Getting going with unpak data

The unpakathonJan2016 package is primarily a datafreeze of the unpak database at the end of Jan 2016.

This vignette shows how to interact with these data, once the package is installed and loaded into the current R session

load the library

```
library(unpakathonJan2016)
```

Tables

There are several tables available, each with some sort of help description that you can access by typing ? (e.g. ?phenolong or ?independent)

Here are the tables at the moment and a list of their current column names

```
#phenolong: long format for all the phenotypic data
names(phenolong)
```

```
## [1] "plantID"         "accession"       "treatment"       "experiment"
## [5] "meta.experiment" "facility"        "variable"        "value"
## [9] "Gene_idGene"
```

```
#phenowide: wide (typical) format for phenotypic data in phenolong
names(phenowide)
```

```
##  [1] "plantID"              "accession"            "treatment"
##  [4] "experiment"           "meta.experiment"      "facility"
##  [7] "aborted.fruits"       "alive.at.harvest"     "avg.fruit.length"
## [10] "basal.branch"         "basalfruit.length"    "biomass"
## [13] "branch.basalbranch"   "days.to.bolt"         "diameter.at.bolt"
## [16] "FruitLength1"         "FruitLength2"         "FruitLength3"
## [19] "FruitLength4"         "FruitLength5"         "FruitLength6"
## [22] "FruitLength7"         "FruitLength8"         "fruitnum"
## [25] "germinants.14d"       "germinants.31d"       "germinants.41d"
## [28] "germinants.7d"        "germinated"           "inflorescence.height"
## [31] "leaf.number"          "mainbranch"           "max.silique"
## [34] "midfruit.length"      "seeds.sown"           "ttl.maininfl.branch"
## [37] "upperfruit.length"    "Gene_idGene"
```

```
#independent: a table of indepenent data, mostly gene expression and protein expression
#indexed by both SALK_Line and Gene
names(independent)
```

```
##  [1] "id"                          "Accession_idAccession"
##  [3] "Gene_idGene"                 "ConservedGroup"
##  [5] "ProteinLevel"                "InsertLocation"
##  [7] "AverageExpress7DaySeedling"  "StdDevExpress7DaySeedling"
```

```
##  [9] "AverageExpressRosetteLeaf2"  "StdDevRosetteLeaf2"
## [11] "AverageExpressRosetteLeaf4"  "AverageExpressRosetteLeaf6"
## [13] "AverageExpressFlowerStage9"  "AverageExpressFlowerStage12"
## [15] "AverageExpressFlowerStage15" "accession"
```

```
##tdna: tdna data we know what it is.  indexed by accession
names(tdna)
```

```
##  [1] "Accession_idAccession" "Endolocus"
##  [3] "EndoArea"              "TDNAArea"
##  [5] "AreaRatio"             "RepsUsed"
##  [7] "RepsFailed"            "RatioCat"
##  [9] "TimeStamp"             "accession"
```

```
#geneont The tair10 gene ontology indexed by gene
names(geneont)
```

```
##  [1] "row_names"    "Gene_idGene"  "Relationship" "GoFunction"
##  [5] "GoNum"        "TairKW"       "Aspect"       "GoSlim"
##  [9] "EvidenceCode" "EvidenceDesc" "EvidenceWith" "Ref"
## [13] "Database"     "Date"
```

Experiments and Meta Experiments

The experiment table is sort of "subexperiment-oriented". the database now provides a way to lump all the data associated with experiments 1a1, 1a2, 1a3, etc into a meta.experiment called "1". This provides a quick way to do subsetting of the huge data set:

```
unique(phenolong[,c("experiment","meta.experiment")])
```

```
##            experiment meta.experiment
## 1           cofc-first           first
## 3400       cofc-second          second
## 5891              1a1               1
## 25076             1a2               1
## 55448            3pt1               3
## 69950          Phyt2A            phyt
## 91959          Phyt2B            phyt
## 114632           3pt2               3
## 128200            1a3               1
## 152306           3pt4               3
## 164452            1a4               1
## 170646           3pt5               3
## 181699       VT_Expt3               3
## 188009      CofC-SF12            farm
## 188777      CofC-SF13            farm
## 189545      CofC-SF14            farm
## 190313      CofC-SF15            farm
## 191081      CofC-SF16            farm
## 192617     TCTC_Expt3               3
```

```
## 204311 Benedict_Expt3              3
## 216572      CofC-SF10           farm
## 217175      CofC-SF11           farm
## 217913      CofC-SF07           farm
```

So, to subset the data to exclude farms one could do a couple of easy lines of code. I'm using dplyr throughout on this document, btw.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
unique(phenolong$experiment)
```

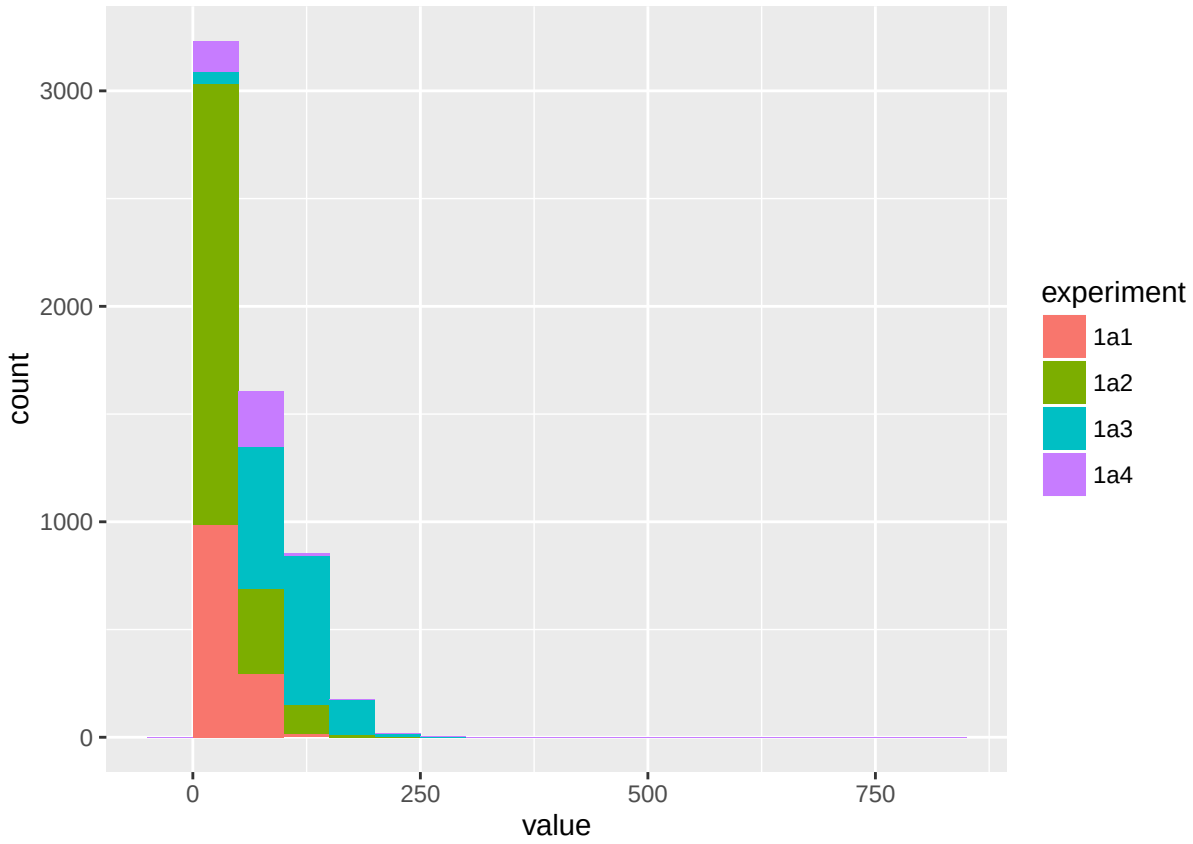
```
##  [1] "cofc-first"     "cofc-second"    "1a1"            "1a2"
##  [5] "3pt1"           "Phyt2A"         "Phyt2B"         "3pt2"
##  [9] "1a3"            "3pt4"           "1a4"            "3pt5"
## [13] "VT_Expt3"       "CofC-SF12"      "CofC-SF13"      "CofC-SF14"
## [17] "CofC-SF15"      "CofC-SF16"      "TCTC_Expt3"     "Benedict_Expt3"
## [21] "CofC-SF10"      "CofC-SF11"      "CofC-SF07"
```

```
nofarm.long <- phenolong %>% filter(meta.experiment!='farm')
unique(nofarm.long$experiment)
```

```
##  [1] "cofc-first"     "cofc-second"    "1a1"            "1a2"
##  [5] "3pt1"           "Phyt2A"         "Phyt2B"         "3pt2"
##  [9] "1a3"            "3pt4"           "1a4"            "3pt5"
## [13] "VT_Expt3"       "TCTC_Expt3"     "Benedict_Expt3"
```

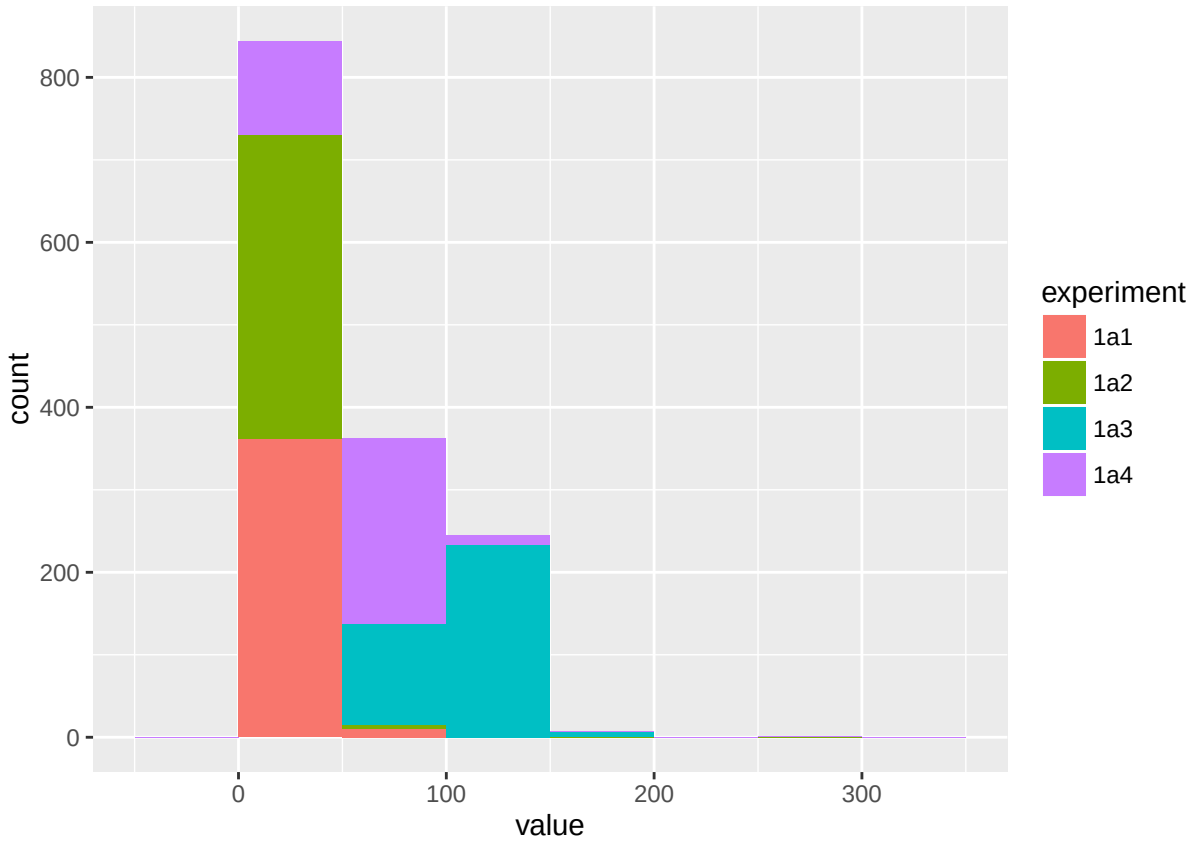
Or maybe you only want to look at the distribution of individual fruitnums for experiment1

```
exp1 <- phenolong %>% filter(meta.experiment=="1") %>% filter(variable=="fruitnum")
library(ggplot2) #might as well use ggplot to start with
ggplot(data=exp1, aes(value, fill=experiment)) + geom_histogram(binwidth=50)
```



Here's the same exercise but getting averages for every replicate within a growth chamber within an experiment

```
exp1mn <- phenolong %>% filter(meta.experiment=="1") %>%
  filter(variable=="fruitnum") %>%
  group_by(experiment,accession) %>%
  summarise(value=mean(value,na.rm=T))
ggplot(data=exp1mn, aes(value, fill=experiment)) + geom_histogram(binwidth=50)
```

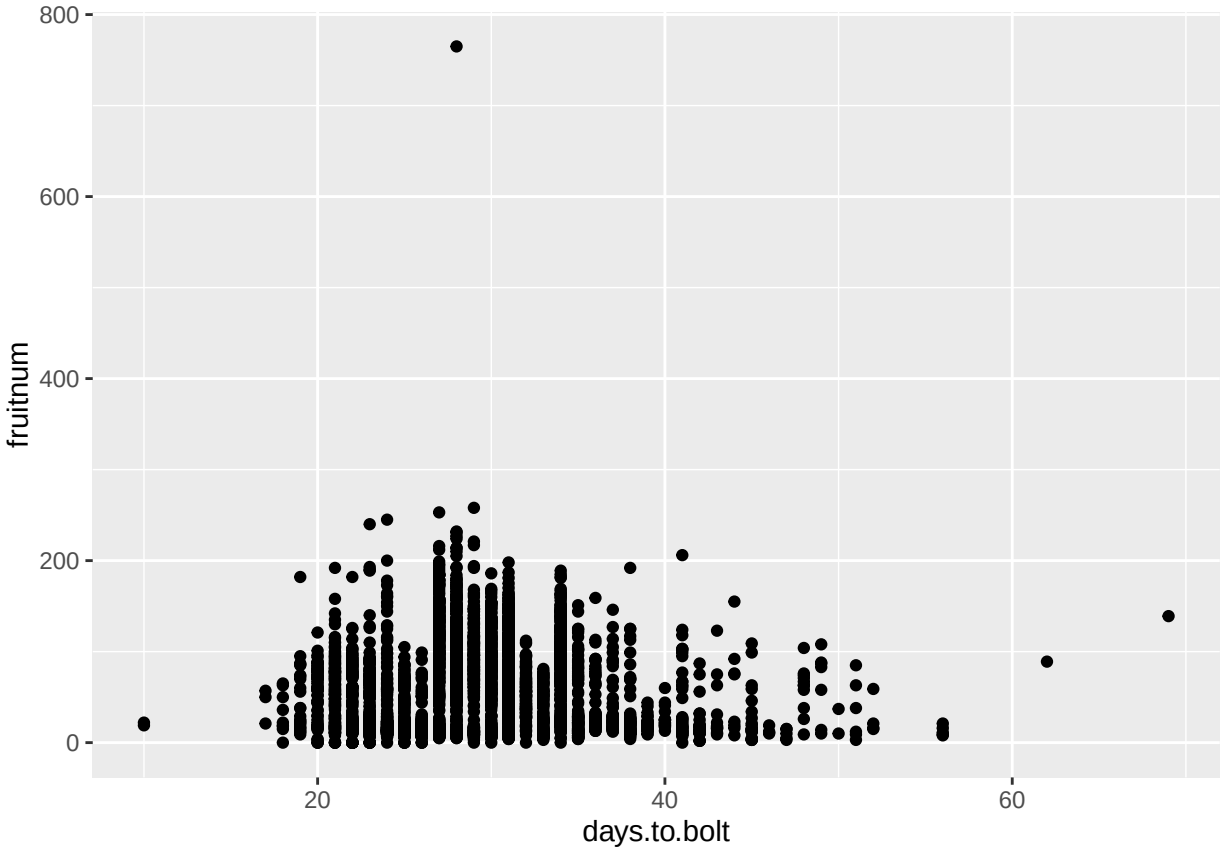


I've done all of the above with the long format, but similar approaches could be used with the wide format in phenowide

For example, it is arguably easier to compare phenotypes when they are in the wide format

```
exp1 <- phenowide %>% filter(meta.experiment=="1")
ggplot(data=exp1,aes(x=days.to.bolt,y=fruitnum))+geom_point()
```

```
## Warning: Removed 1123 rows containing missing values (geom_point).
```



might be nice to see whats with that point at 765 fruits

Merging with other data

The information in the other tables can be compared to phenotypes through merging.

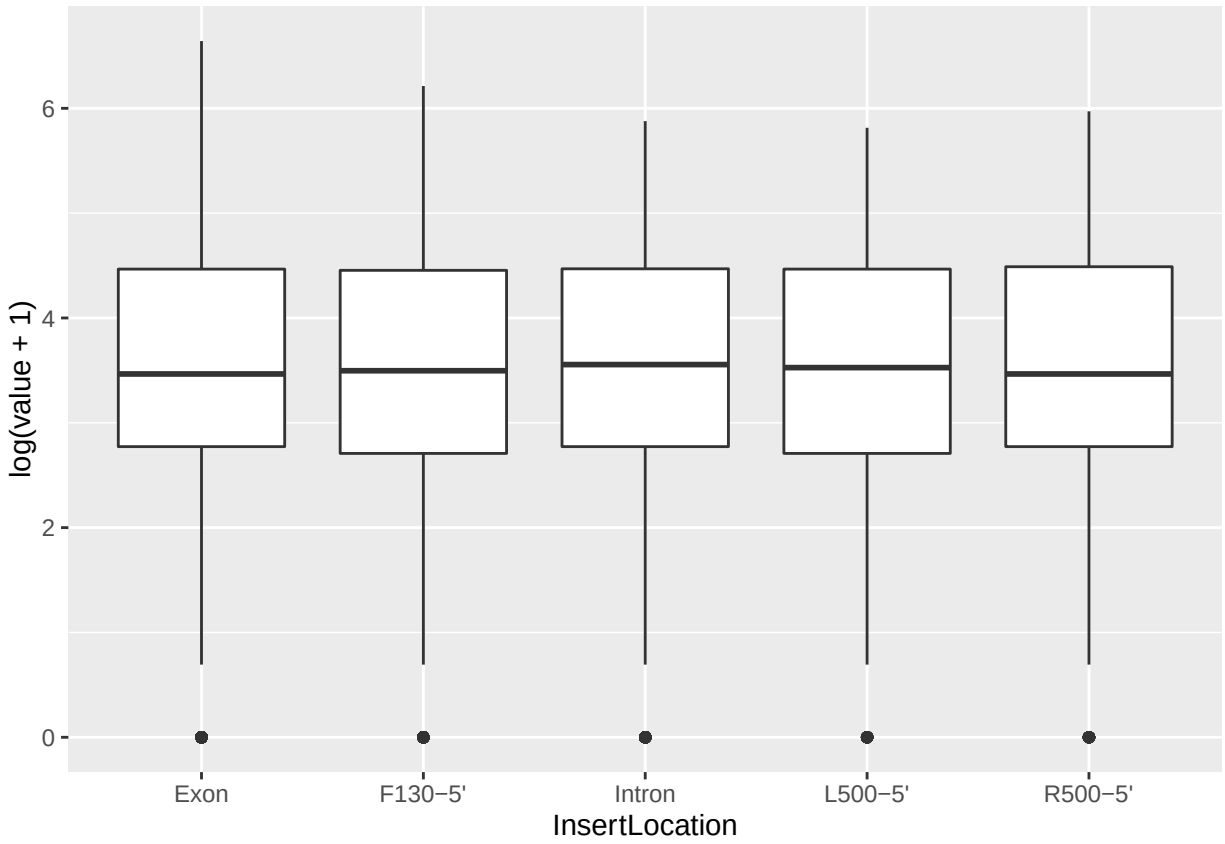
Say you wanted to look at how insertion location might influence phenotype. You could merge in the 'independent' table and look at the result

```
newdf <- left_join(phenolong,independent) %>% filter(variable=="fruitnum")
```

```
## Joining by: c("accession", "Gene_idGene")
```

```
ggplot(data=newdf, aes(x=InsertLocation, y=log(value+1)))+geom_boxplot()
```

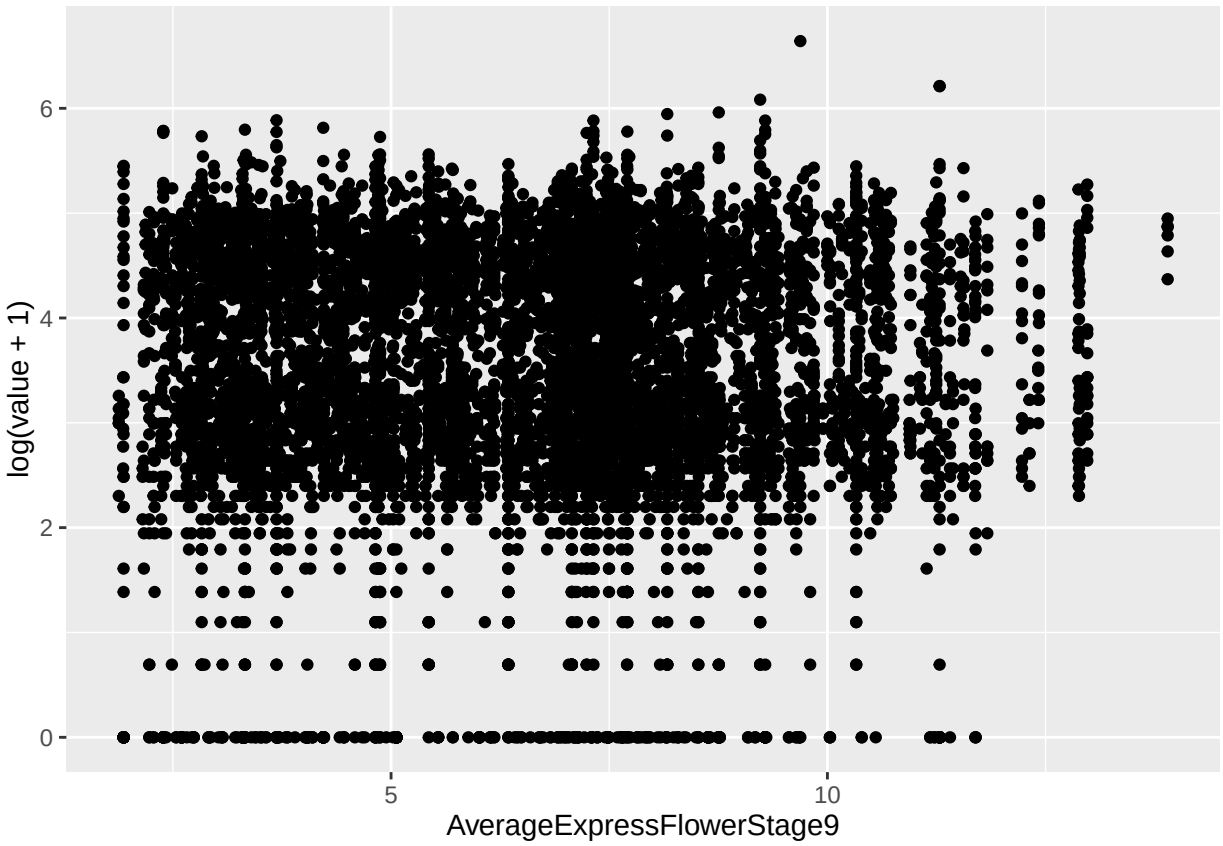
```
## Warning: Removed 5471 rows containing non-finite values (stat_boxplot).
```



or you could look at expression at flower stage 9

```
ggplot(data=newdf, aes(x=AverageExpressFlowerStage9, y=log(value+1)))+geom_point()
```

```
## Warning: Removed 7840 rows containing missing values (geom_point).
```



Correcting phenotypes

Everything so far has been performed on uncorrected phenotypes.

We can correct phenotypes based on the Treatment, Experiment, and Facility combinations

Here's an example correcting for phytometers

```
library(reshape)
```

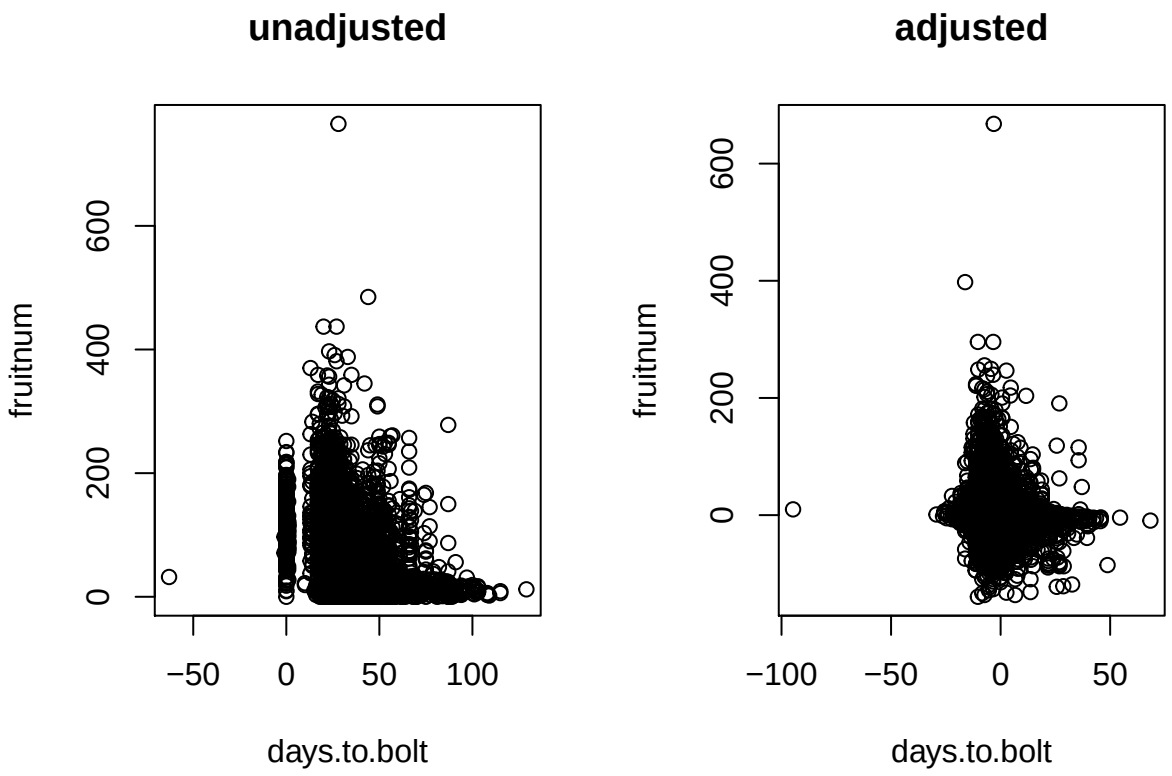
```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```

```
plotdf = phenolong %>% filter(variable %in% c("fruitnum","days.to.bolt")) %>% cast(fun.aggregate=mean)
```

```
plotadj <- phytcorrect(phenolong,c("fruitnum","days.to.bolt"),c("plantID","treatment","facility","experi
```

```
## Joining by: c("treatment", "experiment", "facility", "variable")
```

```
par(mfrow=c(1,2))
plot(fruitnum~days.to.bolt,data=plotdf,main="unadjusted")
plot(fruitnum~days.to.bolt,data=plotadj,main="adjusted")
```



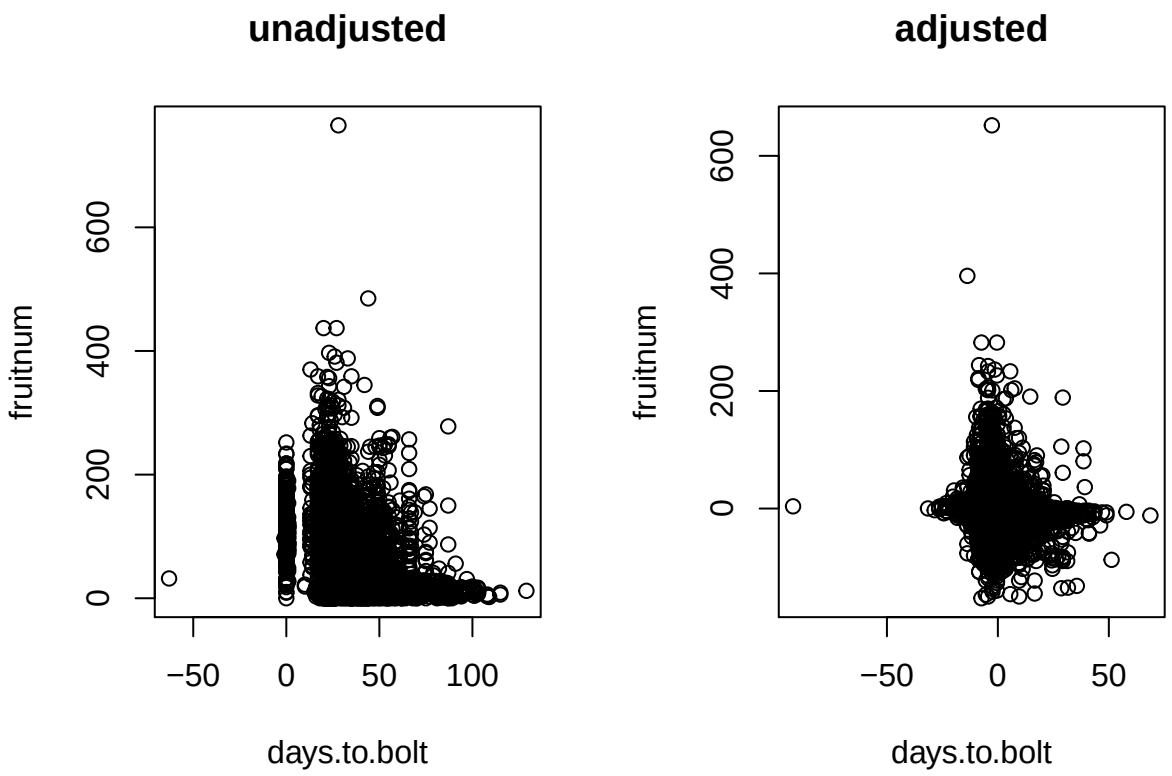

```
par(mfrow=c(1,1))
```

And here correcting by all plant means

```
plotadj <- allcorrect(phenolong,c("fruitnum","days.to.bolt"),c("plantID","treatment","facility","experi
```

```
## Joining by: c("treatment", "experiment", "facility", "variable")
```

```
par(mfrow=c(1,2))
plot(fruitnum~days.to.bolt,data=plotdf,main="unadjusted")
plot(fruitnum~days.to.bolt,data=plotadj,main="adjusted")
```




```
par(mfrow=c(1,1))
```