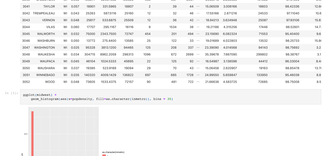
```
library(dplyrverbs)
library(modelr)
library(broom.helpers) # ha.ness)

# questions: what variables that randomly generated numbers are the same everytime you run the scenario
# concern reproductive results
set.seed(1111)
```

```
>> Attaching packages ------------------------------------------ tidyverse 1.3.0 --
ggplot3 3.3.2     v purrr   0.3.4
v tibble  3.0.3     v dplyr   1.0.2
v tidyr   1.1.2     v stringr 1.4.0
v readr   1.3.1     v forcats 1.4.0
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
```

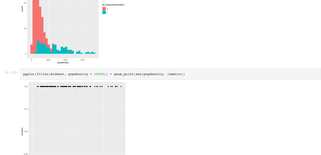
```
# Let's create a model that predicts whether a Millennial Country is in a metro area based on the county's population density using logistic regression.
millmet
```

```
# tibble: 437 x 20
   FIP   county state area popdiv2 popdiv3 popdiv4 popmetro2 popmetro3 popmetro4 popwhite popblack popindian popasian popnhpi popsamerican popmore popnonhispan popmetrohispan popmetrolowpov
   <dbl> <chr>  <chr> <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
 1 001  ADAMS  IL    0.052  69350  125.9870  63817  7552    88   249  19.6725  4.350859  63628  38.27679   12.1
 2 002  ALEXANDER IL    0.024  16000  150.5625  7554   3498   76    16  11.2433  2.875315  16528  38.60714   32.8
 3 003  BOND   IL    0.033  14897  181.4597  14477   429    25    16   7.5182  1.448572  14235  34.83587   12.9
 4 004  BOONE  IL    0.044  54165  124.4444  50788   3170   46   162  17.5968  4.167602  53157  34.47767   7.
 5 005  BROWN  IL    0.016  5638  114.2222   5264   347    14    5   5.4763  0.267860   4816  42.50314   13.5
 6 006  BUREAU IL    0.036  35059  75.7600  33147   1846   35   30  16.9789  1.898802  33157  38.52100   12.
 7 007  CALHOUN IL    0.037  5322  213.0000   5266      1    8    5   7.0736  3.206667   5241  38.47922   15.1
 8 008  CARROLL IL    0.037  16805  82.4074  16018   170   50   22  16.9872  3.505217   16655  37.97728   9.
 9 009  CASS   IL    0.024  13437  150.8750  13364    18    8   23   14.5791  3.106799   13391  37.50360   13.8
10 010  CHAMPAIGN IL    0.036  179553  266.6667  165318  10559   151   510  45.2984  11.575428  164924  35.63428   15.8
11 011  CHRISTIAN IL    0.043  30419  81.6762  30176   82   31   69   13.9172  3.000998   31768   36.16956   11.5
12 012  CLARK  IL    0.032  16501  125.5000  16442   12   26   20   10.7098  2.776235   16876   36.09391   10.
13 013  CLAY   IL    0.028  14491  116.4286  14453    6   17   28   13.6820  2.784432   14248   36.53389   18.7
14 014  CLINTON IL    0.032  37762  135.4839  33988   521   48   104   16.9074  2.675298   33780   34.83307   10.
15 015  COLES  IL    0.032  53644  172.4867  50777   925   82   241   23.7504  8.144363   45883   35.47809   18.7
16 016  COOK   IL    1.000  5194966  4266.0000  5074967  107587  10888  16956   22.8864  5.662624  5023023   36.40358   22.
17 017  CRAWFORD IL    0.028  19464  142.0514  18353   833   34   48   16.8581  3.104304   19123   36.24835   15.5
18 018  CUMBERLAND IL    0.036  11670  150.0000  10937   1   8   16   14.9789  1.898802   10880   35.10200   13.
19 019  DE KALB IL    0.028  77652  200.0645  72868   2588   123   1701   12.8350  1.163867   69127   36.70789   13.6
20 020  DE WITT IL    0.032  16798  78.8679  16387   21   17   42   16.7888  3.200038   16238   36.24678   16.
21 021  DOUGLAS IL    0.025  19464  178.5600  18280   18   19   37   16.8863  2.840337   18140   36.35139   9.6
22 022  DU PAGE IL    0.025  916924  3800.0000  794855  5461   862   24854   14.8867  1.908228   774842   36.74766   8.
23 023  EDGAR  IL    0.036  18285  164.3396  18402   88   24   14   16.8863  2.876608   18258   36.20077   15.0
24 024  EDWARDS IL    0.024  7441  141.6286   7401    3    4   12   16.8394  2.425997   7350   36.56250   12.
25 025  EFFINGHAM IL    0.028  21750  143.2897  21523   12   45   56   23.2520  3.783960   21226   36.50777   8.8
26 026  FAYETTE IL    0.024  22083  214.6019  21540   195   45   31   15.6026  2.888541   19452   37.55286   13.
27 027  FORD   IL    0.025  14279  175.6333  14167   63   14   60   13.5277  2.018325   13271   37.09015   15.6
28 028  FRANKLIN IL    0.025  40259  163.7500  40088   38   58   44   14.7680  2.410371   39753   36.47156   20.7
29 029  FULTON IL    0.032  29886  151.7077  3717   498   83   155   15.6325  2.997538   36390   35.80869   15.4
30 030  GALLATIN IL    0.024  8989  203.6364  8842   42   10   17   11.3870  2.068208   8737   37.55164   21.4
```

```
# ggplot(data=millmet) +
# geom_histogram(aes(x=popdensity, fill=as.character(metro)), bins = 20)
```




```
# ggplot(data=millmet, popdensity < 1000) +
# geom_histogram(aes(x=popdensity), fill=as.character(metro), bins = 20)
```




```
# ggplot(filter(millmet, popdensity < 1000)) + geom_point(aes(popdensity), lineweight=1)
```

Let's plot a single logistic model on the data.

```
# define a logistic model
m1 = glm
m2 = tidyglm

logistic_curve <- function(x) {1L / (1+exp(-(a+b*x)))}

logistic_curve(100)
```

```
0.0379682209652916
```

```
# plot a logistic model
# ggplot(data=millmet, popdensity < 1000) +
#   geom_point(aes(popdensity), lineweight=1) +
#   geom_function(fn = logistic_curve)
```

```
# let's used the filtered dataset (the millions cause some nonsical errors in our implementation)

# on the one fifth on our one in
train = filter(millmet, popdensity < 1000) 5%
  sample_frac(.2)

valid_test <- anti_join(filter(millmet, popdensity < 1000), train, by = "FIP")

valid = valid_test 5%
  sample_frac(.5)

test <- anti_join(valid_test, valid, by = "FIP")
```

Let's write a Binary Cross Entropy cost function. For reference here is the RMSE loss function from last week.

```
RMSE <- function(y, data) {
  preds <- d(.) * data$ - a(.)
  diffs <- data$ - preds
  mean(sqrt(diff^2))
}
```

And here's the formula for BCE Loss:

$$\frac{\sum_{i=1}^{N} -[y_i \log(p(x_i)) + (1-y_i)\log(1-p(x_i))]}{N}$$

```
# compute the outputs of an a value for a particular choice of parameters
logistic_cost <- function(a, y) {-(y*log(a)+(1-y)*log(1-a))}

# now use it work for individual data
logistic_cost(0.5,1), 0.5,1.1,0.01

-0.3010299956639005 -0.0869793797799021 -0.0869793797799021 -0.00201070029968
```

```
BCE <- function(a, data) {
  preds <- d(.) * data$popdensity, data$y
  diffs <- preds - a
  value <- 1ntr * logistic * (1 - 1ntr) * log(1 - preds)
  mean(-value)
}

(cost <- sprintf(c(0, 1)100), BCE, data = millmet)[1:5, ]
```

Var	0.0101007766843 -0.0101007766843
Svalue	0.371200364664
Accuracy	function: f(0) **prediction**: <NA>
Scorecardpoints	0
Grewcapt	NULL

```
m1 = lmer(y~1)
m2 = lmer(y~1)

logistic_curve <- function(x) {1L / (1+exp(-(a+b*x)))}

ggplot(data=
  geom_function(fn = logistic_curve)
  geom_function(fn = logistic_curve)
```

```
BCE(data=lmer, mutate=(a=1, a = popdensity), y = (metro))

ggplot(data=
  geom_function(fn = logistic_curve)
  geom_function(fn = logistic_curve)
```

0.2728974022954

Let's compute a **metric** for our model: Accuracy

- There are other metrics for binary classification such as Precision and Recall

```
avg_accuracy <- function(a, data, threshold=0.5) {
  preds <- logistic_model(a, data)$y  # get predictions for our data
  preds <- (preds >= .5 # turn our predictions into predictions, using our threshold

  # remember this is a boolean vector, T or 1 and F or 0!
  # so we can sum it
  sdiag <- as.logical(y - (preds)
  get_diag <- sdiag[sdiag] # length(sdiag)
  length(get_diag)
}

avg_accuracy(lmer$lpm, data=millmet, a = popdensity, y = (metro))
avg_accuracy(lmer, data=millmet, a = popdensity, y = (metro))

0.8356745973923223
0.8356745973923223
```

For fun we can compare the accuracy of a linear model using RMSE.

```
(lmer_linear <- tp(lmer(m, 8), RMSE, data = millmet)[1:4, ] + popdensity, y = (metro))
```

Var	0.0101007766843 -0.0101007766843
Svalue	0.371200364664
Accuracy	function: f(0) **prediction**: <NA>
Scorecardpoints	0
Grewcapt	NULL

```
ggplot(data=
  geom_point(aes(popdensity), lineweight=1)
  geom_abline(aes(intercept = lmer1_linear$par[1], slope = lmer1_linear$par[2]))
```

```
avg_accuracy_linear <- function(a, data, threshold=0.5) {
  preds <- logistic_model(a, data)$y  # get predictions for our data
  preds <- (preds >= .5 # turn our predictions into predictions, using our threshold

  # remember this is a boolean vector, T or 1 and F or 0!
  # so we can sum it
  sdiag <- as.logical(y - (preds)
  get_diag <- sdiag[sdiag]
  length(get_diag)
}

avg_accuracy_linear(lmer1_linear$par, data=millmet, a = popdensity, y = (metro))
avg_accuracy_linear(lmer1_linear$par, data=millmet, a = popdensity, y = (metro))

0.8356745973923223
0.8356745973923223
```

Using `lm` and `glm`

```
xlmer
```

```
# tibble 20 x 1
   x     .
   <dbl> <dbl>
 1  0.0000 0.
 2  0.0526 0.
 3  0.1053 0.
 4  0.1579 0.
 5  0.2105 0.
 6  0.2632 0.
 7  0.3158 0.
 8  0.3684 0.
 9  0.4211 0.
10  0.4737 0.
11  0.5263 0.
12  0.5789 0.
13  0.6316 0.
14  0.6842 0.
15  0.7368 0.
16  0.7895 0.
17  0.8421 0.
18  0.8947 0.
19  0.9474 0.
20  1.0000 0.
```

```
lm(y ~ x, data)

Call:
lm(formula = y ~ x, data = data)

Coefficients:
(Intercept)            x
          0            0

lm(metro ~ popdensity, train)

Call:
lm(formula = (metro) ~ popdensity, family = binomial)

Coefficients:
(Intercept)  popdensity
  -0.01282      0.00000

glm(data = train, (metro) ~ popdensity, family = binomial)

Call:  glm(formula = (metro) ~ popdensity, family = binomial, data = train)

Coefficients:
(Intercept)  popdensity
  -1.4197    0.00157

Degrees of Freedom: 1 Total (i.e. Null);  1 Residual
Null Deviance:      1.386
Residual Deviance: 1.386    AIC: 77.9

# model to plot: y ~ a*x + a*x^2 + a*x^3 + a*(sqrt x) + a*(sqrt x)^2 + a*(sqrt x)^3
lm(data = dlmmodel, y(x) ~ a + x + x^2 + x^3 + sqrt(x) + sqrt(x)^2 + sqrt(x)^3)

Call:
lm(formula = y(x) ~ a + x + x^2 + x^3 + sqrt(x) + sqrt(x)^2 + sqrt(x)^3,
    data = dlmmodel)

Coefficients:
(Intercept)            a            x       I(x^2)       I(x^3)      sqrt(x)  I(sqrt(x)^2)  I(sqrt(x)^3)
     -1.000       -1.000       -1.000        1.000        1.000        1.000        -1.000        -1.000

# model model_price = 20994.57 - 1214.57x + 90.52y ....
```