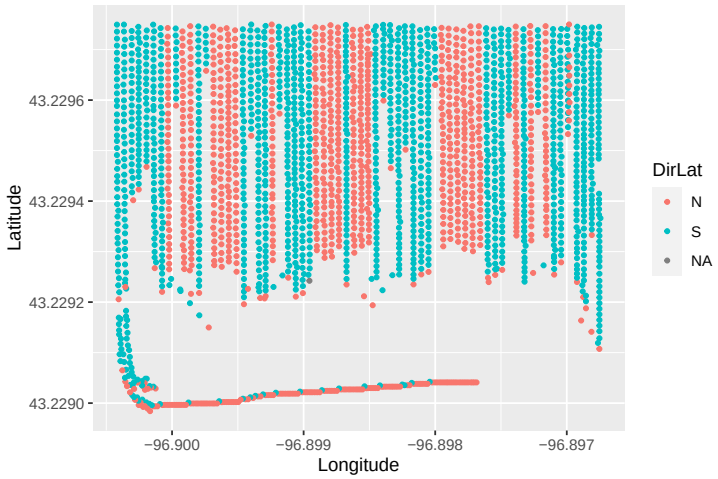
Note that there are many incomplete passes, finished in opposite directions.

```
> endrowsN.dat <- subset(raw.dat,raw.dat$Latitude>northBorder)
> ggplot(endrowsN.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = DirLat),size = 1)
```



At the north end, we see an offset from the southernmost endrow ...

```
> endrowsS.dat <- subset(raw.dat,raw.dat$Latitude<southBorder)
> ggplot(endrowsS.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = DirLat),size = 1)
```



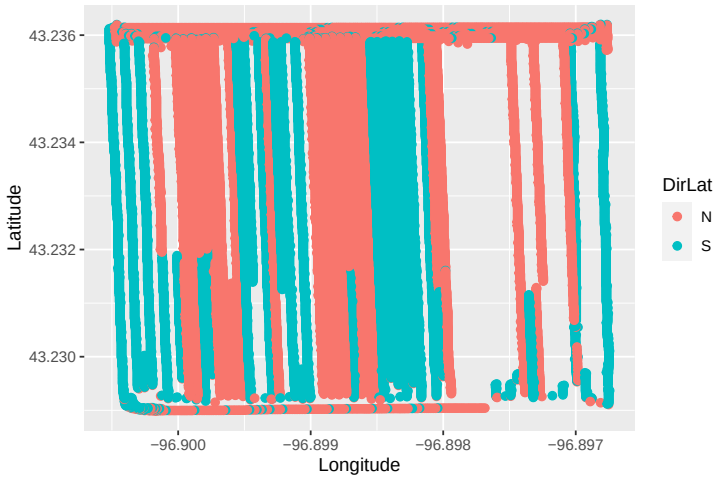
... and a corresponding offset at the south end of the field

This suggests that we can reduce some error by shifting passes. This will have an effect on passes that are harvested in different directions. Points in these passes are offset relative to each other and kriging will be based on incorrect linear distances. This will affect more than just pairs of passes in opposite directions, depending on the sphere of influence detected by the variogram.

```
> raw.dat$ImpPass <- as.factor(raw.dat$ImpPass)
> raw.passes <- tapply(raw.dat$PassNum, list(raw.dat$PassNum),length)
> raw.pasimp <- tapply(raw.dat$ImpPass, list(raw.dat$ImpPass),length)
```

We still don't compute passes as completely as we might wish. There are still passes that are shorter than the length of the field, as we might expect.

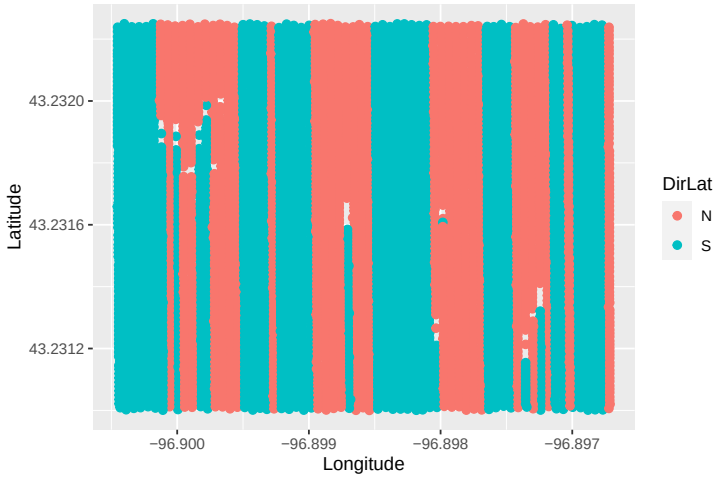
```
> problem.passes <- names(raw.passes[(which(raw.passes<400 | raw.passes>550))])
> problem.imp <- names(raw.pasimp[(which(raw.pasimp<400 | raw.pasimp>550))])
> problem.dat <- subset(raw.dat,raw.dat$ImpPass %in% problem.imp)
> problem.dat$ImpPass <- as.factor(as.numeric(problem.dat$ImpPass))

> ggplot(problem.dat, aes(Longitude, Latitude)) + geom_point(aes(color=DirLat),size = 2)
```



Note that most of the incomplete passes are in the north endrows, which we exclude from analysis. However, there are some incomplete passes in the main data. Note that some passes in the same geographic pass (in line with planting) appear to be harvested in different directions

If we consider an interior portion of the map, there appear to be many breaks along a line. Perhaps there is a geographical feature? Some of these gaps may be explained by a partial harvest pass, followed by a finishing pass in the opposite direction. This is an issue that may be addressed in execution.

```
> breaks.dat <- subset(raw.dat,raw.dat$Latitude < 43.23225)
> breaks.dat <- subset(breaks.dat,breaks.dat$Latitude > 43.2310)
> ggplot(breaks.dat, aes(Longitude, Latitude)) + geom_point(aes(color=DirLat),size = 2)
```



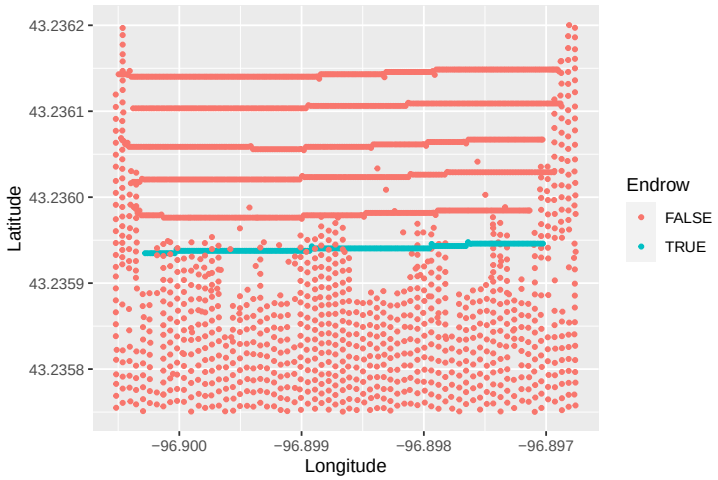
Computing Vertical Offset

We can use the north endrows to estimate a mean offset, based on direction traveled. First, we need to identify the passes that are unique to the north endrows. We assume that passes in the trimmed data set are strictly north-south.

```
> endrowsN.dat$Endrow <- !(endrowsN.dat$PassNum %in% trimmed.dat$PassNum)
```

Plot this to check our assumptions.

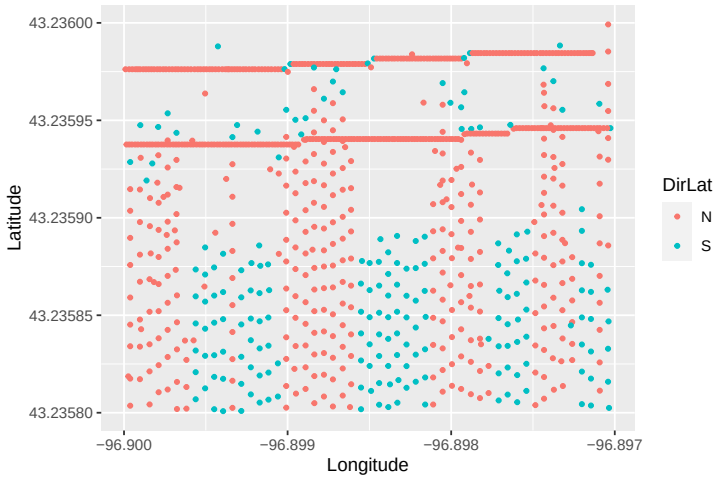
```
> ggplot(endrowsN.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = Endrow),size = 1)
```



It appears that pass number continues across the north end of the field. However, we note the southern-most endrow pass is uniquely identified.

We note that the sides of the field have passes that extend well into the endrow range. We can trim from the sides and remove a portion of the northern endrow. I've chosen latitude and longitude by inspection.

```
> endrowStrip <- subset(raw.dat,raw.dat$Latitude < 43.236)
> endrowStrip <- subset(endrowStrip,endrowStrip$Latitude > 43.2358)
> endrowStrip <- subset(endrowStrip,endrowStrip$Longitude < -96.897)
> endrowStrip <- subset(endrowStrip,endrowStrip$Longitude > -96.900)
> ggplot(endrowStrip, aes(Longitude, Latitude)) + geom_point(aes(colour = DirLat),size = 1)
```



We now have only two east-west passes. Since this data set is trimmed from north to south, we can remove these two rows based on the number of observations.

```
> endrowStrip$Pass <- as.factor(endrowStrip$PassNum)
> endrowStrip.count <- tapply(endrowStrip$Pass, list(endrowStrip$PassNum), length)
```

The east-west passes are the only passes with more than 100 observations, so select on that criteria.

```
> ew.rows <- names(endrowStrip.count)[which(endrowStrip.count>100)]
> ew.rows <- as.numeric(ew.rows)
> ew.rows
```

```
[1] 17 20
```

Looks like we take out rows 17 and 20.

```
> endrowStrip <- subset(endrowStrip, !(endrowStrip$PassNum %in% ew.rows))
```

```
> tapply(endrowStrip$Latitude, list(endrowStrip$PassNum), max)
```

```
       1        2        3        4        5        6        7        8
43.23596 43.23598 43.23596 43.23595 43.23597 43.23595 43.23596 43.23584
       9       10       13       14       18       19       22       23
43.23590 43.23596 43.23588 43.23596 43.23586 43.23600 43.23593 43.23597
      24       25       26       27       28       29       30       31
43.23595 43.23587 43.23594 43.23598 43.23595 43.23599 43.23587 43.23595
      37       39       41       46       47       50       54       55
43.23592 43.23595 43.23593 43.23588 43.23594 43.23587 43.23588 43.23598
      56       57       58       59       60       61       64       65
43.23588 43.23599 43.23588 43.23596 43.23589 43.23597 43.23588 43.23596
      66       67       68       69       70       71       72       73
43.23589 43.23598 43.23587 43.23595 43.23589 43.23595 43.23588 43.23596
```

Now we classify passes by direction. We'll take a ratio to allow for endpoints. The direction of travel for endpoints may not be correctly inferred, since direction is based on distance from a previous point.

```
> endrowStrip$North <- endrowStrip$DirLat=="N"
> endrowStrip$South <- endrowStrip$DirLat=="S"
> Ncounts <- tapply(endrowStrip$North,list(endrowStrip$Pass),sum)
> counts <- tapply(endrowStrip$South,list(endrowStrip$Pass),length)
> nRatio <- Ncounts/counts
> Northward <- nRatio>0.5
> Northward <- as.factor(Northward)
```

Having classified passes by direction, we create subsets for later use.

```
> northPasses <- as.numeric(names(Northward)[Northward==TRUE])
> southPasses <- as.numeric(names(Northward)[Northward==FALSE])
> southPasses <- subset(southPasses,!is.na(southPasses))
> northPasses <- subset(northPasses,!is.na(northPasses))
```

Now we compute for each pass the northern most point, and take the average for north and south passes.

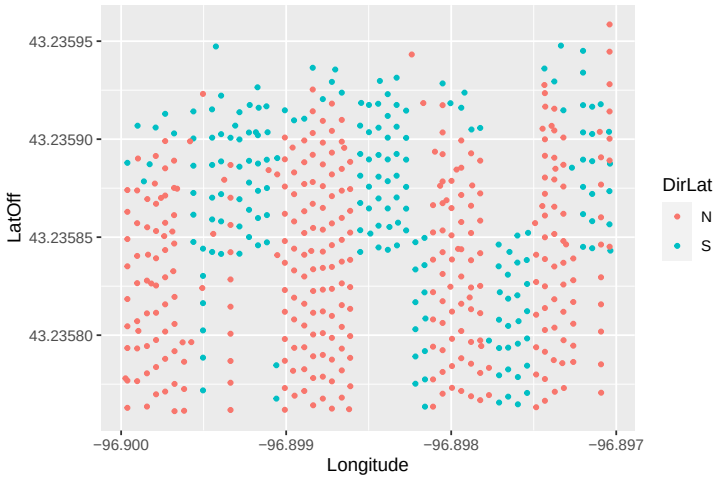
```
> maxPoint <- tapply(endrowStrip$Latitude,list(endrowStrip$Pass),max)
> NSpoints <- tapply(maxPoint,list(Northward),mean)
```

Now we can compute an offset; we assume the midpoint between pass maximums for north and south passes.

```
> offset <- abs((NSpoints[2]-NSpoints[1]))/2
> endrowStrip$LatOff <- endrowStrip$Latitude + offset
> mask <- endrowStrip$Pass %in% northPasses
> endrowStrip$LatOff[mask] <- endrowStrip$Latitude[mask] - offset
```

So, what does the offset look like?

```
> ggplot(endrowStrip, aes(Longitude, LatOff)) + geom_point(aes(colour = DirLat),size = 1)
```



This doesn't adjust all the passes in the endrow set but we might improve this when we use estimated pass numbers in the larger data set.

Estimating passes, trimmed data set

Now we apply the shift to a slightly larger data set than the trimmed set. After we make adjustments, we'll trim to match our previous data.

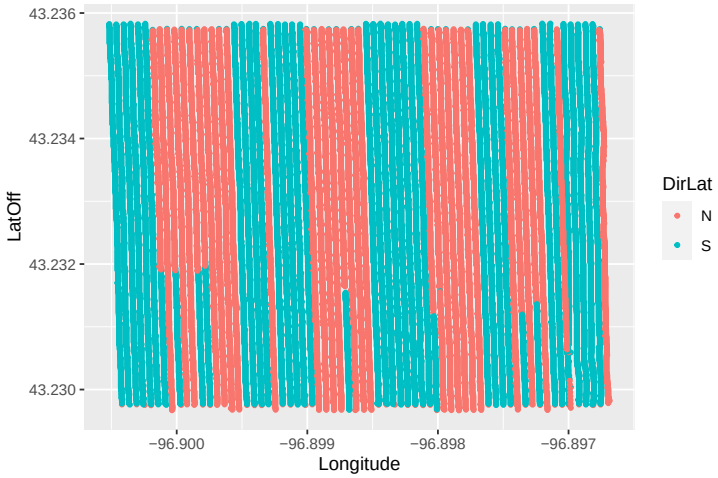
```
> northExtra <- northBorder+offset
> southExtra <- southBorder-offset
> reduced.dat <- subset(raw.dat,raw.dat$Latitude>=southExtra)
> reduced.dat <- subset(reduced.dat,reduced.dat$Latitude<=northExtra)
```

Repeat pass estimation, without the endrows. This may improve our ability to detect passes. I've also included a check for a time gap. Since we've trimmed ends, each geographical pass represents a gap in sample ids.

```
> reduced.dat$ImpPass <- 0
> impPass <- 1
> i <- 1
> reduced.dat$ImpPass[i] <- impPass
> reduced.dat$DirLon <- NA
> reduced.dat$DirLat <- NA
> reduced.dat$DirLon[i] <- "W"
> reduced.dat$DirLat[i] <- "N"
> current.count = 1
> while(i < length(reduced.dat$ImpPass)) {
+
+    delta.Lon <- reduced.dat$Longitude[i+1]-reduced.dat$Longitude[i]
+    delta.Lat <- reduced.dat$Latitude[i+1]-reduced.dat$Latitude[i]
+    if(delta.Lon > 0 ) {
```

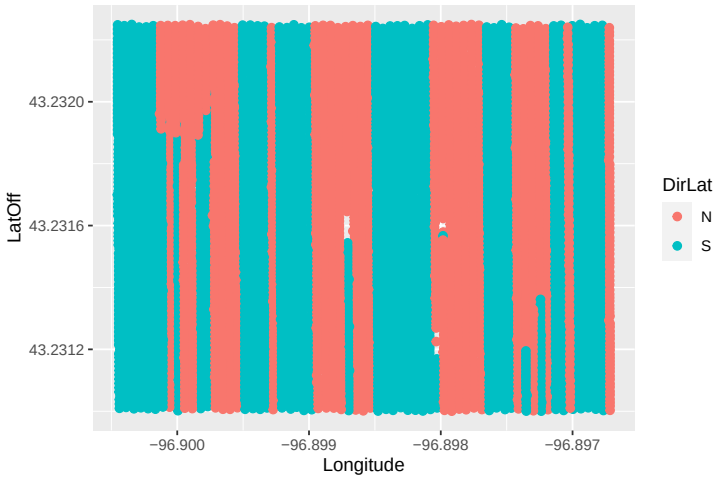
```
+       #moving west
+       reduced.dat$DirLon[i+1] <- "W"
+    } else {
+       reduced.dat$DirLon[i+1] <- "E"
+    }
+
+    if(delta.Lat > 0 ) {
+       #moving south
+       reduced.dat$DirLat[i+1] <- "S"
+    } else {
+       reduced.dat$DirLat[i+1] <- "N"
+    }
+
+    #is there a time gap
+    #time.gap <- reduced.dat$ID[i+1]-reduced.dat$ID[i]
+    #OID works better than ID,
+    time.gap <- reduced.dat$OID[i+1]-reduced.dat$OID[i]
+
+    if(time.gap > 1) {
+       impPass = impPass+1
+    }
+
+    reduced.dat$ImpPass[i+1] <- impPass
+    i <- i+1
+    current.count <- current.count+1
+ }
> reduced.dat$DirLon <- as.factor(reduced.dat$DirLon)
> reduced.dat$DirLat <- as.factor(reduced.dat$DirLat)
> reduced.dat$ImpPass <- as.factor(reduced.dat$ImpPass)
> tapply(reduced.dat$ImpPass,list(reduced.dat$ImpPass),length)

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
421 425 447 402 459 423 445 445 444 429 465 310 452 396 427 303 241 427 307
 20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
441 313 422 293 223 415 332 450 462 461 439 121 437 434 442 322 435 388 424
 39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
286 304 292 408 262 424 253 449 284 439 272 303 474 437 430 274 466 446 302
 58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
501 501 495 311 503 341 515 518 317 534 356 528 492 506 526 478 492 480 485

> reduced.dat$North <- reduced.dat$DirLat=="N"
> reduced.dat$South <- reduced.dat$DirLat=="S"
> Ncounts <- tapply(reduced.dat$North,list(reduced.dat$ImpPass),sum,na.rm=TRUE)
> Ncounts[is.na(Ncounts)] <- 0
> counts <- tapply(reduced.dat$North,list(reduced.dat$ImpPass),length)
> nRatio <- Ncounts/counts
```

```
> Northward <- nRatio>0.5
> Northward <- as.factor(Northward)
> northPasses <- as.numeric(names(Northward)[Northward==TRUE])
> southPasses <- as.numeric(names(Northward)[Northward==FALSE])
> southPasses <- subset(southPasses,!is.na(southPasses))
> northPasses <- subset(northPasses,!is.na(northPasses))
> reduced.dat$LatOff <- reduced.dat$Latitude + offset
> mask <- reduced.dat$ImpPass %in% northPasses
> reduced.dat$LatOff[mask] <- reduced.dat$Latitude[mask] - offset

> ggplot(reduced.dat, aes(Longitude, LatOff)) + geom_point(aes(colour = DirLat),size = 1)
```



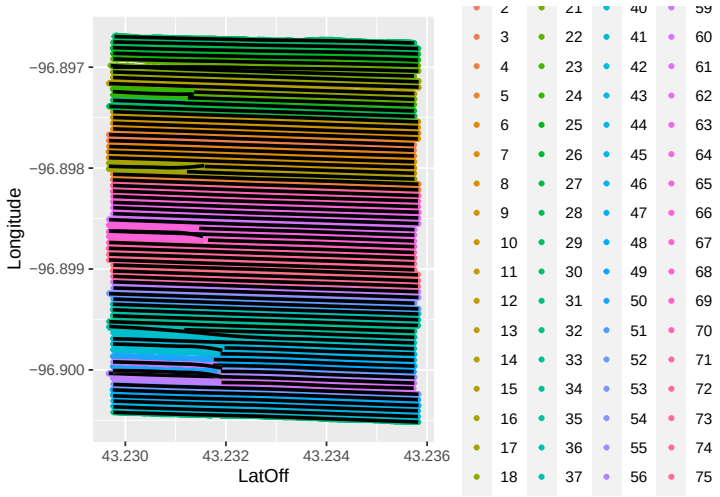
This appears to remove some of the gaps in the center of the field, so we'll examine the middle in greater detail

```
> red.breaks.dat <- subset(reduced.dat,reduced.dat$LatOff < 43.23225)
> red.breaks.dat <- subset(red.breaks.dat,red.breaks.dat$LatOff > 43.2310)
> ggplot(red.breaks.dat, aes(Longitude, LatOff)) + geom_point(aes(color=DirLat),size = 2)
```



This reduces most of the mid-field gaps, but we are left with shortened passes. If we try to adjust horizontal displacements, we combine some passes.

```
> ggplot(reduced.dat, aes(LatOff,Longitude)) +
+     geom_point(aes(colour = ImpPass),size = 1) +
+     geom_smooth(method = "loess",aes(group = ImpPass),color="black")
```



We note that short passes may result from a change in direction, so I've added a check for change in direction, and a check for the length of the current pass. When we guess direction, the end point in a pass is not always correct.

```
> reduced.dat$ImpPass <- 0
> impPass <- 1
> i <- 1
> reduced.dat$ImpPass[i] <- impPass
> reduced.dat$DirLon <- NA
> reduced.dat$DirLat <- NA
> reduced.dat$DirLon[i] <- "W"
> reduced.dat$DirLat[i] <- "N"
> current.count = 1
> while(i < length(reduced.dat$ImpPass)) {
+
+    delta.Lon <- reduced.dat$Longitude[i+1]-reduced.dat$Longitude[i]
+    delta.Lat <- reduced.dat$Latitude[i+1]-reduced.dat$Latitude[i]
+    if(delta.Lon > 0 ) {
+       #moving west
+       reduced.dat$DirLon[i+1] <- "W"
+    } else {
+       reduced.dat$DirLon[i+1] <- "E"
+    }
+
+    if(delta.Lat > 0 ) {
+       #moving south
+       reduced.dat$DirLat[i+1] <- "S"
+    } else {
+       reduced.dat$DirLat[i+1] <- "N"
+    }
+
+    #is there a time gap
+    #time.gap <- reduced.dat$ID[i+1]-reduced.dat$ID[i]
+    #OID works better than ID,
+    time.gap <- reduced.dat$OID[i+1]-reduced.dat$OID[i]
+
+    delta.Lon <- abs(delta.Lon)
+    delta.Lat <- abs(delta.Lat)
+
+    if(time.gap > 1) {
+       if(current.count>2) {
+          impPass = impPass+1
+          current.count <- 1
+       }
+    } else if(reduced.dat$DirLat[i+1] != reduced.dat$DirLat[i]) {
+       if(current.count>2) {
```

```
+          impPass = impPass+1
+          current.count <- 1
+       }
+    }
+
+    reduced.dat$ImpPass[i+1] <- impPass
+    i <- i+1
+    current.count <- current.count+1
+ }
> reduced.dat$DirLon <- as.factor(reduced.dat$DirLon)
> reduced.dat$DirLat <- as.factor(reduced.dat$DirLat)
> reduced.dat$ImpPass <- as.factor(reduced.dat$ImpPass)
> tapply(reduced.dat$ImpPass,list(reduced.dat$ImpPass),length)

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
421 425 447 402 459 423 445 445 444 429 465 310 452 396 427 303 107 134 427
 20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
307  72 369 313 422 293 115 108 415 332 450 462 461 439 121 437 434 442 322
 39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
435 388 424 286 157 147 147 145 408 262 424 253 449 284 439 272 158 145 474
 58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
437 430 274 466 446 151 151 501 501 495   2 309 503 341 515 518 158 159 534
 77  78  79  80  81  82  83  84  85
356 528 492 506 526 478 492 480 485
```

This looks good; all passes have a reasonable number of points. We'll further process as before.

```
> reduced.dat$North <- reduced.dat$DirLat=="N"
> reduced.dat$South <- reduced.dat$DirLat=="S"
> Ncounts <- tapply(reduced.dat$North,list(reduced.dat$ImpPass),sum,na.rm=TRUE)
> Ncounts[is.na(Ncounts)] <- 0
> counts <- tapply(reduced.dat$North,list(reduced.dat$ImpPass),length)
> nRatio <- Ncounts/counts
> Northward <- nRatio>0.5
> Northward <- as.factor(Northward)
> northPasses <- as.numeric(names(Northward)[Northward==TRUE])
> southPasses <- as.numeric(names(Northward)[Northward==FALSE])
> northPasses

 [1]  2  4  6  8 10 12 14 16 18 20 23 25 27 29 31 33 36 38 40 42 44 46 48 50
[25] 52 54 56 58 60 62 64 66 69 71 73 75 77 79 81 83 85

> southPasses

 [1]  1  3  5  7  9 11 13 15 17 19 21 22 24 26 28 30 32 34 35 37 39 41 43 45
[25] 47 49 51 53 55 57 59 61 63 65 67 68 70 72 74 76 78 80 82 84
```

```
> ggplot(subset(adj.trimmed.dat,adj.trimmed.dat$ImpPassNum==longest_pass), aes(Longitude, La
```

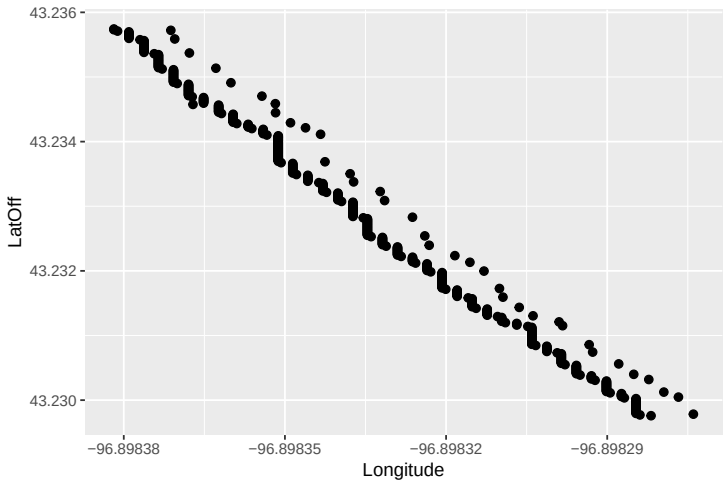


Figure 4: Longest pass in trimmed data

```
> southPasses <- subset(southPasses,!is.na(southPasses))
> northPasses <- subset(northPasses,!is.na(northPasses))
> reduced.dat$LatOff <- reduced.dat$Latitude + offset
> mask <- reduced.dat$ImpPass %in% northPasses
> reduced.dat$LatOff[mask] <- reduced.dat$Latitude[mask] - offset
```

Now we trim the reduced data set to match our previously trimmed set.

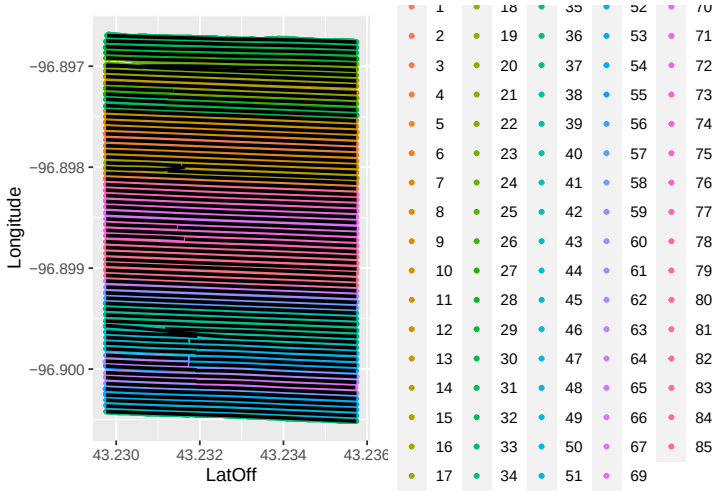
```
> adj.trimmed.dat <- subset(reduced.dat,reduced.dat$LatOff>=southBorder)
> adj.trimmed.dat <- subset(adj.trimmed.dat,adj.trimmed.dat$LatOff<=northBorder)
```

We want to visualize passes. We also want to remove the extra imputed pass numbers.

```
> adj.trimmed.dat$ImpPassNum <- as.numeric(adj.trimmed.dat$ImpPass)
> adj.trimmed.dat$ImpPass <- as.factor(adj.trimmed.dat$ImpPassNum)
> counts <- tapply(adj.trimmed.dat$ImpPass,list(adj.trimmed.dat$ImpPass),length)
> longest_pass <- as.numeric(names(counts)[which(counts==max(counts))])
```

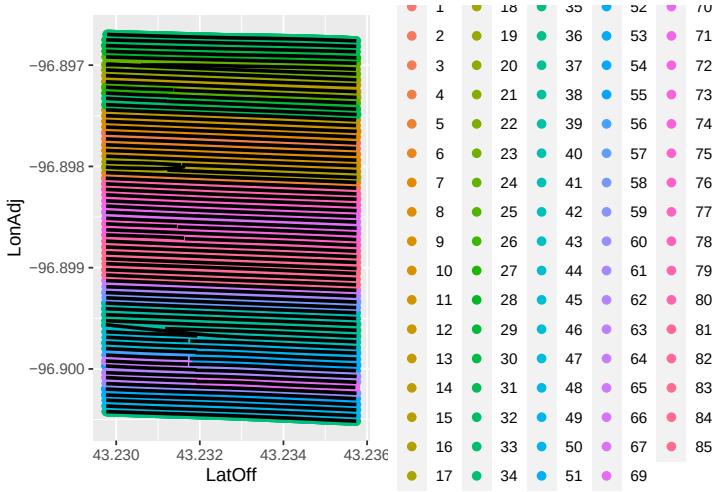
Figure **??** shows the longest, with regard to the number of points, in the adjusted and trimmed data. Note that since there is only one pass in the, horizontal displacement is greatly exaggerated. Now that we've estimated passes, we can try to estimate longitude for the full data set.

```
> ggplot(adj.trimmed.dat, aes(LatOff,Longitude)) +
+    geom_point(aes(colour = ImpPass),size = 1) +
+    geom_smooth(method = "loess",aes(group = ImpPass),color="black")
```



We see that we don't combine short passes.

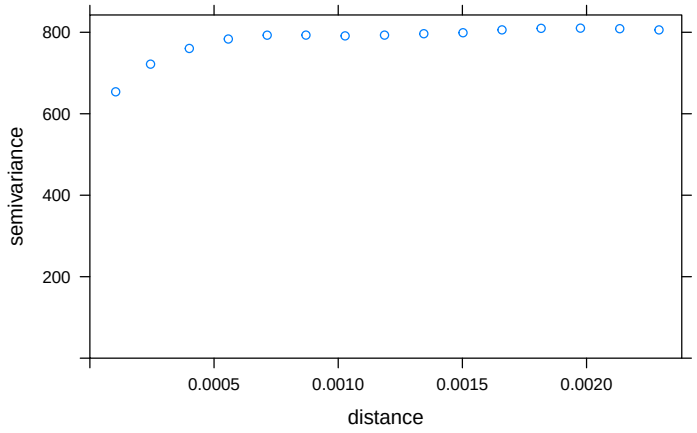
```
> adj.trimmed.dat$LonAdj <- adj.trimmed.dat$Longitude
> adj.trimmed.dat$LonIsAdj <- FALSE
> for (pass in adj.trimmed.dat$ImpPass) {
+    pass.dat <- subset(adj.trimmed.dat,adj.trimmed.dat$ImpPass==pass)
+    if(length(pass.dat$LonAdj)<550 && length(pass.dat$LonAdj)>100) {
+       pass.loess <- loess(Longitude ~ LatOff,data=pass.dat)
+       adj.trimmed.dat$LonAdj[adj.trimmed.dat$ImpPass==pass] <- pass.loess$fitted
+       adj.trimmed.dat$LonIsAdj[adj.trimmed.dat$ImpPass==pass] <- TRUE
+    }
+ }

> ggplot(adj.trimmed.dat, aes(LatOff,LonAdj)) +
+    geom_point(aes(colour = ImpPass),size = 2) +
+    geom_smooth(method = "loess",aes(group = ImpPass),color="black")
```



0.1.1 Kriging with adjusted points

Now we can repeat our previous analysis.

```
> adj.trimmed.var <- variogram(YldVolDry~1,
+                              locations=~LonAdj+LatOff,
+                              data=adj.trimmed.dat)
> plot(adj.trimmed.var)
```

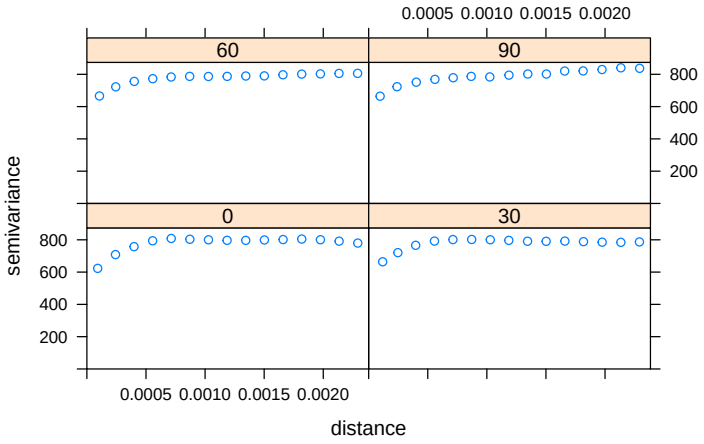



```
> adj.trimmed.vgm <- fit.variogram(adj.trimmed.var, vgm(800,"Exp",0.0004,500))
> adj.trimmed.vgm
```

```
  model    psill        range
1   Nug 568.4244 0.0000000000
2   Exp 232.4126 0.0002266433
```

Examining the variogram, we get a small - very small - decrease in the sill. So, only a very small change our estimate of spatial correlation.

```
> adj.trimmed90.var <- variogram(YldVolDry~1,
+                            locations=~LonAdj+LatOff,
+                            alpha=c(0,30,60,90),
+                            data=adj.trimmed.dat)
> plot(adj.trimmed90.var)
```

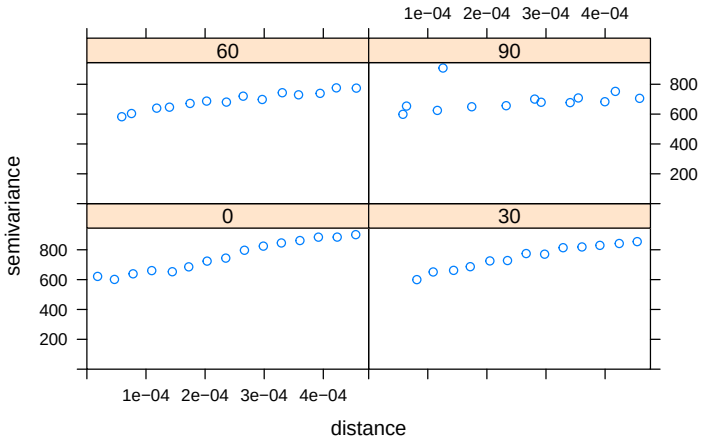


0.1.2 Kriging sample data

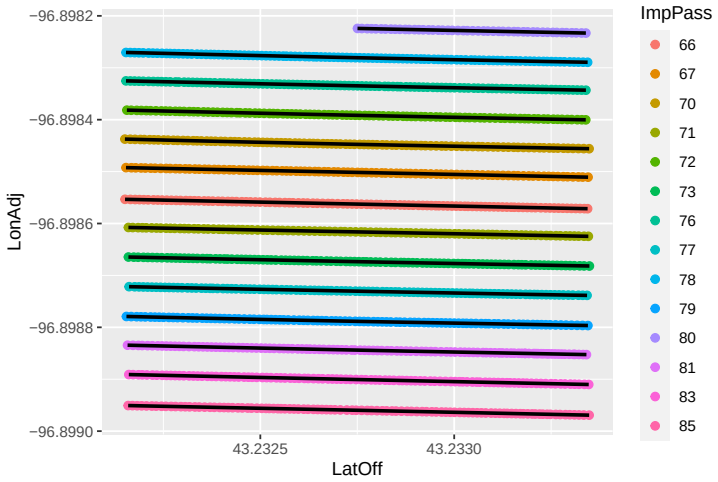
Repeat the analysis on the sample data. We remember that there were some issues with estimating variograms. Does correcting points change this?

```
> min.lat <- min(adj.trimmed.dat$LatOff)
> max.lat <- max(adj.trimmed.dat$LatOff)
> min.lon <- min(adj.trimmed.dat$LonAdj)
> max.lon <- max(adj.trimmed.dat$LonAdj)
> lat.rng <- max.lat-min.lat
> lon.rng <- max.lon-min.lon
> adj.trimmed.dat$Sample <- adj.trimmed.dat$LatOff>(min.lat + 2*(lat.rng/5)) &
+                     adj.trimmed.dat$LatOff<(max.lat - 2*(lat.rng/5)) &
+                     adj.trimmed.dat$LonAdj>(min.lon + 2*(lon.rng/5)) &
+                     adj.trimmed.dat$LonAdj<(max.lon - 2*(lon.rng/5))
> adj.sample.dat <- subset(adj.trimmed.dat,adj.trimmed.dat$Sample)

> adj.sample90.var <- variogram(YldVolDry~1,
+                            locations=~LonAdj+LatOff,
+                            alpha=c(0,30,60,90),
+                            data=adj.sample.dat)
```

```
> plot(adj.sample90.var)
> adj.sample.var <- variogram(YldVolDry~1,
+                             locations=~LonAdj+LatOff,
+                             data=adj.sample.dat)
> adj.sample.vgm <- fit.variogram(adj.sample.var, vgm(800,"Exp",0.0004,500))
> adj.sample.vgm

  model     psill      range
1   Nug   613.672 0.00000000
2   Exp 12784.646 0.01848298

> plot(adj.sample.var)
```



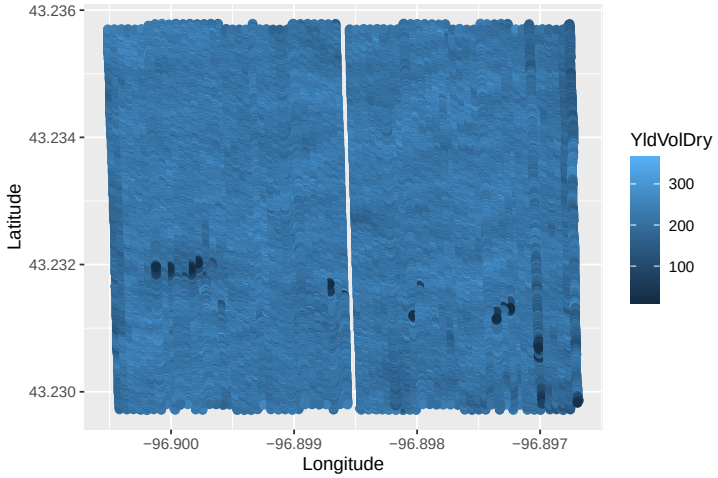

```
> ggplot(adj.sample.dat, aes(LatOff,LonAdj)) +
+    geom_point(aes(colour = ImpPass),size = 2) +
+    geom_smooth(method = "loess",aes(group = ImpPass),color="black")
```



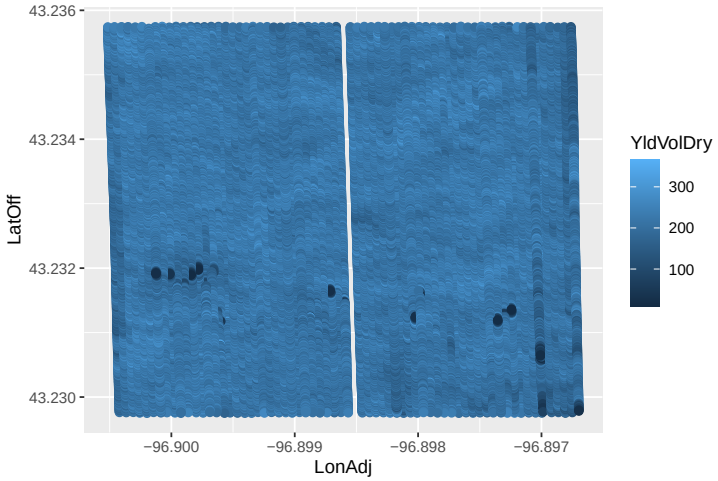
Interpolation using adjusted locations

Repeating interpolation of a missing pass.

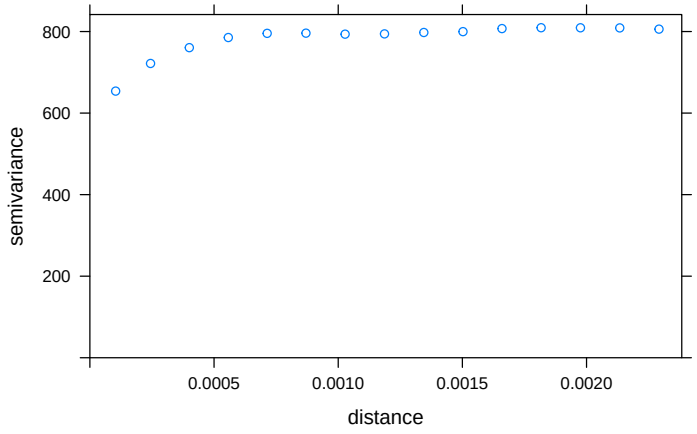
```
> adj.trimmed.drop1.dat <- subset(adj.trimmed.dat,adj.trimmed.dat$PassNum!=dropped.pass)
> adj.trimmed.dropped.dat <- subset(adj.trimmed.dat,adj.trimmed.dat$PassNum==dropped.pass)

> ggplot(adj.trimmed.drop1.dat, aes(Longitude, Latitude)) + geom_point(aes(colour = YldVolDr
```




```
> ggplot(adj.trimmed.drop1.dat, aes(LonAdj, LatOff)) + geom_point(aes(colour = YldVolDry),si
```




```
> adj.trimmed.dropped.var <- variogram(adj.trimmed.drop1.dat$YldVolDry~1, locations=~LonAdj+
> plot(adj.trimmed.dropped.var)
```



Kriging the excluded pass from the whole field data is very slow - on my machine, this step took about 2 hours. I'm caching the results.

```
> adj.trimmed.dropped.vgm <- fit.variogram(adj.trimmed.dropped.var,
+                                          vgm(800,"Exp",0.0004,500))
> if(!file.exists("adj.trimmed.dropped.krig.Rda")) {
+    adj.trimmed.dropped.krig <- krige(id="YldVolDry", formula=YldVolDry~1,
+                                      data = adj.trimmed.drop1.dat,
+                                      newdata = adj.trimmed.dropped.dat,
+                                      model = adj.trimmed.dropped.vgm,
+                                      locations=~LonAdj+LatOff)
```

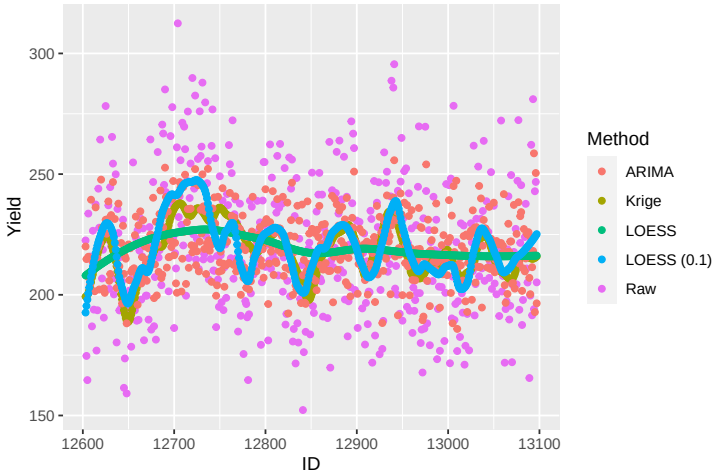
```
+    save(adj.trimmed.dropped.krig,file="adj.trimmed.dropped.krig.Rda")
+ } else {
+    load("adj.trimmed.dropped.krig.Rda")
+ }
```

```
[using ordinary kriging]
```

Since this is larger data set, we'll use both the default loess span and a reduced span.

```
> adj.trimmed.dropped.arima <- auto.arima(adj.trimmed.dropped.dat$YldVolDry)
> adj.trimmed.dropped.arima

Series: adj.trimmed.dropped.dat$YldVolDry
ARIMA(1,0,5) with non-zero mean

Coefficients:
          ar1     ma1     ma2     ma3     ma4      ma5      mean
      -0.5278  0.7377  0.1509  0.4603  0.1761  -0.2472  219.6585
s.e.   0.0985  0.0958  0.0563  0.0519  0.0575   0.0438    1.6165

sigma^2 estimated as 591.4:  log likelihood=-2279.11
AIC=4574.22   AICc=4574.51   BIC=4607.85

> adj.trimmed.dropped.loess <- loess(YldVolDry ~ ID, data=adj.trimmed.dropped.dat)
> adj.trimmed.dropped.loess

Call:
loess(formula = YldVolDry ~ ID, data = adj.trimmed.dropped.dat)

Number of Observations: 495
Equivalent Number of Parameters: 4.35
Residual Standard Error: 26.78

> adj.trimmed.dropped.loess10 <- loess(YldVolDry ~ ID, data=adj.trimmed.dropped.dat,span=0.1
> adj.trimmed.dropped.loess10

Call:
loess(formula = YldVolDry ~ ID, data = adj.trimmed.dropped.dat,
    span = 0.1)

Number of Observations: 495
Equivalent Number of Parameters: 29.39
Residual Standard Error: 25.07

> points <- length(adj.trimmed.dropped.dat$YldVolDry)
> adj.trimmed.dropped.plot <- data.frame(
+    Yield = c(adj.trimmed.dropped.dat$YldVolDry,
```

```
+             adj.trimmed.dropped.krig$YldVolDry.pred,
+             adj.trimmed.dropped.dat$YldVolDr - adj.trimmed.dropped.arima$residuals,
+             predict(adj.trimmed.dropped.loess),
+             predict(adj.trimmed.dropped.loess10)
+    ),
+    Method = c(rep("Raw",points),
+               rep("Krige",points),
+               rep("ARIMA",points),
+               rep("LOESS",points),
+               rep("LOESS (0.1)",points)
+    ),
+    ID = rep(adj.trimmed.dropped.dat$ID,5)
+ )
> adj.trimmed.dropped.plot$Method <- as.factor(adj.trimmed.dropped.plot$Method)
> ggplot(adj.trimmed.dropped.plot, aes(ID, Yield)) +
+        geom_point(aes(colour = Method),size = 1.5)
```

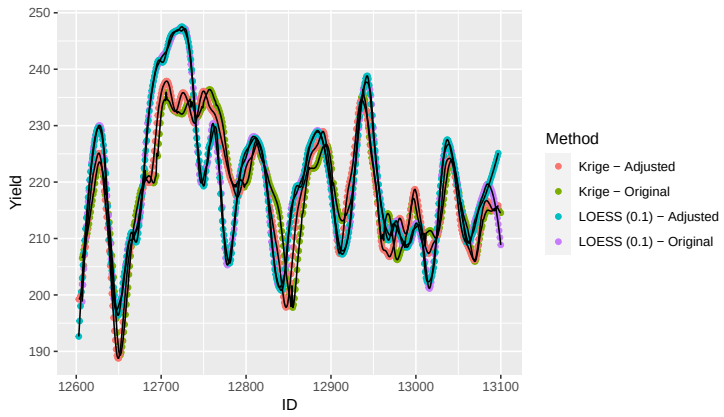


A quick comparison between kriging raw and adjusted yield values, relative to loess estimates from the dropped pass. This is just a simple sum of squares comparison.

```
> sum((predict(adj.trimmed.dropped.loess10)-adj.trimmed.dropped.krig$YldVolDry.pred)^2)

[1] 19497.6

> sum((predict(trimmed.dropped.loess10)-trimmed.dropped.krig$YldVolDry.pred)^2)

[1] 28863.23
```

There is an reduction in variance when kriging with adjusted point data.

```
> comp.dropped.plot <- data.frame(
+    Yield = c(trimmed.dropped.krig$YldVolDry.pred,
+              adj.trimmed.dropped.krig$YldVolDry.pred,
+              predict(trimmed.dropped.loess10),
+              predict(adj.trimmed.dropped.loess10)
+    ),
+    Method = c(rep("Krige - Original",length(trimmed.dropped.krig$YldVolDry.pred)),
+               rep("Krige - Adjusted",length(adj.trimmed.dropped.krig$YldVolDry.pred)),
+               rep("LOESS (0.1) - Original",length(trimmed.dropped.krig$YldVolDry.pred)),
+               rep("LOESS (0.1) - Adjusted",length(adj.trimmed.dropped.krig$YldVolDry.pred)
+    ),
+    ID = c(trimmed.dropped.dat$ID,
+           adj.trimmed.dropped.dat$ID,
+           trimmed.dropped.dat$ID,
+           adj.trimmed.dropped.dat$ID)
+ )
> comp.dropped.plot$Method <- as.factor(comp.dropped.plot$Method)
> ggplot(comp.dropped.plot, aes(ID, Yield)) +
+     geom_point(aes(colour = Method),size = 1.5) +
+     geom_line(aes(group = Method))
```



Some of the peaks and troughs in yield along the pass, interpolated by kriging on adjusted points, are shifted to be closer to that estimated by loess, as measured in time.

Further Work

In our exploration of the sample data, we dropped a single pass, then attempted to estimate the yield of that pass, using kriging. We might wish to explore this with the larger data set. For instance, how many adjacent passes are needed, on either side of a dropped pass, to estimate expected yield for that pass? We might infer this from the variogram, perhaps choosing at distance where the

variogram reaches 95% of the sill. However, we might want to confirm this by including only a limited number of passes in the varigram; starting with one adjacent pass and adding passes until a sill is reached. We need to remember that the variogram represents an average over the entire area, but may vary at different scales. We note that the variogram of the sample data was distinctly different than the variogram of the trimmed data.

A test of this type might be useful to determine the number of experimental passes that can be fit into a single field. Since passes are not always contiguous in this data, such a test is not practical.