Hypothesis testing and basic linear models
Week 4, Lecture 07

Richard E.W. Berl

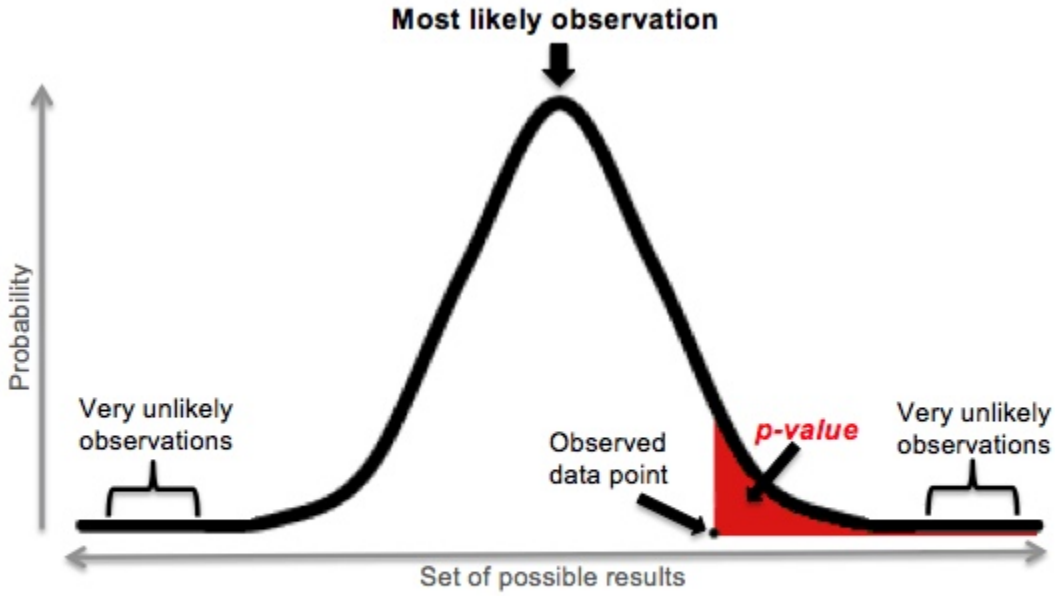
Spring 2019

Hypothesis testing

What is a *p*-value?

"The probability of getting the observed result, or a more extreme result, if the null hypothesis is true."

Source: McDonald, Basic concepts of hypothesis testing



Source: Pomeroy, R. (2016). The biggest myth about p-values. RealClearScience.

Now let's get some data!

Go to the page for the PanTHERIA database (hosted by the Ecological Society of America) at: http://esapubs.org/archive/ecol/E090/184/

Near the bottom, download the text file with "WR05" in the name (referring to the 2005 edition of a mammal classification book) and put it in your `/data` folder.

Metadata and descriptions of all the variables are available here: http://esapubs.org/archive/ecol/E090/184/metadata.htm

```
pan = read.table("./data/PanTHERIA_1-0_WR05_Aug2008.txt", header=T, sep="\t",
                 na.strings=c("-999","-999.00"), stringsAsFactors=F)
```

```
str(pan)
```

```
## 'data.frame':    5416 obs. of  55 variables:
##  $ MSW05_Order                 : chr  "Artiodactyla" "Carnivora" "Carnivora" "Carnivora" ...
##  $ MSW05_Family                : chr  "Camelidae" "Canidae" "Canidae" "Canidae" ...
##  $ MSW05_Genus                 : chr  "Camelus" "Canis" "Canis" "Canis" ...
##  $ MSW05_Species               : chr  "dromedarius" "adustus" "aureus" "latrans" ...
##  $ MSW05_Binomial              : chr  "Camelus dromedarius" "Canis adustus" "Canis aureus" "Canis la
##  $ X1.1_ActivityCycle          : num  3 1 2 2 2 2 NA 2 3 NA ...
##  $ X5.1_AdultBodyMass_g        : num  492714 10392 9659 11989 31757 ...
##  $ X8.1_AdultForearmLen_mm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X13.1_AdultHeadBodyLen_mm   : num  NA 745 828 872 1055 ...
##  $ X2.1_AgeatEyeOpening_d      : num  NA NA 7.5 11.9 14 ...
##  $ X3.1_AgeatFirstBirth_d      : num  1652 NA NA 365 548 ...
##  $ X18.1_BasalMetRate_mLO2hr   : num  40293 NA NA 3699 11254 ...
##  $ X5.2_BasalMetRateMass_g     : num  407000 NA NA 10450 33100 ...
##  $ X6.1_DietBreadth            : num  3 6 6 1 1 3 2 2 NA NA ...
##  $ X7.1_DispersalAge_d         : num  NA 330 NA 255 180 ...
##  $ X9.1_GestationLen_d         : num  386.5 65 61.2 61.7 63.5 ...
##  $ X12.1_HabitatBreadth        : num  1 1 1 1 1 NA NA 1 1 NA ...
##  $ X22.1_HomeRange_km2         : num  196.32 1.01 2.95 18.88 159.86 ...
##  $ X22.2_HomeRange_Indiv_km2   : num  NA 1.01 3.13 19.91 43.13 ...
##  $ X14.1_InterbirthInterval_d  : num  614 NA 365 365 365 ...
##  $ X15.1_LitterSize            : num  0.98 4.5 3.74 5.72 4.98 1.22 1 1.22 1.01 NA ...
##  $ X16.1_LittersPerYear        : num  1 NA NA NA 2 1 1 1 NA NA ...
##  $ X17.1_MaxLongevity_m        : num  480 137 192 262 354 ...
##  $ X5.3_NeonateBodyMass_g      : num  36751 NA 212 200 412 ...
##  $ X13.2_NeonateHeadBodyLen_mm : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X21.1_PopulationDensity_n.km2: num  0.98 0.74 0.22 0.25 0.01 0.54 NA 0.75 4.89 NA ...
##  $ X10.1_PopulationGrpSize     : num  11 NA NA NA NA NA NA 21 NA NA ...
##  $ X23.1_SexualMaturityAge_d   : num  1948 250 371 373 679 ...
##  $ X10.2_SocialGrpSize         : num  10 NA NA NA NA 40 110 40 2.05 NA ...
##  $ X24.1_TeatNumber            : int  NA 8 8 8 9 NA NA NA NA NA ...
##  $ X12.2_Terrestriality        : num  1 1 1 1 1 NA NA 1 2 NA ...
##  $ X6.2_TrophicLevel           : int  1 2 2 3 3 1 1 1 NA NA ...
##  $ X25.1_WeaningAge_d          : num  389.4 52.9 61.3 43.7 44.8 ...
##  $ X5.4_WeaningBodyMass_g      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X13.3_WeaningHeadBodyLen_mm : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ References                  : chr  "511;543;719;1274;1297;1594;1654;1822;1848;2655;3044" "542;543
##  $ X5.5_AdultBodyMass_g_EXT    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X16.2_LittersPerYear_EXT    : num  NA NA 1.1 1.1 NA NA NA NA 1.05 NA ...
##  $ X5.6_NeonateBodyMass_g_EXT  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X5.7_WeaningBodyMass_g_EXT  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ X26.1_GR_Area_km2           : num  NA 10581413 25739527 17099094 50803440 ...
##  $ X26.2_GR_MaxLat_dd          : num  NA 16.7 47 71.4 83.3 ...
##  $ X26.3_GR_MinLat_dd          : num  NA -28.73 -4.71 8.02 11.48 ...
##  $ X26.4_GR_MidRangeLat_dd     : num  NA -6 21.1 39.7 47.4 ...
##  $ X26.5_GR_MaxLong_dd         : num  NA 43.5 108.5 -67.1 179.7 ...
##  $ X26.6_GR_MinLong_dd         : num  NA -17.5 -17.1 -168.1 -171.8 ...
```

```
##  $ X26.7_GR_MidRangeLong_dd     : num  NA 13 45.7 -117.6 3.9 ...
##  $ X27.1_HuPopDen_Min_n.km2     : int  NA 0 0 0 0 1 0 1 0 NA ...
##  $ X27.2_HuPopDen_Mean_n.km2    : num  NA 35.2 79.3 27.3 37.9 ...
##  $ X27.3_HuPopDen_5p_n.km2      : num  NA 1 0 0 0 8 0 4 0 NA ...
##  $ X27.4_HuPopDen_Change        : num  NA 0.14 0.1 0.06 0.04 0.09 0.05 0.11 0.05 NA ...
##  $ X28.1_Precip_Mean_mm         : num  NA 90.8 44.6 53 34.8 ...
##  $ X28.2_Temp_Mean_01degC       : num  NA 236.51 217.23 58.18 4.82 ...
##  $ X30.1_AET_Mean_mm            : num  NA 923 438 503 313 ...
##  $ X30.2_PET_Mean_mm            : num  NA 1534 1359 728 561 ...
```

Column names are messy. Let's try to clean them up.

```
library(stringr)
```

Loop through each column name, check if it has an underscore, and try to split it into two pieces using the underscore as a separator:

```
for (i in 1:length(colnames(pan))) {
  if (grepl("_", colnames(pan)[i])) {
    colnames(pan)[i] = str_split_fixed(colnames(pan)[i], "_", n=2)[2]
  }
}
```

```
colnames(pan)
```

```
##  [1] "Order"                  "Family"
##  [3] "Genus"                  "Species"
##  [5] "Binomial"               "ActivityCycle"
##  [7] "AdultBodyMass_g"        "AdultForearmLen_mm"
##  [9] "AdultHeadBodyLen_mm"    "AgeatEyeOpening_d"
## [11] "AgeatFirstBirth_d"      "BasalMetRate_mLO2hr"
## [13] "BasalMetRateMass_g"     "DietBreadth"
## [15] "DispersalAge_d"         "GestationLen_d"
## [17] "HabitatBreadth"         "HomeRange_km2"
## [19] "HomeRange_Indiv_km2"    "InterbirthInterval_d"
## [21] "LitterSize"             "LittersPerYear"
## [23] "MaxLongevity_m"         "NeonateBodyMass_g"
## [25] "NeonateHeadBodyLen_mm"  "PopulationDensity_n.km2"
## [27] "PopulationGrpSize"      "SexualMaturityAge_d"
## [29] "SocialGrpSize"          "TeatNumber"
## [31] "Terrestriality"         "TrophicLevel"
## [33] "WeaningAge_d"           "WeaningBodyMass_g"
## [35] "WeaningHeadBodyLen_mm"  "References"
## [37] "AdultBodyMass_g_EXT"    "LittersPerYear_EXT"
## [39] "NeonateBodyMass_g_EXT"  "WeaningBodyMass_g_EXT"
## [41] "GR_Area_km2"            "GR_MaxLat_dd"
## [43] "GR_MinLat_dd"           "GR_MidRangeLat_dd"
## [45] "GR_MaxLong_dd"          "GR_MinLong_dd"
## [47] "GR_MidRangeLong_dd"     "HuPopDen_Min_n.km2"
## [49] "HuPopDen_Mean_n.km2"    "HuPopDen_5p_n.km2"
## [51] "HuPopDen_Change"        "Precip_Mean_mm"
## [53] "Temp_Mean_01degC"       "AET_Mean_mm"
## [55] "PET_Mean_mm"
```

Statistical tests

1 Nominal Variable

Exact binomial test

```
?binom.test
```

Classic example: toss a coin a bunch of times and test whether the probability of getting a heads is different from 0.5.

"Is an artiodactyl species more likely to be an herbivore than an omnivore?"

```
table(pan$TrophicLevel[pan$Order == "Artiodactyla"])
```

```
##
##   1   2
## 139  25
```

```
binom.test(table(pan$TrophicLevel[pan$Order == "Artiodactyla"]),
           alternative="greater")
```

```
##
##  Exact binomial test
##
## data:  table(pan$TrophicLevel[pan$Order == "Artiodactyla"])
## number of successes = 139, number of trials = 164, p-value <
## 2.2e-16
## alternative hypothesis: true probability of success is greater than 0.5
## 95 percent confidence interval:
##  0.7936443 1.0000000
## sample estimates:
## probability of success
##               0.847561
```

2 Nominal Variables

Fisher's exact test

```
?fisher.test
```

Tests a contingency table to ask whether the values of each category differ across groups.

"Does the time of day that primate species are active differ across families?"

```
table(pan$ActivityCycle[pan$Order == "Primates"],
      pan$Family[pan$Order == "Primates"])
```

```
##
##     Aotidae Atelidae Cebidae Cercopithecidae Cheirogaleidae Daubentoniidae
##   1       8        0       0               0              6              1
##   2       0        0       0               1              2              0
##   3       0       18      42              95              0              0
##
##     Galagidae Hominidae Hylobatidae Indriidae Lemuridae Lepilemuridae
##   1         5         0           0         1         0             7
##   2         0         0           0         1        12             0
##   3         0         5          11         7         0             0
##
```

```
##     Lorisidae Pitheciidae Tarsiidae
##   1         4           0         3
##   2         0           0         0
##   3         0          22         0
```

Notice it's still a 2x2 table, even though each variable has a bunch of categories.

```
fisher.test(table(pan$ActivityCycle[pan$Order == "Primates"],
                  pan$Family[pan$Order == "Primates"]))
```

```
## Error in fisher.test(table(pan$ActivityCycle[pan$Order == "Primates"], : FEXACT error 7(location). LI
##   (pastp=7.78322, ipn_0:=ipoin[itp=40]=86, stp[ipn_0]=6.44572).
## Increase workspace or consider using 'simulate.p.value=TRUE'
```

It doesn't want to do it, because it's a big table, but we can force it to simulate a p-value, which will be good enough for now (see the McDonald page for alternatives).

```
fisher.test(table(pan$ActivityCycle[pan$Order == "Primates"],
                  pan$Family[pan$Order == "Primates"]),
            simulate.p.value=T)
```

```
##
##  Fisher's Exact Test for Count Data with simulated p-value (based
##  on 2000 replicates)
##
## data:
## p-value = 0.0004998
## alternative hypothesis: two.sided
```

1 Measurement Variable

One-sample t-test

```
?t.test
```

Is the mean of a set of values different from some value I specify?

"Does the forearm length of fruit bats (Family Pteropodidae) tend to be greater than 100 mm?"

```
na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae"])[1:20]
```

```
##  [1]  40.96  45.50  56.00  64.21  73.52  71.00  76.94 122.55 105.74 127.50
## [11] 122.37 115.64 114.76 107.25 148.84  80.00 138.69 110.83 115.10 116.50
```

```
mean(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae"], na.rm=T)
```

```
## [1] 100.1316
```

```
t.test(na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae"]),
       mu=100, alternative="greater")
```

```
##
##  One Sample t-test
##
## data:  na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae"])
## t = 0.042288, df = 164, p-value = 0.4832
## alternative hypothesis: true mean is greater than 100
## 95 percent confidence interval:
##  94.98233      Inf
## sample estimates:
```

```
## mean of x
##  100.1316
```

1 Nominal Variable and 1 Measurement Variable

Two-sample t-test

```
?t.test
```

Do the means of two sets of values differ?

"Does the average forearm length of fruit bats (Family Pteropodidae) differ between species that have one litter of offspring per year and species that have two?"

```
na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae" & pan$LittersPerYear == 1])[1:10]
```

```
##  [1] 105.74 138.69 127.72 120.50  50.56  77.00  65.36 165.99 177.97 132.00
```

```
na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae" & pan$LittersPerYear == 2])[1:10]
```

```
##  [1]  64.21  71.00  76.94 107.25  80.00 116.50  69.00  85.27  73.23  80.66
```

```
t.test(na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae" & pan$LittersPerYear == 1]),
       na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae" & pan$LittersPerYear == 2]))
```

```
##
##  Welch Two Sample t-test
##
## data:  na.omit(pan$AdultForearmLen_mm[pan$Family == "Pteropodidae" &  and na.omit(pan$AdultForearmLe
## t = 5.9168, df = 25.131, p-value = 3.487e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  39.08336 80.80292
## sample estimates:
## mean of x mean of y
##  137.6268   77.6837
```

Note that if you have paired values (like pre- and post-test values from the same individuals), then a paired t-test is more powerful (better chance of detecting a significant effect with a smaller sample size) and you should use that instead. You can do this in R by setting the `t.test()` argument `paired=TRUE`.

2 Measurement Variables

Correlation test

```
?cor.test
```

Are two variables correlated (positively or negatively)?

"Is wild mammal population density associated with human population density?"

```
pan[complete.cases(pan$PopulationDensity_n.km2, pan$HuPopDen_Mean_n.km2),
    c("PopulationDensity_n.km2", "HuPopDen_Mean_n.km2")][1:20,]
```

```
##    PopulationDensity_n.km2 HuPopDen_Mean_n.km2
## 2                     0.74               35.20
## 3                     0.22               79.29
## 4                     0.25               27.27
## 5                     0.01               37.87
## 6                     0.54              152.67
```

```
## 8                    0.75             139.21
## 9                    4.89               1.07
## 19                  11.86              53.45
## 21                  62.32              39.68
## 22                   8.85               3.51
## 24                   0.06               1.83
## 27                   2.39             313.30
## 29                   0.74              30.81
## 30                   1.20              99.87
## 41                  13.49               2.79
## 42                   3.39               1.62
## 44                  19.06              55.14
## 47                   8.00               3.25
## 48                  18.43               2.40
## 54                  13.03              47.74
```

```
cor.test(pan[complete.cases(pan$PopulationDensity_n.km2,
                            pan$HuPopDen_Mean_n.km2),
             "PopulationDensity_n.km2"],
         pan[complete.cases(pan$PopulationDensity_n.km2,
                            pan$HuPopDen_Mean_n.km2),
             "HuPopDen_Mean_n.km2"])
```

```
##
##  Pearson's product-moment correlation
##
## data:  pan[complete.cases(pan$PopulationDensity_n.km2, pan$HuPopDen_Mean_n.km2),  and pan[complete.ca
## t = -0.077896, df = 891, p-value = 0.9379
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.06820173  0.06300495
## sample estimates:
##          cor
## -0.002609622
```

Regression

Correlation and regression can both examine how two variables are associated.

So what's the difference between correlation and regression?

1. Causality: Correlation measures association, but regression measures one variable's ability to predict the value of another variable (which is "predictive," not necessarily always causal).

2. Estimation: Regression estimates parameter values (e.g. the `m` and `b` of a `y = mx + b` equation) so that you can create a model of your data and plot a line of best fit through it.

3. Prediction: Using the model you estimate, you can plug in new values to predict what the response should be.

2 Measurement Variables

Linear regression

```
?lm
```

Does one variable predict another (positively or negatively)?

"Does litter size predict home range in canids?"

```
panCanid = pan[pan$Family == "Canidae",]
panCanid = panCanid[complete.cases(panCanid$LitterSize,
                                   panCanid$HomeRange_km2),]

panCanidModel = lm(HomeRange_km2 ~ LitterSize, data=panCanid)
summary(panCanidModel)
```

```
##
## Call:
## lm(formula = HomeRange_km2 ~ LitterSize, data = panCanid)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -115.07  -39.74  -16.59   26.48  254.07
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -85.60      50.20  -1.705   0.1054
## LitterSize     28.07      10.83   2.592   0.0184 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 80.41 on 18 degrees of freedom
## Multiple R-squared:  0.2718, Adjusted R-squared:  0.2314
## F-statistic:  6.72 on 1 and 18 DF,  p-value: 0.0184
```

Response (Y variable) comes first, followed by predictors (X variables). Read the ~ as "as a function of" or "as a result of."

So, from this, our model equation would look something like:

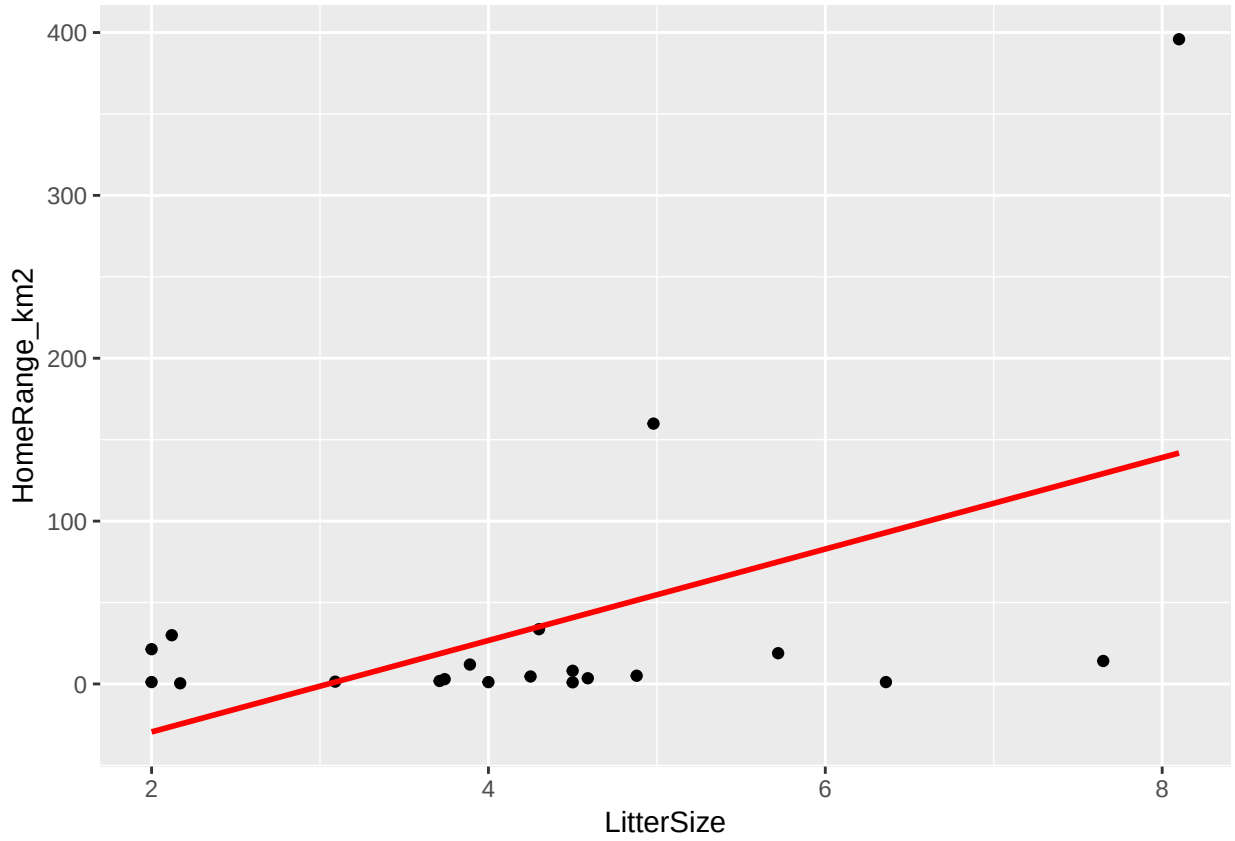
$$Y = b + m * X$$

$$HomeRange\_km2 = -85.60 + 28.07 * LitterSize$$

We can plot our regression line:

```
library(ggplot2)

ggplot(panCanid, aes(x=LitterSize, y=HomeRange_km2)) +
  geom_point() +
  geom_smooth(method="lm", se=F, color="red")
```



This re-runs the `lm()` function with the specified variables to plot the line.

But if we want to make sure we're using our own model estimates, we can `fortify()` our model results, which turns them into a data frame:

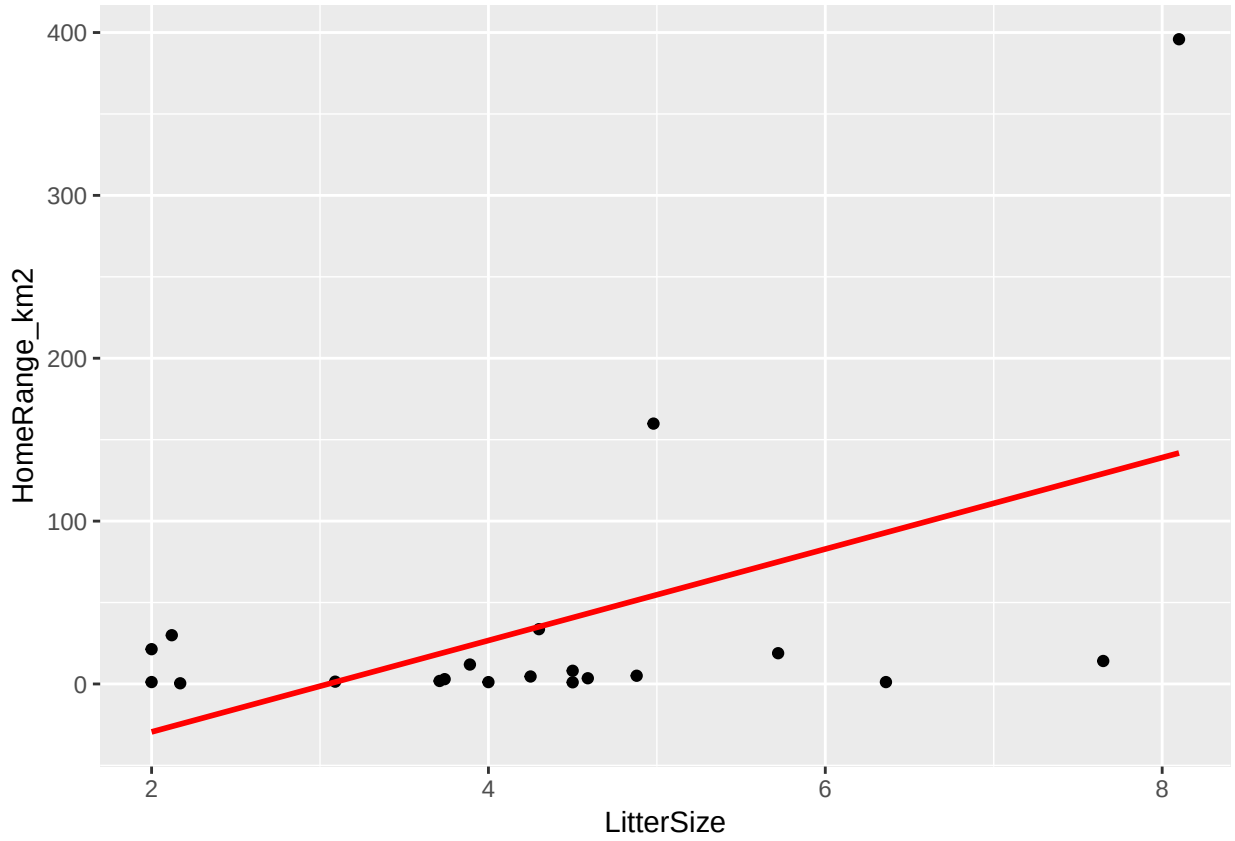
```
fortify(panCanidModel)
```

```
##      HomeRange_km2 LitterSize       .hat   .sigma      .cooksd    .fitted
## 2             1.01       4.50 0.05053979 82.14831 6.842384e-03  40.737331
## 3             2.95       3.74 0.05626123 82.63931 1.321991e-03  19.400799
## 4            18.88       5.72 0.08517500 81.50889 2.477626e-02  74.988081
## 5           159.86       4.98 0.05772333 78.41785 5.611221e-02  54.213036
## 29           11.93       3.89 0.05347216 82.69005 6.298555e-04  23.611957
## 193           1.41       3.09 0.07778012 82.74129 4.691101e-07   1.152449
## 882          21.36       2.00 0.14827048 81.65682 4.080140e-02 -29.448630
## 1132         33.64       4.30 0.05001372 82.74049 9.418158e-06  35.122454
## 2224          5.05       4.88 0.05553743 81.92856 1.034593e-02  51.405598
## 2230        395.88       8.10 0.30816788 36.84250 3.214091e+00 141.805116
## 3133          1.15       6.36 0.12493847 79.24366 1.063425e-01  92.955687
## 3249          1.13       4.00 0.05194566 82.49580 2.922118e-03  26.700139
## 5312          1.83       3.71 0.05691700 82.63577 1.384868e-03  18.558567
## 5313          0.42       2.17 0.13443944 82.48226 8.739549e-03 -24.675985
## 5373          1.17       2.00 0.14827048 82.34912 1.481739e-02 -29.448630
## 5377         14.10       7.65 0.25025054 76.20518 4.558565e-01 129.171643
## 5378          8.10       4.50 0.05053979 82.34155 4.618042e-03  40.737331
## 5380         29.93       2.12 0.13839855 81.43678 4.522661e-02 -26.079704
## 5381          4.59       4.25 0.05010895 82.42318 3.643844e-03  33.718735
## 5382          3.50       4.59 0.05124998 82.14676 6.961768e-03  43.264026
```

```
##           .resid    .stdresid
## 2     -39.727331 -0.507037867
## 3     -16.450799 -0.210596178
## 4     -56.108081 -0.729534421
## 5     105.646964  1.353496665
## 29    -11.681957 -0.149326981
## 193     0.257551  0.003335303
## 882    50.808630  0.684661686
## 1132   -1.482454 -0.018915249
## 2224  -46.355598 -0.593197340
## 2230  254.074884  3.798837911
## 3133  -91.805687 -1.220505316
## 3249  -25.570139 -0.326592240
## 5312  -16.728567 -0.214226495
## 5313   25.095985  0.335463211
## 5373   30.618630  0.412595319
## 5377 -115.071643 -1.652721284
## 5378  -32.637331 -0.416548565
## 5380   56.009704  0.750411455
## 5381  -29.128735 -0.371684215
## 5382  -39.764026 -0.507696111
```

We can then use the `.fitted` values (the result of plugging each of those `LitterSize` values into the model) as the values for our line:

```
ggplot(panCanid, aes(x=LitterSize, y=HomeRange_km2)) +
  geom_point() +
  geom_line(data=fortify(panCanidModel), aes(x=LitterSize, y=.fitted),
            color="red", size=1)
```



Finally, we can use our model to predict new responses.

```
?predict
```

```
predict(panCanidModel, newdata=data.frame(LitterSize=1:10))
```

```
##          1          2          3          4          5          6
## -57.523015 -29.448630  -1.374246  26.700139  54.774524  82.848908
##          7          8          9         10
## 110.923293 138.997678 167.072062 195.146447
```

```
range(panCanid$LitterSize)
```

```
## [1] 2.0 8.1
```

Note we can predict the results of values outside the range of our data.

If we want to use a different "link function" (i.e. if we think our data fit a different distribution), we can use a generalized linear model with `glm()`. We can also do basic linear regression with `glm()`, because `lm()` is a special case of `glm()`.

We also see, with our plot, that we probably should have checked for and removed outliers first. Outliers can have a huge effect on the parameters of your model and its accuracy, because the model fitting procedure has to take those outliers into account when it's trying to find the line that fits everything as closely as possible.

3+ Measurement Variables

Multiple regression

Do any of these variables predict another (positively or negatively)?

First, let's make sure we're working with the complete cases for our new variable, `DispersalAge_d`.

```
panCanid2 = pan[pan$Family == "Canidae",]
panCanid2 = panCanid2[complete.cases(panCanid2$LitterSize,
                                     panCanid2$DispersalAge_d,
                                     panCanid2$HomeRange_km2),]
```

Same function, we just add predictor variables with `+`:

```
panCanidModel2 = lm(HomeRange_km2 ~ LitterSize + DispersalAge_d, data=panCanid2)
summary(panCanidModel2)
```

```
##
## Call:
## lm(formula = HomeRange_km2 ~ LitterSize + DispersalAge_d, data = panCanid2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -109.25  -47.84  -13.82   36.70  160.69
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -206.284     78.959  -2.613   0.0227 *
## LitterSize       24.217     13.717   1.765   0.1029
## DispersalAge_d    0.629      0.337   1.866   0.0866 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 82.55 on 12 degrees of freedom
## Multiple R-squared:  0.4756, Adjusted R-squared:  0.3882
## F-statistic: 5.442 on 2 and 12 DF,  p-value: 0.0208
```

Now that we take the effect of dispersal age into account, litter size is no longer significant. But dispersal age isn't significant either ("almost significant" or "marginally significant" don't count–it's either below your alpha value or it's not), which is incredibly frustrating. There must be something going on.

Let's check for interactions. We can do so in the model by changing the `+` to a `*` to specify it should include an interaction term between our main effects:

```
panCanidModel3 = lm(HomeRange_km2 ~ LitterSize * DispersalAge_d, data=panCanid2)
summary(panCanidModel3)
```

```
##
## Call:
## lm(formula = HomeRange_km2 ~ LitterSize * DispersalAge_d, data = panCanid2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.081 -20.475 -16.314   8.685 133.484
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               312.64756  112.58178   2.777  0.01800 *
## LitterSize                -64.69160   19.33256  -3.346  0.00652 **
## DispersalAge_d             -1.92514    0.54285  -3.546  0.00458 **
## LitterSize:DispersalAge_d   0.42661    0.08471   5.036  0.00038 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 47.42 on 11 degrees of freedom
## Multiple R-squared:  0.8414, Adjusted R-squared:  0.7981
## F-statistic: 19.45 on 3 and 11 DF,  p-value: 0.000105
```

Now everything is significant, including our interaction term.

What is an interaction?

"The effect of Factor A is different, depending on which level of Factor B we're talking about."

Source: Navarro, Section 16.2.1

Significant interactions are generally bad news, because it can be very difficult to explain them in a way that is biologically (or physically, or socially...) meaningful.

Sometimes it is helpful to visualize the effects of the two predictors to see what the interaction is. But to do so with measurement variables, we have to make them discrete so we can see the change according to different levels. There are different ways to do this "binning." For `LitterSize`, let's split it by `Low` and `High` based on the median value:

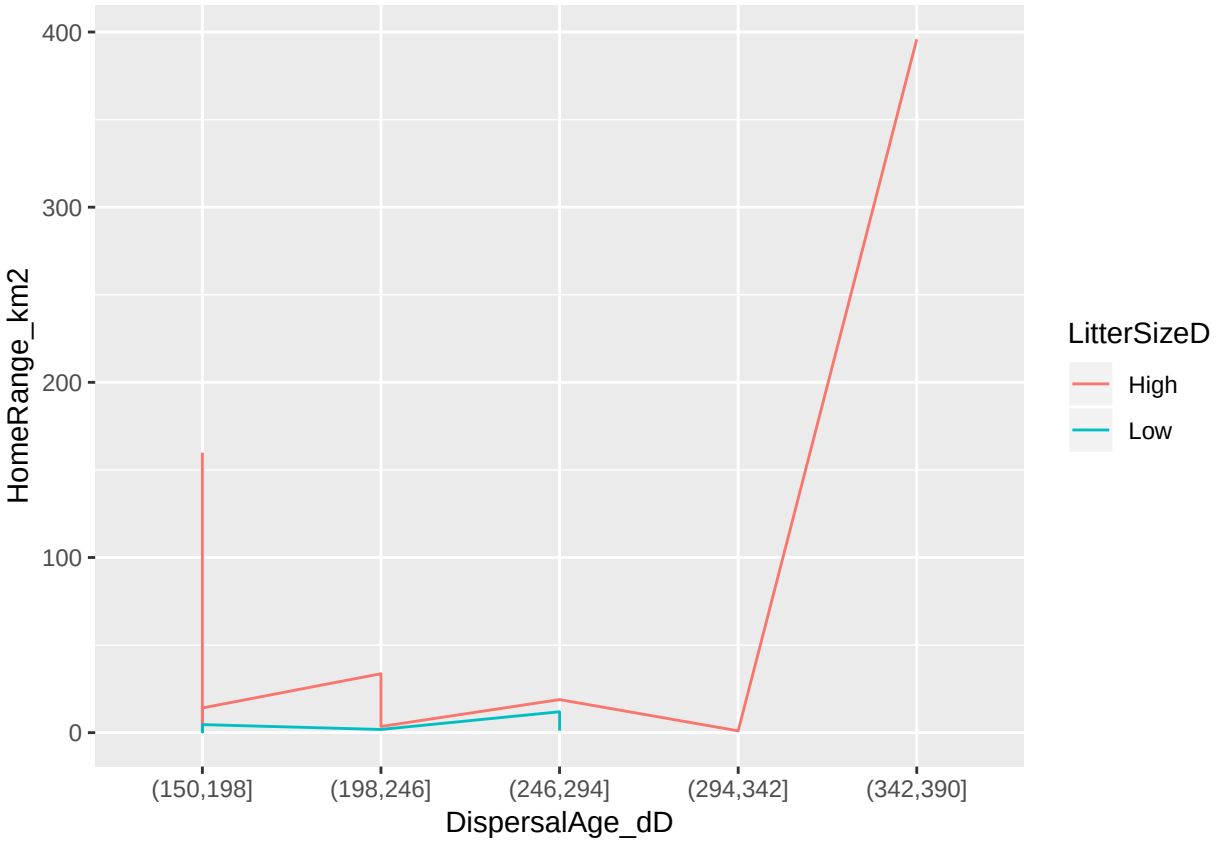
```
panCanid2$LitterSizeD = ifelse(panCanid2$LitterSize < median(panCanid2$LitterSize),
                               "Low", "High")
```

And for `DispersalAge_d`, we'll tell R we want to split it into 5 intervals using the `cut()` function:

```
panCanid2$DispersalAge_dD = cut(panCanid2$DispersalAge_d, 5)
```

You could also use quantiles or `cut2()` from the `Hmisc` package to cut by intervals of equal numbers of values rather than equal spacing.

```
ggplot(panCanid2, aes(x=DispersalAge_dD, y=HomeRange_km2, group=LitterSizeD)) +
  geom_line(aes(color=LitterSizeD))
```



Like I mentioned before, difficult to interpret. Usually, in an interaction plot, the lines will cross each other (showing that the relationships are different for each level of the factor), instead of being parallel (which would show similar effects); here, they do cross in the first level of dispersal age, though it's difficult to see.

(pdf / Rmd)