Replication File Chapter 2

Theresa Küntzler

23 11 2020

Packages and Folders

```
library(dplyr)
library(caret)
library(purrr)
library(ggplot2)
library(tidyr)
library(pROC)
library(plotROC)

data <- "../data/"

# for plotting
my_lightgreen <- "#74c476"
my_darkgreen <- "#31a354"
my_lightblue <- "#a6cee3"
my_darkblue <- "#1f78b4"
```

Compare Prototypical Data to Azure

```
load(file = paste0(data, "labdata_azure_emo_max.Rda"))

# extract those with undetected faces: 0: 0% undetected)
labdata_azure_na <- filter(labdata_azure, is.na(emo_anger))
rm(labdata_azure_na)

# confusion matrix
emo_order <- c("neutral", "happy", "angry", "disgust",
               "surprise", "sad", "fear")
confmat <- confusionMatrix(data =
                             factor(as.factor(labdata_azure$emo_azure_max_char),
                                    levels = emo_order),
                           reference = factor(labdata_azure$emotion,
                                              levels = emo_order),
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neutral happy angry disgust surprise sad fear
##   neutral      178     0    72       1        3  20    8
##   happy          0   178     1       0        1   1    1
##   angry          0     0    90       9        0   0    0
##   disgust        0     0     2     151        0   0    1
##   surprise       0     0     1       0      174   0   62
##   sad            0     0    11      17        0 157   25
##   fear           0     0     1       0        0   0   81
##
## Overall Statistics
##
##                Accuracy : 0.8098
##                  95% CI : (0.7869, 0.8312)
##     No Information Rate : 0.1429
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7781
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: neutral Class: happy Class: angry Class: disgust
## Sensitivity                  1.0000       1.0000      0.50562         0.8483
## Specificity                  0.9026       0.9963      0.99157         0.9972
## Pos Pred Value               0.6312       0.9780      0.90909         0.9805
## Neg Pred Value               1.0000       1.0000      0.92328         0.9753
## Precision                    0.6312       0.9780      0.90909         0.9805
## Recall                       1.0000       1.0000      0.50562         0.8483
## F1                           0.7739       0.9889      0.64982         0.9096
## Prevalence                   0.1429       0.1429      0.14286         0.1429
## Detection Rate               0.1429       0.1429      0.07223         0.1212
## Detection Prevalence         0.2263       0.1461      0.07945         0.1236
## Balanced Accuracy            0.9513       0.9981      0.74860         0.9228
##                      Class: surprise Class: sad Class: fear
## Sensitivity                   0.9775     0.8820     0.45506
## Specificity                   0.9410     0.9504     0.99906
## Pos Pred Value                0.7342     0.7476     0.98780
## Neg Pred Value                0.9960     0.9797     0.91667
## Precision                     0.7342     0.7476     0.98780
## Recall                        0.9775     0.8820     0.45506
## F1                            0.8386     0.8093     0.62308
## Prevalence                    0.1429     0.1429     0.14286
## Detection Rate                0.1396     0.1260     0.06501
## Detection Prevalence          0.1902     0.1685     0.06581
## Balanced Accuracy             0.9593     0.9162     0.72706
```

```r
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.8097913
```

```r
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.8526342
```

```r
rm(confmat)

# confusions as table
emo_order <- c("neutral", "happy", "angry", "disgust", "surprise", "sad", "fear")

## Prop Table
my.prop.tab <- prop.table(table(factor(labdata_azure$emotion,
                                       levels = emo_order),
                                factor(as.factor(labdata_azure$emo_azure_max_char),
                                       levels = emo_order)), margin = 1)
my.prop.tab[is.na(my.prop.tab)] <- 0
my.prop.tab
```

```
##
##                neutral       happy       angry     disgust    surprise
##   neutral  1.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##   happy    0.000000000 1.000000000 0.000000000 0.000000000 0.000000000
##   angry    0.404494382 0.005617978 0.505617978 0.011235955 0.005617978
##   disgust  0.005617978 0.000000000 0.050561798 0.848314607 0.000000000
##   surprise 0.016853933 0.005617978 0.000000000 0.000000000 0.977528090
##   sad      0.112359551 0.005617978 0.000000000 0.000000000 0.000000000
##   fear     0.044943820 0.005617978 0.000000000 0.005617978 0.348314607
##
##                    sad        fear
##   neutral  0.000000000 0.000000000
##   happy    0.000000000 0.000000000
##   angry    0.061797753 0.005617978
##   disgust  0.095505618 0.000000000
##   surprise 0.000000000 0.000000000
##   sad      0.882022472 0.000000000
##   fear     0.140449438 0.455056180
```

```r
rm(labdata_azure, my.prop.tab, emo_order)
```

Compare Prototypical Data to Face++

```r
load(file = paste0(data, "labdata_facepp_emo_max.Rda"))

labdata_facepp_na <- filter(labdata_facepp, is.na(emo_anger))
rm(labdata_facepp_na)

# confusion matrix
emo_names_fpp <- c("angry", "disgust", "fear", "happy", "neutral", "sad",
                   "surprise")
```

```r
confmat <- confusionMatrix(data =
                             factor(as.factor(labdata_facepp$emo_fpp_max_char),
                                    levels = emo_names_fpp),
                           reference = factor(labdata_facepp$emotion,
                                              levels = emo_names_fpp),
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction angry disgust fear happy neutral sad surprise
##   angry       87      11    1     0       1   3        0
##   disgust     25     159   15     1       1   5        0
##   fear         1       0   71     0       0   2        1
##   happy        0       0    3   177       2   1        1
##   neutral     43       1    6     0     168  21        3
##   sad         19       7   21     0       2 144        0
##   surprise     3       0   61     0       4   2      173
##
## Overall Statistics
##
##                Accuracy : 0.7857
##                  95% CI : (0.7619, 0.8082)
##     No Information Rate : 0.1429
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.75
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: angry Class: disgust Class: fear Class: happy
## Sensitivity               0.48876         0.8933     0.39888       0.9944
## Specificity               0.98502         0.9560     0.99625       0.9934
## Pos Pred Value            0.84466         0.7718     0.94667       0.9620
## Neg Pred Value            0.92038         0.9817     0.90863       0.9991
## Precision                 0.84466         0.7718     0.94667       0.9620
## Recall                    0.48876         0.8933     0.39888       0.9944
## F1                        0.61922         0.8281     0.56126       0.9779
## Prevalence                0.14286         0.1429     0.14286       0.1429
## Detection Rate            0.06982         0.1276     0.05698       0.1421
## Detection Prevalence      0.08266         0.1653     0.06019       0.1477
## Balanced Accuracy         0.73689         0.9246     0.69757       0.9939
##                      Class: neutral Class: sad Class: surprise
## Sensitivity                  0.9438     0.8090          0.9719
## Specificity                  0.9307     0.9541          0.9345
## Pos Pred Value               0.6942     0.7461          0.7119
## Neg Pred Value               0.9900     0.9677          0.9950
## Precision                    0.6942     0.7461          0.7119
## Recall                       0.9438     0.8090          0.9719
```

```
## F1                           0.8000     0.7763          0.8219
## Prevalence                   0.1429     0.1429          0.1429
## Detection Rate               0.1348     0.1156          0.1388
## Detection Prevalence         0.1942     0.1549          0.1950
## Balanced Accuracy            0.9373     0.8816          0.9532
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.7857143
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.8110559
```

```
rm(confmat)

# confusions as table
emo_order_fpp <- c("neutral", "happy", "angry", "disgust", "surprise", "sad",
                   "fear")

## Prop Table
my.prop.tab <- prop.table(table(factor(labdata_facepp$emotion,
                                       levels = emo_order_fpp),
                                factor(as.factor(labdata_facepp$emo_fpp_max_char),
                                       levels = emo_order_fpp)), margin = 1)
my.prop.tab[is.na(my.prop.tab)] <- 0
my.prop.tab
```

```
##
##                neutral       happy       angry     disgust    surprise
##   neutral  0.943820225 0.011235955 0.005617978 0.005617978 0.022471910
##   happy    0.000000000 0.994382022 0.000000000 0.005617978 0.000000000
##   angry    0.241573034 0.000000000 0.488764045 0.140449438 0.016853933
##   disgust  0.005617978 0.000000000 0.061797753 0.893258427 0.000000000
##   surprise 0.016853933 0.005617978 0.000000000 0.000000000 0.971910112
##   sad      0.117977528 0.005617978 0.016853933 0.028089888 0.011235955
##   fear     0.033707865 0.016853933 0.005617978 0.084269663 0.342696629
##
##                    sad        fear
##   neutral  0.011235955 0.000000000
##   happy    0.000000000 0.000000000
##   angry    0.106741573 0.005617978
##   disgust  0.039325843 0.000000000
##   surprise 0.000000000 0.005617978
##   sad      0.808988764 0.011235955
##   fear     0.117977528 0.398876404
```

```
rm(labdata_facepp, my.prop.tab, emo_names_fpp)
```

Compare Prototypical Data to FaceReader

```
load(file = paste0(data, "labdata_fr.Rda"))

# extract those with undetected faces: 2
labdata_fr_na <- filter(labdata_fr, is.na(Angry))
table(labdata_fr_na$emotion)
```

```
##
##   angry disgust
##       1       1
```

```
rm(labdata_fr_na)

# per category
# angry
1/178*100
```

```
## [1] 0.5617978
```

```
# disgust
1/178*100
```

```
## [1] 0.5617978
```

```
# images with quality below 0.7
labdata_qual_sm_7 <- filter(labdata_fr, Quality <0.7)
table(labdata_qual_sm_7$emotion)
```

```
##
##    angry  disgust     fear  neutral surprise
##        1        1        2        1        4
```

```
## all dropout per category (na + below 0.7)
# angry
2/178*100
```

```
## [1] 1.123596
```

```
# digust
2/178*100
```

```
## [1] 1.123596
```

```
# fear
2/178*100
```

```
## [1] 1.123596
```

```r
# neutral
1/178*100
```

```
## [1] 0.5617978
```

```r
# surprise
4/178*100
```

```
## [1] 2.247191
```

```r
rm(labdata_qual_sm_7)

# filter quality above 0.7
labdata_fr <- filter(labdata_fr, Quality>0.7)

# confusion matrix
emo_order_fpp <- c("neutral", "happy", "angry", "disgust", "surprise", "sad",
                   "fear")

confmat <- confusionMatrix(data = factor(as.factor(labdata_fr$emo_fr_max_char),
                                         levels = emo_order_fpp),
                           reference = factor(labdata_fr$emotion,
                                              levels = emo_order_fpp),
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neutral happy angry disgust surprise sad fear
##   neutral      176     0     3       2        1   3    6
##   happy          0   178     0       0        0   0    1
##   angry          0     0   169       2        0   0    0
##   disgust        0     0     2     171        0   0    0
##   surprise       0     0     0       0      171   1   11
##   sad            1     0     1       0        0 174    3
##   fear           0     0     1       1        2   0  155
##
## Overall Statistics
##
##                Accuracy : 0.9668
##                  95% CI : (0.9552, 0.9761)
##     No Information Rate : 0.1441
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9613
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: neutral Class: happy Class: angry Class: disgust
## Sensitivity                  0.9944       1.0000       0.9602         0.9716
## Specificity                  0.9858       0.9991       0.9981         0.9981
## Pos Pred Value               0.9215       0.9944       0.9883         0.9884
## Neg Pred Value               0.9990       1.0000       0.9934         0.9953
## Precision                    0.9215       0.9944       0.9883         0.9884
## Recall                       0.9944       1.0000       0.9602         0.9716
## F1                           0.9565       0.9972       0.9741         0.9799
## Prevalence                   0.1433       0.1441       0.1425         0.1425
## Detection Rate               0.1425       0.1441       0.1368         0.1385
## Detection Prevalence         0.1547       0.1449       0.1385         0.1401
## Balanced Accuracy            0.9901       0.9995       0.9792         0.9849
##                      Class: surprise Class: sad Class: fear
## Sensitivity                   0.9828     0.9775      0.8807
## Specificity                   0.9887     0.9953      0.9962
## Pos Pred Value                0.9344     0.9721      0.9748
## Neg Pred Value                0.9971     0.9962      0.9805
## Precision                     0.9344     0.9721      0.9748
## Recall                        0.9828     0.9775      0.8807
## F1                            0.9580     0.9748      0.9254
## Prevalence                    0.1409     0.1441      0.1425
## Detection Rate                0.1385     0.1409      0.1255
## Detection Prevalence          0.1482     0.1449      0.1287
## Balanced Accuracy             0.9857     0.9864      0.9385
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.9667339
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.9677084
```

```
rm(confmat)

# confusions as table

## Prop Table
my.prop.tab <- prop.table(table(factor(labdata_fr$emotion,
                                       levels = emo_order_fpp),
                                factor(as.factor(labdata_fr$emo_fr_max_char),
                                       levels = emo_order_fpp)), margin = 1)
my.prop.tab[is.na(my.prop.tab)] <- 0
my.prop.tab
```

```
##
##                neutral       happy       angry     disgust    surprise
##   neutral  0.994350282 0.000000000 0.000000000 0.000000000 0.000000000
##   happy    0.000000000 1.000000000 0.000000000 0.000000000 0.000000000
##   angry    0.017045455 0.000000000 0.960227273 0.011363636 0.000000000
##   disgust  0.011363636 0.000000000 0.011363636 0.971590909 0.000000000
```

```
##   surprise 0.005747126 0.000000000 0.000000000 0.000000000 0.982758621
##   sad      0.016853933 0.000000000 0.000000000 0.000000000 0.005617978
##   fear     0.034090909 0.005681818 0.000000000 0.000000000 0.062500000
##
##                    sad        fear
##   neutral  0.005649718 0.000000000
##   happy    0.000000000 0.000000000
##   angry    0.005681818 0.005681818
##   disgust  0.000000000 0.005681818
##   surprise 0.000000000 0.011494253
##   sad      0.977528090 0.000000000
##   fear     0.017045455 0.880681818
```

```
rm(labdata_fr, my.prop.tab)
```

Compare Naturalistic Data to Azure

```
load(file = paste0(data, "sfew_azure_emo_max.Rda"))

# extract those with undetected faces: 282 undetected)
sfew_azure_na <- filter(sfew_azure, is.na(emo_anger))
1- nrow(sfew_azure_na)/1387
```

```
## [1] 0.7966835
```

```
table(sfew_azure_na$emotion)/1387*100
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
## 4.614275 1.514059 2.739726 2.162942 2.883922 4.470079 1.946647
```

```
# drop out shares
table(sfew_azure_na$emotion)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       64       21       38       30       40       62       27
```

```
# drop out shares
table(sfew_azure_na$emotion)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       64       21       38       30       40       62       27
```

```
rm(sfew_azure_na)
# % based on amount in category
# anger
64/254
```

```
## [1] 0.2519685
```

```
# disgust
21/88
```

```
## [1] 0.2386364
```

```
# fear
38/143
```

```
## [1] 0.2657343
```

```
# happy
30/270
```

```
## [1] 0.1111111
```

```
# neutral
40/236
```

```
## [1] 0.1694915
```

```
#sad
62/245
```

```
## [1] 0.2530612
```

```
# surprise
27/151
```

```
## [1] 0.1788079
```

```
# Confusion Matrix
emo_order <- c("neutral", "happy", "angry", "disgust", "surprise", "sad", "fear")

confmat <- confusionMatrix(data =
                             factor(as.factor(sfew_azure$emo_azure_max_char),
                                    levels = emo_order),
                           reference =
                             factor(sfew_azure$emotion, levels = emo_order),
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neutral happy angry disgust surprise sad fear
##   neutral      185    29    70      37       45  78   43
##   happy          1   206     5      11        5   5    2
```

```
##   angry          1     0    73       2        2   1    5
##   disgust        0     0     7       7        0   0    0
##   surprise       8     5    26       7       70   8   39
##   sad            1     0     7       3        2  87   13
##   fear           0     0     2       0        0   4    3
##
## Overall Statistics
##
##                Accuracy : 0.571
##                  95% CI : (0.5412, 0.6005)
##     No Information Rate : 0.2172
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4816
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: neutral Class: happy Class: angry Class: disgust
## Sensitivity                  0.9439       0.8583      0.38421       0.104478
## Specificity                  0.6678       0.9665      0.98798       0.993256
## Pos Pred Value               0.3799       0.8766      0.86905       0.500000
## Neg Pred Value               0.9822       0.9609      0.88541       0.945005
## Precision                    0.3799       0.8766      0.86905       0.500000
## Recall                       0.9439       0.8583      0.38421       0.104478
## F1                           0.5417       0.8674      0.53285       0.172840
## Prevalence                   0.1774       0.2172      0.17195       0.060633
## Detection Rate               0.1674       0.1864      0.06606       0.006335
## Detection Prevalence         0.4407       0.2127      0.07602       0.012670
## Balanced Accuracy            0.8058       0.9124      0.68609       0.548867
##                      Class: surprise Class: sad Class: fear
## Sensitivity                  0.56452    0.47541    0.028571
## Specificity                  0.90520    0.97180    0.994000
## Pos Pred Value               0.42945    0.76991    0.333333
## Neg Pred Value               0.94268    0.90323    0.906934
## Precision                    0.42945    0.76991    0.333333
## Recall                       0.56452    0.47541    0.028571
## F1                           0.48780    0.58784    0.052632
## Prevalence                   0.11222    0.16561    0.095023
## Detection Rate               0.06335    0.07873    0.002715
## Detection Prevalence         0.14751    0.10226    0.008145
## Balanced Accuracy            0.73486    0.72361    0.511286
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.4799138
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.5940304
```

```r
rm(confmat)

# confusions as table
## Prop Table
my.prop.tab <- prop.table(table(factor(sfew_azure$emotion, levels = emo_order),
                factor(as.factor(sfew_azure$emo_azure_max_char),
                       levels = emo_order)),
                margin = 1)
my.prop.tab[is.na(my.prop.tab)] <- 0
my.prop.tab
```

```
##
##                neutral       happy       angry     disgust    surprise
##   neutral  0.943877551 0.005102041 0.005102041 0.000000000 0.040816327
##   happy    0.120833333 0.858333333 0.000000000 0.000000000 0.020833333
##   angry    0.368421053 0.026315789 0.384210526 0.036842105 0.136842105
##   disgust  0.552238806 0.164179104 0.029850746 0.104477612 0.104477612
##   surprise 0.362903226 0.040322581 0.016129032 0.000000000 0.564516129
##   sad      0.426229508 0.027322404 0.005464481 0.000000000 0.043715847
##   fear     0.409523810 0.019047619 0.047619048 0.000000000 0.371428571
##
##                    sad        fear
##   neutral  0.005102041 0.000000000
##   happy    0.000000000 0.000000000
##   angry    0.036842105 0.010526316
##   disgust  0.044776119 0.000000000
##   surprise 0.016129032 0.000000000
##   sad      0.475409836 0.021857923
##   fear     0.123809524 0.028571429
```

```r
rm(sfew_azure, my.prop.tab)
```

Compare Naturalistic Data to Face++

```r
load(file = paste0(data, "sfew_facepp_emo_max.Rda"))

# extract those with undetected faces
sfew_facepp_na <- filter(sfew_facepp, is.na(emo_anger))
1- (nrow(sfew_facepp_na)/nrow(sfew_facepp))*100
```

```
## [1] 0.4953136
```

```r
table(sfew_facepp_na$emotion)/1387*100
```

```
##
##      angry    disgust       fear      happy    neutral        sad   surprise
## 0.07209805 0.07209805 0.07209805 0.14419611 0.00000000 0.14419611 0.00000000
```

```r
# drop out shares
table(sfew_facepp_na$emotion)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##        1        1        1        2        0        2        0
```

```r
rm(sfew_facepp_na)
# % based on amount in category
# anger
1/254
```

```
## [1] 0.003937008
```

```r
# disgust
1/88
```

```
## [1] 0.01136364
```

```r
# fear
1/143
```

```
## [1] 0.006993007
```

```r
# happy
2/270
```

```
## [1] 0.007407407
```

```r
# neutral
0/236
```

```
## [1] 0
```

```r
#sad
2/245
```

```
## [1] 0.008163265
```

```r
# surprise
0/151
```

```
## [1] 0
```

```r
# confusion matrix
confmat <- confusionMatrix(data = factor(as.factor(sfew_facepp$emo_fpp_max_char),
                                         levels = levels(sfew_facepp$emotion)),
                           reference = sfew_facepp$emotion,
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction angry disgust fear happy neutral sad surprise
##   angry       38      10   12    14      12  14        5
##   disgust     32      14    4    14       8   9        2
##   fear        33       9   26    17      14  32       15
##   happy        8       5    5   128       6  15        1
##   neutral     33      22   20    20      95  60       26
##   sad          9       4    9    22      20  45        3
##   surprise   100      23   66    53      81  68       99
##
## Overall Statistics
##
##                Accuracy : 0.3225
##                  95% CI : (0.2978, 0.3478)
##     No Information Rate : 0.1942
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2125
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: angry Class: disgust Class: fear Class: happy
## Sensitivity               0.15020        0.16092     0.18310      0.47761
## Specificity               0.94055        0.94664     0.90307      0.96403
## Pos Pred Value            0.36190        0.16867     0.17808      0.76190
## Neg Pred Value            0.83137        0.94372     0.90600      0.88449
## Precision                 0.36190        0.16867     0.17808      0.76190
## Recall                    0.15020        0.16092     0.18310      0.47761
## F1                        0.21229        0.16471     0.18056      0.58716
## Prevalence                0.18333        0.06304     0.10290      0.19420
## Detection Rate            0.02754        0.01014     0.01884      0.09275
## Detection Prevalence      0.07609        0.06014     0.10580      0.12174
## Balanced Accuracy         0.54537        0.55378     0.54308      0.72082
##                      Class: neutral Class: sad Class: surprise
## Sensitivity                 0.40254    0.18519         0.65563
## Specificity                 0.84178    0.94107         0.68186
## Pos Pred Value              0.34420    0.40179         0.20204
## Neg Pred Value              0.87228    0.84385         0.94157
## Precision                   0.34420    0.40179         0.20204
## Recall                      0.40254    0.18519         0.65563
## F1                          0.37109    0.25352         0.30889
## Prevalence                  0.17101    0.17609         0.10942
## Detection Rate              0.06884    0.03261         0.07174
## Detection Prevalence        0.20000    0.08116         0.35507
## Balanced Accuracy           0.62216    0.56313         0.66874
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.3164549
```

```r
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.3455137
```

```r
rm(confmat)

# confusions as table
emo_order_fpp <- c("neutral", "happy", "angry", "disgust", "surprise",
                   "sad", "fear")

## Prop Table
my.prop.tab <- prop.table(table(factor(sfew_facepp$emotion,
                                       levels = emo_order_fpp),
                factor(as.factor(sfew_facepp$emo_fpp_max_char),
                       levels = emo_order_fpp)),
                margin = 1)
my.prop.tab[is.na(my.prop.tab)] <- 0
my.prop.tab
```

```
##
##                neutral       happy       angry     disgust    surprise
##   neutral  0.402542373 0.025423729 0.050847458 0.033898305 0.343220339
##   happy    0.074626866 0.477611940 0.052238806 0.052238806 0.197761194
##   angry    0.130434783 0.031620553 0.150197628 0.126482213 0.395256917
##   disgust  0.252873563 0.057471264 0.114942529 0.160919540 0.264367816
##   surprise 0.172185430 0.006622517 0.033112583 0.013245033 0.655629139
##   sad      0.246913580 0.061728395 0.057613169 0.037037037 0.279835391
##   fear     0.140845070 0.035211268 0.084507042 0.028169014 0.464788732
##
##                   sad        fear
##   neutral  0.084745763 0.059322034
##   happy    0.082089552 0.063432836
##   angry    0.035573123 0.130434783
##   disgust  0.045977011 0.103448276
##   surprise 0.019867550 0.099337748
##   sad      0.185185185 0.131687243
##   fear     0.063380282 0.183098592
```

```r
rm(sfew_facepp, my.prop.tab, emo_order_fpp)
```

Compare Naturalistic Data to FaceReader

```r
load(file = paste0(data, "sfew_fr.Rda"))

# extract those with undetected faces: 750
sfew_fr_na <- filter(sfew_fr, is.na(Angry))
# % detected face
1- length(sfew_fr_na$filename)/length(sfew_fr$filename)
```

```
## [1] 0.4592646
```

```
nrow(sfew_fr_na)/nrow(sfew_fr)/1387
```

```
## [1] 0.0003898597
```

```
table(sfew_fr_na$emotion)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##      143       48       85      135      126      136       77
```

```
# quality too low
sfew_qual_sm_7 <- filter(sfew_fr, Quality <0.7)
nrow(sfew_qual_sm_7)/nrow(sfew_fr)
```

```
## [1] 0.2011536
```

```
table(sfew_qual_sm_7$emotion)/1387
```

```
##
##       angry     disgust        fear       happy     neutral         sad
## 0.037490988 0.009372747 0.020187455 0.038932949 0.034607066 0.034607066
##    surprise
## 0.025955299
```

```
table(sfew_qual_sm_7$emotion)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       52       13       28       54       48       48       36
```

```
# sum quality and undetected
# % based on amount in category
# anger
(52+143)/254 * 100
```

```
## [1] 76.77165
```

```
# disgust
(13+48)/88 * 100
```

```
## [1] 69.31818
```

```
# fear
(28+85)/143 * 100
```

```
## [1] 79.02098
```

```r
# happy
(54+135)/270 * 100
```

```
## [1] 70
```

```r
# neutral
(48+126)/236 * 100
```

```
## [1] 73.72881
```

```r
#sad
(48+136)/245 * 100
```

```
## [1] 75.10204
```

```r
# surprise
(36+77)/151 * 100
```

```
## [1] 74.83444
```

```r
rm(sfew_fr_na, sfew_qual_sm_7)

# keep those with qual >= .7 for the analysis
sfew_fr <- filter(sfew_fr, Quality >=0.7)

# confusion matrix
confmat <- confusionMatrix(data = factor(as.factor(sfew_fr$emo_fr_max_char),
                                         levels = levels(sfew_fr$emotion)),
                           reference = sfew_fr$emotion,
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction angry disgust fear happy neutral sad surprise
##   angry        8       4    0     2       3   0        2
##   disgust     13       4    0     4       0   1        1
##   fear         1       0    0     1       0   5        0
##   happy        1       1    0    34       0   1        0
##   neutral     26      14   22    38      42  39       20
##   sad          3       4    1     0      11  10        2
##   surprise     7       0    7     2       6   5       13
##
## Overall Statistics
##
##                Accuracy : 0.3101
##                  95% CI : (0.2625, 0.3608)
##     No Information Rate : 0.2263
##     P-Value [Acc > NIR] : 0.0001583
```

```
##
##                   Kappa : 0.1762
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: angry Class: disgust Class: fear Class: happy
## Sensitivity               0.13559        0.14815     0.00000      0.41975
## Specificity               0.96321        0.94260     0.97866      0.98917
## Pos Pred Value            0.42105        0.17391     0.00000      0.91892
## Neg Pred Value            0.84956        0.93134     0.91453      0.85358
## Precision                 0.42105        0.17391     0.00000      0.91892
## Recall                    0.13559        0.14815     0.00000      0.41975
## F1                        0.20513        0.16000         NaN      0.57627
## Prevalence                0.16480        0.07542     0.08380      0.22626
## Detection Rate            0.02235        0.01117     0.00000      0.09497
## Detection Prevalence      0.05307        0.06425     0.01955      0.10335
## Balanced Accuracy         0.54940        0.54537     0.48933      0.70446
##                      Class: neutral Class: sad Class: surprise
## Sensitivity                  0.6774    0.16393         0.34211
## Specificity                  0.4628    0.92929         0.91563
## Pos Pred Value               0.2090    0.32258         0.32500
## Neg Pred Value               0.8726    0.84404         0.92138
## Precision                    0.2090    0.32258         0.32500
## Recall                       0.6774    0.16393         0.34211
## F1                           0.3194    0.21739         0.33333
## Prevalence                   0.1732    0.17039         0.10615
## Detection Rate               0.1173    0.02793         0.03631
## Detection Prevalence         0.5615    0.08659         0.11173
## Balanced Accuracy            0.5701    0.54661         0.62887
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.2695648
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.3386315
```

```
rm(confmat)

rm(sfew_fr)
```

ROC Curve Analysis

Prototypical Data

If you wish to replicate the full procedure mark the box with 'eval = TRUE' to do the following to: 1) Create long data frames from the wide ones 2) Create roc-objects from the long dataframe 3) run the tests Note, the tests are bootstrapped and therefore stochastic.

You can also jump to loading the test-obejcts reported in the Dissertation

```
# Prototypical Data Azure
load(file = paste0(data, "labdata_azure_emo_max.Rda"))

labdata_azure_long <- labdata_azure %>%
  select(fullpath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(fullpath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "emo_disgust" ~ 1,
                              emotion == "fear" &
                                emo_measure == "emo_fear" ~ 1,
                              emotion == "happy" &
                                emo_measure == "emo_hapiness" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "emo_neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "emo_sadness" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "emo_surprise" ~ 1,
                              TRUE ~ 0))
rm(labdata_azure)

# Prototypical Data Face++
load(file = paste0(data, "labdata_facepp_emo_max.Rda"))

labdata_facepp_long <- labdata_facepp %>%
  select(fullpath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(fullpath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "emo_disgust" ~ 1,
                              emotion == "fear" &
                                emo_measure == "emo_fear" ~ 1,
                              emotion == "happy" &
                                emo_measure == "emo_hapiness" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "emo_neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "emo_sadness" ~ 1,
                              emotion == "surprise" &
```

```
                                emo_measure == "emo_surprise" ~ 1,
                              TRUE ~ 0))

rm(labdata_facepp)

# Prototypical Data FaceReader
load(file = paste0(data, "labdata_fr.Rda"))

labdata_fr_long <- labdata_fr %>%
  select(Filename2, emotion, Neutral, Happy, Sad, Angry, Surprised,
         Scared, Disgusted) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(Filename2, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "Angry" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "Disgusted" ~ 1,
                              emotion == "fear" &
                                emo_measure == "Scared" ~ 1,
                              emotion == "happy" &
                                emo_measure == "Happy" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "Neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "Sad" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "Surprised" ~ 1,
                              TRUE ~ 0))


rm(labdata_fr)

# create roc objects
labdata_facepp_roc <- roc(response = labdata_facepp_long$true_emo,
                          predictor = labdata_facepp_long$prediction_prob)

labdata_azure_roc <- roc(response = labdata_azure_long$true_emo,
                         predictor = labdata_azure_long$prediction_prob)

labdata_fr_roc <- roc(response = labdata_fr_long$true_emo,
                      predictor = labdata_fr_long$prediction_prob)

# run the tests

roc_test_lab_facepp_azure <- roc.test(roc1 = labdata_facepp_roc,
                                      roc2 = labdata_azure_roc,
                                      method = "bootstrap")

roc_test_lab_facepp_fr <- roc.test(roc1 = labdata_facepp_roc,
                                   roc2 = labdata_fr_roc,
```

```
                                    method = "bootstrap")

roc_test_lab_fr_azure <- roc.test(roc1 = labdata_fr_roc,
                                  roc2 = labdata_azure_roc,
                                  method = "bootstrap")
```

Loading the test-objects reported in the Dissertation

```
# load the test objects
load(file = paste0(data, "roc_test_lab_facepp_azure.Rda")) # p-value = 0.001253
load(file = paste0(data, "roc_test_lab_facepp_fr.Rda")) # p-value < 2.2e-16
load(file = paste0(data, "roc_test_lab_fr_azure.Rda")) # p-value < 2.2e-16

# inspect them
roc_test_lab_facepp_azure
```

```
##
##  Bootstrap test for two correlated ROC curves
##
## data:  labdata_facepp_roc and labdata_azure_roc
## D = -4.4542, boot.n = 2000, boot.stratified = 1, p-value = 8.423e-06
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.9641995   0.9752051
```

```
roc_test_lab_facepp_fr
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  labdata_facepp_roc and labdata_fr_roc
## D = -12.231, boot.n = 2000, boot.stratified = 1, p-value < 2.2e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.9641995   0.9975338
```

```
roc_test_lab_fr_azure
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  labdata_fr_roc and labdata_azure_roc
## D = 9.9153, boot.n = 2000, boot.stratified = 1, p-value < 2.2e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.9975338   0.9752051
```

```
rm(roc_test_lab_facepp_azure, roc_test_lab_facepp_fr, roc_test_lab_fr_azure)
```

Naturalistic Data

If you wish to replicate the full procedure mark the box with 'eval = TRUE' to do the following to: 1) Create long data frames from the wide ones 2) Create roc-objects from the long dataframe 3) run the tests Note, the tests are bootstrapped and therefore stochastic.

You can also jump to loading the test-objects reported in the Dissertation

```
# Naturalistic Data Azure
load(file = paste0(data, "sfew_azure_emo_max.Rda"))

sfew_azure_long <- sfew_azure %>%
  select(fullpath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(fullpath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "emo_disgust" ~ 1,
                              emotion == "fear" &
                                emo_measure == "emo_fear" ~ 1,
                              emotion == "happy" &
                                emo_measure == "emo_hapiness" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "emo_neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "emo_sadness" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "emo_surprise" ~ 1,
                              TRUE ~ 0))

rm(sfew_azure)

# Naturalistic Data Face++
load(file = paste0(data, "sfew_facepp_emo_max.Rda"))

sfew_facepp_long <- sfew_facepp %>%
  select(filepath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(filepath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
```

```
                                  emo_measure == "emo_disgust" ~ 1,
                                emotion == "fear" &
                                  emo_measure == "emo_fear" ~ 1,
                                emotion == "happy" &
                                  emo_measure == "emo_hapiness" ~ 1,
                                emotion == "neutral" &
                                  emo_measure == "emo_neutral" ~ 1,
                                emotion == "sad" &
                                  emo_measure == "emo_sadness" ~ 1,
                                emotion == "surprise" &
                                  emo_measure == "emo_surprise" ~ 1,
                                TRUE ~ 0))

rm(sfew_facepp)

# Naturalistic Data FaceReader
load(file = paste0(data, "sfew_fr.Rda"))

sfew_fr_long <- sfew_fr %>%
  select(Filename2, emotion, Neutral, Happy, Sad, Angry, Surprised, Scared,
         Disgusted) %>% # select needed variables
  #all except those variables named are observations for long format
  pivot_longer(-c(Filename2, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                  emo_measure == "Angry" ~ 1,
                                emotion == "disgust" &
                                  emo_measure == "Disgusted" ~ 1,
                                emotion == "fear" &
                                  emo_measure == "Scared" ~ 1,
                                emotion == "happy" &
                                  emo_measure == "Happy" ~ 1,
                                emotion == "neutral" &
                                  emo_measure == "Neutral" ~ 1,
                                emotion == "sad" &
                                  emo_measure == "Sad" ~ 1,
                                emotion == "surprise" &
                                  emo_measure == "Surprised" ~ 1,
                                TRUE ~ 0))

rm(sfew_fr)


# create ROC Objects
sfew_facepp_roc <- roc(response = sfew_facepp_long$true_emo,
                       predictor = sfew_facepp_long$prediction_prob)

sfew_azure_roc <- roc(response = sfew_azure_long$true_emo,
                      predictor = sfew_azure_long$prediction_prob)

sfew_fr_roc <- roc(response = sfew_fr_long$true_emo,
```

```
                  predictor = sfew_fr_long$prediction_prob)

# run the tests
roc_test_sfew_facepp_azure <- roc.test(roc1 = sfew_facepp_roc,
                                       roc2 = sfew_azure_roc,
                                       method = "bootstrap")

roc_test_sfew_facepp_fr <- roc.test(roc1 = sfew_facepp_roc,
                                    roc2 = sfew_fr_roc,
                                    method = "bootstrap")

roc_test_sfew_fr_azure <- roc.test(roc1 = sfew_fr_roc,
                                   roc2 = sfew_azure_roc,
                                   method = "bootstrap")
```

Loading the test-objects reported in the Dissertation

```
# load the tests
load(file = paste0(data, "roc_test_sfew_facepp_azure.Rda"))
load(file = paste0(data, "roc_test_sfew_facepp_fr.Rda"))
load(file = paste0(data, "roc_test_sfew_fr_azure.Rda"))

roc_test_sfew_facepp_azure
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  sfew_facepp_roc and sfew_azure_roc
## D = -16.336, boot.n = 2000, boot.stratified = 1, p-value < 2.2e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.6976288   0.8575833
```

```
roc_test_sfew_facepp_fr
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  sfew_facepp_roc and sfew_fr_roc
## D = -0.91551, boot.n = 2000, boot.stratified = 1, p-value = 0.3599
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.6976288   0.7119290
```

```
roc_test_sfew_fr_azure
```

```
##
##  Bootstrap test for two ROC curves
```

```
##
## data:  sfew_fr_roc and sfew_azure_roc
## D = -9.4415, boot.n = 2000, boot.stratified = 1, p-value < 2.2e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.7119290   0.8575833
```

```
rm(roc_test_sfew_facepp_azure, roc_test_sfew_facepp_fr, roc_test_sfew_fr_azure)
```

Chi-Square Tests for Accuracies

Prototypical Data

```
# Azure
load(file = paste0(data, "labdata_azure_emo_max.Rda"))
labdata_azure_face_found <- subset(labdata_azure, !is.na(emo_neutral))
# sum of correct images
count_labdata_azure <- sum(labdata_azure_face_found$emo_azure_max_char ==
                             labdata_azure_face_found$emotion)

# Facepp
load(file = paste0(data, "labdata_facepp_emo_max.Rda"))
labdata_facepp_face_found <- subset(labdata_facepp, !is.na(emo_neutral))
count_labdata_facepp <- sum(labdata_facepp_face_found$emo_fpp_max_char ==
                              labdata_facepp_face_found$emotion)

# FaceReader
load(file = paste0(data, "labdata_fr.Rda"))
labdata_fr <- filter(labdata_fr, Quality>0.7)
labdata_fr_face_found <- subset(labdata_fr, !is.na(Neutral))
# sum of correct images
count_labdata_fr <- sum(labdata_fr_face_found$emo_fr_max_char ==
                          labdata_fr_face_found$emotion)

prop.test(x = c(count_labdata_facepp, count_labdata_azure),
          n = c(nrow(labdata_facepp_face_found), nrow(labdata_azure_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_labdata_facepp, count_labdata_azure) out of c(nrow(labdata_facepp_face_found), nrow(la
## X-squared = 2.0917, df = 1, p-value = 0.1481
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.056406738  0.008252645
## sample estimates:
##    prop 1    prop 2
## 0.7857143 0.8097913
```

```r
prop.test(x = c(count_labdata_facepp, count_labdata_fr),
          n = c(nrow(labdata_facepp_face_found), nrow(labdata_fr_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_labdata_facepp, count_labdata_fr) out of c(nrow(labdata_facepp_face_found), nrow(labda
## X-squared = 185.4, df = 1, p-value < 2.2e-16
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.2067715 -0.1554031
## sample estimates:
##    prop 1    prop 2
## 0.7857143 0.9668016
```

```r
prop.test(x = c(count_labdata_fr, count_labdata_azure),
          n = c(nrow(labdata_fr_face_found), nrow(labdata_azure_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_labdata_fr, count_labdata_azure) out of c(nrow(labdata_fr_face_found), nrow(labdata_az
## X-squared = 152.1, df = 1, p-value < 2.2e-16
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  0.1322310 0.1817896
## sample estimates:
##    prop 1    prop 2
## 0.9668016 0.8097913
```

```r
rm(labdata_azure, labdata_azure_face_found, labdata_facepp,
   labdata_facepp_face_found, labdata_fr, labdata_fr_face_found,
   count_labdata_fr, count_labdata_facepp, count_labdata_azure)
```

Naturalistic Data

```r
# Azure
load(file = paste0(data, "sfew_azure_emo_max.Rda"))
sfew_azure_face_found <- subset(sfew_azure, !is.na(emo_neutral))
# sum of correct images
count_sfew_azure <- sum(sfew_azure_face_found$emo_azure_max_char ==
                          sfew_azure_face_found$emotion)

# Facepp
load(file = paste0(data, "sfew_facepp_emo_max.Rda"))
sfew_facepp_face_found <- subset(sfew_facepp, !is.na(emo_neutral))
# sum of correct images
count_sfew_facepp <- sum(sfew_facepp_face_found$emo_fpp_max_char ==
                           sfew_facepp_face_found$emotion)
```

```r
# FaceReader
load(file = paste0(data, "sfew_fr.Rda"))
sfew_fr <- filter(sfew_fr, Quality>0.7)
sfew_fr_face_found <- subset(sfew_fr, !is.na(Neutral))
# sum of correct images
count_sfew_fr <- sum(sfew_fr_face_found$emo_fr_max_char ==
                       sfew_fr_face_found$emotion)


prop.test(x = c(count_sfew_facepp, count_sfew_azure),
          n = c(nrow(sfew_facepp_face_found), nrow(sfew_azure_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_sfew_facepp, count_sfew_azure) out of c(nrow(sfew_facepp_face_found), nrow(sfew_azure
## X-squared = 153.43, df = 1, p-value < 2.2e-16
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.2875983 -0.2095556
## sample estimates:
##    prop 1    prop 2
## 0.3224638 0.5710407
```

```r
prop.test(x = c(count_sfew_facepp, count_sfew_fr),
          n = c(nrow(sfew_facepp_face_found), nrow(sfew_fr_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_sfew_facepp, count_sfew_fr) out of c(nrow(sfew_facepp_face_found), nrow(sfew_fr_face_
## X-squared = 0.14816, df = 1, p-value = 0.7003
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.04323635  0.06805215
## sample estimates:
##    prop 1    prop 2
## 0.3224638 0.3100559
```

```r
prop.test(x = c(count_sfew_fr, count_sfew_azure),
          n = c(nrow(sfew_fr_face_found), nrow(sfew_azure_face_found)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_sfew_fr, count_sfew_azure) out of c(nrow(sfew_fr_face_found), nrow(sfew_azure_face_fou
## X-squared = 72.645, df = 1, p-value < 2.2e-16
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.3189322 -0.2030375
## sample estimates:
##    prop 1    prop 2
## 0.3100559 0.5710407
```

```
rm(sfew_azure, sfew_azure_face_found, sfew_facepp, sfew_facepp_face_found,
   sfew_fr, sfew_fr_face_found, count_sfew_fr, count_sfew_facepp,
   count_sfew_azure)
```

Appendix: Subset of Commonly Recognized Images of Naturalistic Data

Sensitivity, Precision & Accuracy

Azure

```
load(file = paste0(data, "sfew_azure_detected_all.Rda"))

emo_order <- c("neutral", "happy", "angry", "disgust", "surprise", "sad", "fear")

confmat <- confusionMatrix(data =
                             factor(
                               as.factor(sfew_azure_detected_all$emo_azure_max_char),
                               levels = emo_order),
                           reference = factor(sfew_azure_detected_all$emotion,
                                              levels = emo_order),
                            mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neutral happy angry disgust surprise sad fear
##   neutral       56     5    24      16       12  23   12
##   happy          1    75     2       5        2   2    1
##   angry          0     0    18       1        1   0    2
##   disgust        0     0     6       3        0   0    0
##   surprise       2     1     4       0       18   2   10
##   sad            1     0     3       1        1  29    3
##   fear           0     0     0       0        0   3    0
##
## Overall Statistics
##
##                Accuracy : 0.5768
##                  95% CI : (0.5228, 0.6295)
##     No Information Rate : 0.2348
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4857
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: neutral Class: happy Class: angry Class: disgust
## Sensitivity                  0.9333       0.9259      0.31579       0.115385
## Specificity                  0.6772       0.9508      0.98611       0.981191
## Pos Pred Value               0.3784       0.8523      0.81818       0.333333
## Neg Pred Value               0.9797       0.9767      0.87926       0.931548
## Precision                    0.3784       0.8523      0.81818       0.333333
## Recall                       0.9333       0.9259      0.31579       0.115385
## F1                           0.5385       0.8876      0.45570       0.171429
## Prevalence                   0.1739       0.2348      0.16522       0.075362
## Detection Rate               0.1623       0.2174      0.05217       0.008696
## Detection Prevalence         0.4290       0.2551      0.06377       0.026087
## Balanced Accuracy            0.8053       0.9383      0.65095       0.548288
##                      Class: surprise Class: sad Class: fear
## Sensitivity                  0.52941    0.49153    0.000000
## Specificity                  0.93891    0.96853    0.990536
## Pos Pred Value               0.48649    0.76316    0.000000
## Neg Pred Value               0.94805    0.90228    0.918129
## Precision                    0.48649    0.76316    0.000000
## Recall                       0.52941    0.49153    0.000000
## F1                           0.50704    0.59794         NaN
## Prevalence                   0.09855    0.17101    0.081159
## Detection Rate               0.05217    0.08406    0.000000
## Detection Prevalence         0.10725    0.11014    0.008696
## Balanced Accuracy            0.73416    0.73003    0.495268
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.4730529
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.5188301
```

```
rm(confmat, emo_order)
```

Face++

```
load(file = paste0(data, "sfew_facepp_detected_all.Rda"))

emo_order <- c("neutral", "happy", "angry", "disgust", "surprise", "sad", "fear")

confmat <- confusionMatrix(data =
                             factor(as.factor(
                               sfew_facepp_detected_all$emo_fpp_max_char),
                               levels = levels(sfew_facepp_detected_all$emotion)),
                           reference = sfew_facepp_detected_all$emotion,
```

```
                          mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction angry disgust fear happy neutral sad surprise
##   angry        6       4    1     3       1   0        1
##   disgust      9       3    1     5       0   1        1
##   fear         7       3    2     8       4   6        2
##   happy        1       3    2    41       1   5        1
##   neutral      7       7    4     5      23  12        4
##   sad          4       0    2     5       9   9        0
##   surprise    23       6   16    14      22  26       25
##
## Overall Statistics
##
##                Accuracy : 0.3159
##                  95% CI : (0.2672, 0.3679)
##     No Information Rate : 0.2348
##     P-Value [Acc > NIR] : 0.0003542
##
##                   Kappa : 0.2049
##
##  Mcnemar's Test P-Value : 1.36e-15
##
## Statistics by Class:
##
##                      Class: angry Class: disgust Class: fear Class: happy
## Sensitivity               0.10526       0.115385    0.071429       0.5062
## Specificity               0.96528       0.946708    0.905363       0.9508
## Pos Pred Value            0.37500       0.150000    0.062500       0.7593
## Neg Pred Value            0.84498       0.929231    0.916933       0.8625
## Precision                 0.37500       0.150000    0.062500       0.7593
## Recall                    0.10526       0.115385    0.071429       0.5062
## F1                        0.16438       0.130435    0.066667       0.6074
## Prevalence                0.16522       0.075362    0.081159       0.2348
## Detection Rate            0.01739       0.008696    0.005797       0.1188
## Detection Prevalence      0.04638       0.057971    0.092754       0.1565
## Balanced Accuracy         0.53527       0.531047    0.488396       0.7285
##                      Class: neutral Class: sad Class: surprise
## Sensitivity                 0.38333    0.15254         0.73529
## Specificity                 0.86316    0.93007         0.65595
## Pos Pred Value              0.37097    0.31034         0.18939
## Neg Pred Value              0.86926    0.84177         0.95775
## Precision                   0.37097    0.31034         0.18939
## Recall                      0.38333    0.15254         0.73529
## F1                          0.37705    0.20455         0.30120
## Prevalence                  0.17391    0.17101         0.09855
## Detection Rate              0.06667    0.02609         0.07246
## Detection Prevalence        0.17971    0.08406         0.38261
## Balanced Accuracy           0.62325    0.54131         0.69562
```

```r
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.2956313
```

```r
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.3167808
```

```r
rm(confmat, emo_order)
```

FaceReader

```r
load(file = paste0(data, "sfew_fr_detected_all.Rda"))

confmat <- confusionMatrix(data =
                             factor(as.factor(sfew_fr_detected_all$emo_fr_max_char),
                                    levels = levels(sfew_fr_detected_all$emotion)),
                           reference = sfew_fr_detected_all$emotion,
                           mode = "everything")

confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction angry disgust fear happy neutral sad surprise
##   angry        8       4    0     2       2   0        1
##   disgust     13       4    0     4       0   1        1
##   fear         1       0    0     1       0   5        0
##   happy        1       1    0    34       0   1        0
##   neutral     25      13   20    38      41  37       18
##   sad          3       4    1     0      11  10        2
##   surprise     6       0    7     2       6   5       12
##
## Overall Statistics
##
##                Accuracy : 0.3159
##                  95% CI : (0.2672, 0.3679)
##     No Information Rate : 0.2348
##     P-Value [Acc > NIR] : 0.0003542
##
##                   Kappa : 0.1827
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: angry Class: disgust Class: fear Class: happy
## Sensitivity               0.14035        0.15385     0.00000      0.41975
```

```
## Specificity               0.96875        0.94044     0.97792      0.98864
## Pos Pred Value            0.47059        0.17391     0.00000      0.91892
## Neg Pred Value            0.85061        0.93168     0.91716      0.84740
## Precision                 0.47059        0.17391     0.00000      0.91892
## Recall                    0.14035        0.15385     0.00000      0.41975
## F1                        0.21622        0.16327         NaN      0.57627
## Prevalence                0.16522        0.07536     0.08116      0.23478
## Detection Rate            0.02319        0.01159     0.00000      0.09855
## Detection Prevalence      0.04928        0.06667     0.02029      0.10725
## Balanced Accuracy         0.55455        0.54714     0.48896      0.70419
##                       Class: neutral Class: sad Class: surprise
## Sensitivity                   0.6833    0.16949         0.35294
## Specificity                   0.4702    0.92657         0.91640
## Pos Pred Value                0.2135    0.32258         0.31579
## Neg Pred Value                0.8758    0.84395         0.92834
## Precision                     0.2135    0.32258         0.31579
## Recall                        0.6833    0.16949         0.35294
## F1                            0.3254    0.22222         0.33333
## Prevalence                    0.1739    0.17101         0.09855
## Detection Rate                0.1188    0.02899         0.03478
## Detection Prevalence          0.5565    0.08986         0.11014
## Balanced Accuracy             0.5768    0.54803         0.63467
```

```
# average sensitivity and precision
mean(data.frame(confmat$byClass)$Sensitivity)
```

```
## [1] 0.2742452
```

```
mean(data.frame(confmat$byClass)$Precision, na.rm = T)
```

```
## [1] 0.3450474
```

```
rm(confmat)
```

Chi-Square Tests for Accuracies

```
count_sfew_sample_azure <- sum(sfew_azure_detected_all$emo_azure_max_char ==
                                 sfew_azure_detected_all$emotion)

load(file = paste0(data, "sfew_facepp_detected_all.Rda"))
count_sfew_sample_facepp <- sum(sfew_facepp_detected_all$emo_fpp_max_char ==
                                  sfew_facepp_detected_all$emotion)

load(file = paste0(data, "sfew_fr_detected_all.Rda"))
count_sfew_sample_fr <- sum(sfew_fr_detected_all$emo_fr_max_char ==
                              sfew_fr_detected_all$emotion)

prop.test(x = c(count_sfew_sample_facepp, count_sfew_sample_azure),
          n = c(nrow(sfew_facepp_detected_all), nrow(sfew_azure_detected_all)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_sfew_sample_facepp, count_sfew_sample_azure) out of c(nrow(sfew_facepp_detected_all),
## X-squared = 46.453, df = 1, p-value = 9.384e-12
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.3353532 -0.1863859
## sample estimates:
##    prop 1    prop 2
## 0.3159420 0.5768116
```

```
prop.test(x = c(count_sfew_sample_facepp, count_sfew_sample_fr),
          n = c(nrow(sfew_facepp_detected_all), nrow(sfew_fr_detected_all)))
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(count_sfew_sample_facepp, count_sfew_sample_fr) out of c(nrow(sfew_facepp_detected_all), nr
## X-squared = 3.7055e-30, df = 1, p-value = 1
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.06937511  0.06937511
## sample estimates:
##   prop 1   prop 2
## 0.315942 0.315942
```

```
prop.test(x = c(count_sfew_sample_fr, count_sfew_sample_azure),
          n = c(nrow(sfew_fr_detected_all), nrow(sfew_azure_detected_all)))
```

```
##
##  2-sample test for equality of proportions with continuity correction
##
## data:  c(count_sfew_sample_fr, count_sfew_sample_azure) out of c(nrow(sfew_fr_detected_all), nrow(sf
## X-squared = 46.453, df = 1, p-value = 9.384e-12
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.3353532 -0.1863859
## sample estimates:
##    prop 1    prop 2
## 0.3159420 0.5768116
```

```
rm(count_sfew_sample_fr, count_sfew_sample_facepp, count_sfew_sample_azure)
```

ROC Analysis

If you wish to replicate the full procedure mark the box with 'eval = TRUE' to do the following to: 1) Create long data frames from the wide ones 2) Create roc-objects from the long dataframe 3) run the tests Note, the tests are bootstrapped and therefore stochastic.

You can also jump to loading the test-objects reported in the Dissertation

```r
# Azure
sfew_azure_detected_all_long <- sfew_azure_detected_all %>%
  select(fullpath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  pivot_longer(-c(fullpath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "emo_disgust" ~ 1,
                              emotion == "fear" &
                                emo_measure == "emo_fear" ~ 1,
                              emotion == "happy" &
                                emo_measure == "emo_hapiness" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "emo_neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "emo_sadness" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "emo_surprise" ~ 1,
                              TRUE ~ 0))

# Face++
sfew_facepp_detected_all_long <- sfew_facepp_detected_all %>%
  select(filepath, emo_anger, emo_disgust, emo_fear, emo_hapiness, emo_neutral,
         emo_sadness, emo_surprise, emotion) %>% # select needed variables
  pivot_longer(-c(filepath, emotion),
               names_to = "emo_measure",
               values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" &
                                emo_measure == "emo_anger" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "emo_disgust" ~ 1,
                              emotion == "fear" &
                                emo_measure == "emo_fear" ~ 1,
                              emotion == "happy" &
                                emo_measure == "emo_hapiness" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "emo_neutral" ~ 1,
                              emotion == "sad" &
                                emo_measure == "emo_sadness" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "emo_surprise" ~ 1,
                              TRUE ~ 0))

# FaceReader
sfew_fr_detected_all_long <- sfew_fr_detected_all %>%
  select(Filename2, emotion, Neutral, Happy, Sad, Angry, Surprised, Scared,
         Disgusted) %>% # select needed variables
  pivot_longer(-c(Filename2, emotion),
```

```
              names_to = "emo_measure",
              values_to = "prediction_prob") %>%
  # create a variable that indicates the true emotion for each images
  mutate(true_emo = case_when(emotion == "angry" & emo_measure == "Angry" ~ 1,
                              emotion == "disgust" &
                                emo_measure == "Disgusted" ~ 1,
                              emotion == "fear" & emo_measure == "Scared" ~ 1,
                              emotion == "happy" & emo_measure == "Happy" ~ 1,
                              emotion == "neutral" &
                                emo_measure == "Neutral" ~ 1,
                              emotion == "sad" & emo_measure == "Sad" ~ 1,
                              emotion == "surprise" &
                                emo_measure == "Surprised" ~ 1,
                              TRUE ~ 0))


# create roc objects
sfew_sample_facepp_roc <- roc(response = sfew_facepp_detected_all_long$true_emo,
                       predictor = sfew_facepp_detected_all_long$prediction_prob)

sfew_sample_azure_roc <- roc(response = sfew_azure_detected_all_long$true_emo,
                      predictor = sfew_azure_detected_all_long$prediction_prob)

sfew_sample_fr_roc <- roc(response = sfew_fr_detected_all_long$true_emo,
                  predictor = sfew_fr_detected_all_long$prediction_prob)

# run the tests
#
roc_test_sfew_sample_facepp_azure <- roc.test(roc1 = sfew_sample_facepp_roc,
                                      roc2 = sfew_sample_azure_roc,
                                      method = "bootstrap")

#
roc_test_sfew_sample_facepp_fr <- roc.test(roc1 = sfew_sample_facepp_roc,
                                    roc2 = sfew_sample_fr_roc,
                                    method = "bootstrap")

#
roc_test_sfew_sample_fr_azure <- roc.test(roc1 = sfew_sample_fr_roc,
                                  roc2 = sfew_sample_azure_roc,
                                  method = "bootstrap")
rm(sfew_azure_detected_all_long, sfew_facepp_detected_all_long,
   sfew_fr_detected_all_long)

rm(sfew_azure_detected_all, sfew_facepp_detected_all, sfew_fr_detected_all)
```

Loading the test-objects reported in the Dissertation

```
# load the tests
load(file = paste0(data, "roc_test_sfew_sample_facepp_azure.Rda"))
load(file = paste0(data, "roc_test_sfew_sample_facepp_fr.Rda"))
load(file = paste0(data, "roc_test_sfew_sample_fr_azure.Rda"))
```

```
roc_test_sfew_sample_facepp_azure
```

```
##
##  Bootstrap test for two correlated ROC curves
##
## data:  sfew_sample_facepp_roc and sfew_sample_azure_roc
## D = -10.099, boot.n = 2000, boot.stratified = 1, p-value < 2.2e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.7065154   0.8628215
```

```
roc_test_sfew_sample_facepp_fr
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  sfew_sample_facepp_roc and sfew_sample_fr_roc
## D = -0.5234, boot.n = 2000, boot.stratified = 1, p-value = 0.6007
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.7065154   0.7173507
```

```
roc_test_sfew_sample_fr_azure
```

```
##
##  Bootstrap test for two ROC curves
##
## data:  sfew_sample_fr_roc and sfew_sample_azure_roc
## D = -8.1132, boot.n = 2000, boot.stratified = 1, p-value = 4.931e-16
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.7173507   0.8628215
```

```
rm(roc_test_sfew_sample_facepp_azure, roc_test_sfew_sample_facepp_fr,
   roc_test_sfew_sample_fr_azure)
```

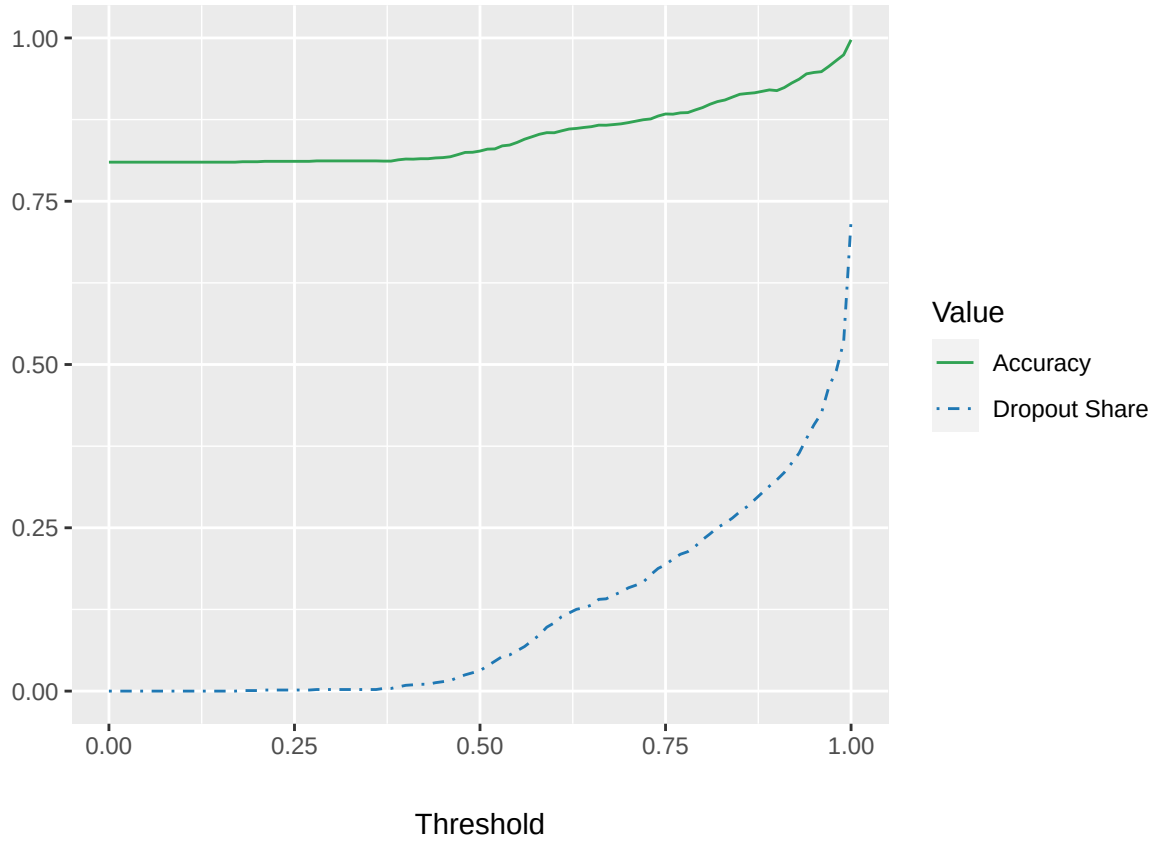
Appendix: Thresholds

Compare Prototypical Data to Azure

```
load(file = paste0(data, "labdata_azure_emo_max.Rda"))

# Tresholding
tresholds_labdata_azure <- data.frame(threshold = seq(from = 0, to = 1,
```

```
                                                  by = 0.01),
                                      accuracy = as.numeric(NA),
                                      dropout = as.numeric(NA),
                                      dropout_share = as.numeric(NA))

my_accuracy <- function(threshold) {
  overview_df <- labdata_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                                  "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  accuracy <- sum(overview_df$emo_azure_th==overview_df$emotion, na.rm = T)/
    sum(!is.na(overview_df$emo_azure_th))
  return(accuracy)
}

my_dropout <- function(threshold) {
  overview_df <- labdata_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                                  "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  dropout <- sum(is.na(overview_df$emo_azure_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- labdata_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                                  "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  dropout_share <- sum(is.na(overview_df$emo_azure_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_labdata_azure$accuracy <- unlist(map(.x =
                                                 tresholds_labdata_azure$threshold,
                                               .f = my_accuracy))
tresholds_labdata_azure$dropout <- unlist(map(.x =
                                                tresholds_labdata_azure$threshold,
                                              .f = my_dropout))
tresholds_labdata_azure$dropout_share <- unlist(
  map(.x = tresholds_labdata_azure$threshold,
      .f = my_dropout_share))

gather(tresholds_labdata_azure[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
```

```
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```



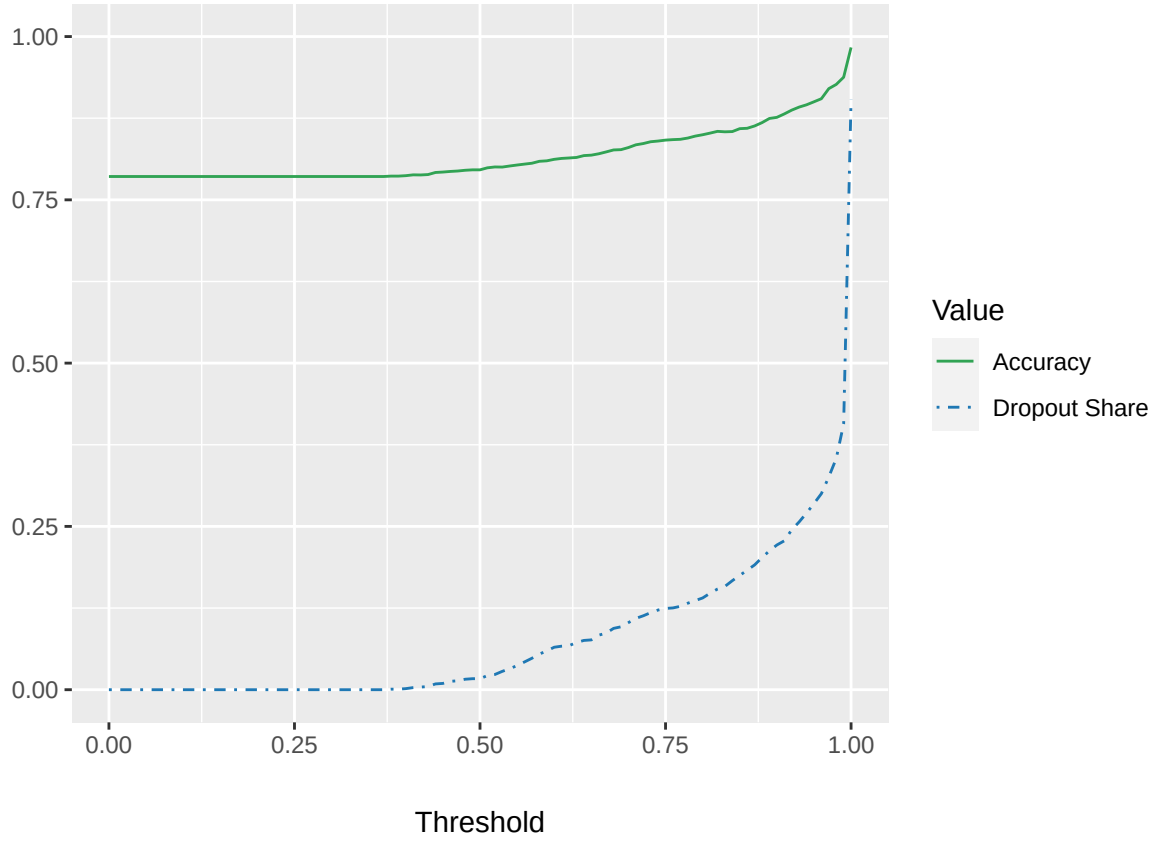

```
rm(labdata_azure, tresholds_labdata_azure, my_accuracy,
   my_dropout, my_dropout_share)
```

Compare Prototypical Data to Face++

```
load(file = paste0(data, "labdata_facepp_emo_max.Rda"))

tresholds_labdata_facepp <- data.frame(threshold = seq(from = 0, to = 1,
                                                       by = 0.01),
                                       accuracy = as.numeric(NA),
                                       dropout = as.numeric(NA),
                                       dropout_share = as.numeric(NA))
```

```
my_accuracy <- function(threshold) {
  overview_df <- labdata_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                   "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  accuracy <- sum(overview_df$emo_fpp_th==overview_df$emotion, na.rm = T)/
    sum(!is.na(overview_df$emo_fpp_th))
  return(accuracy)
}

my_dropout <- function(threshold) {
  overview_df <- labdata_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                   "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  dropout <- sum(is.na(overview_df$emo_fpp_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- labdata_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                   "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  dropout_share <- sum(is.na(overview_df$emo_fpp_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_labdata_facepp$accuracy <-
  unlist(map(.x = tresholds_labdata_facepp$threshold,
                                                .f = my_accuracy))
tresholds_labdata_facepp$dropout <-
  unlist(map(.x = tresholds_labdata_facepp$threshold,
                                               .f = my_dropout))
tresholds_labdata_facepp$dropout_share <-
  unlist(map(.x = tresholds_labdata_facepp$threshold,
                                                    .f = my_dropout_share))

gather(tresholds_labdata_facepp[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
```

```
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```



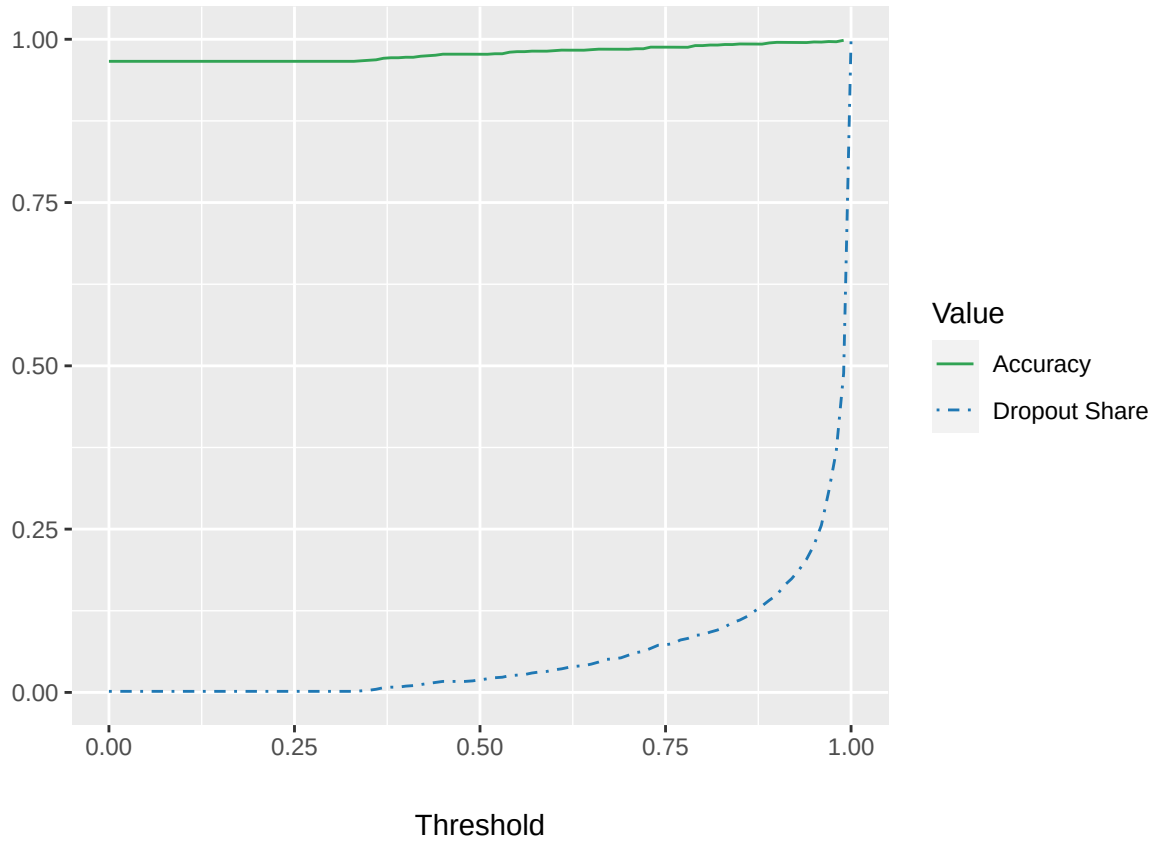

```
rm(labdata_facepp, tresholds_labdata_facepp, my_accuracy,
   my_dropout, my_dropout_share)
```

Compare Prototypical Data to FaceReader

```
load(file = paste0(data, "labdata_fr.Rda"))

tresholds_labdata_fr <- data.frame(threshold = seq(from = 0, to = 1, by = 0.01),
                                   accuracy = as.numeric(NA),
                                   dropout = as.numeric(NA),
                                   dropout_share = as.numeric(NA))

my_accuracy <- function(threshold) {
  overview_df <- labdata_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                               "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
```

```
  ))
  accuracy <- sum(as.character(overview_df$emo_fr_th) ==
                    as.character(overview_df$emotion), na.rm = T)/
    sum(!is.na(overview_df$emo_fr_th))
  return(accuracy)
}

my_dropout <- function(threshold) {
  overview_df <- labdata_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                               "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
  ))
  dropout <- sum(is.na(overview_df$emo_fr_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- labdata_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                               "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
  ))
  dropout_share <- sum(is.na(overview_df$emo_fr_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_labdata_fr$accuracy <- unlist(map(.x = tresholds_labdata_fr$threshold,
                                            .f = my_accuracy))
tresholds_labdata_fr$dropout <- unlist(map(.x = tresholds_labdata_fr$threshold,
                                           .f = my_dropout))
tresholds_labdata_fr$dropout_share <-
  unlist(map(.x = tresholds_labdata_fr$threshold,
                                                .f = my_dropout_share))

gather(tresholds_labdata_fr[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

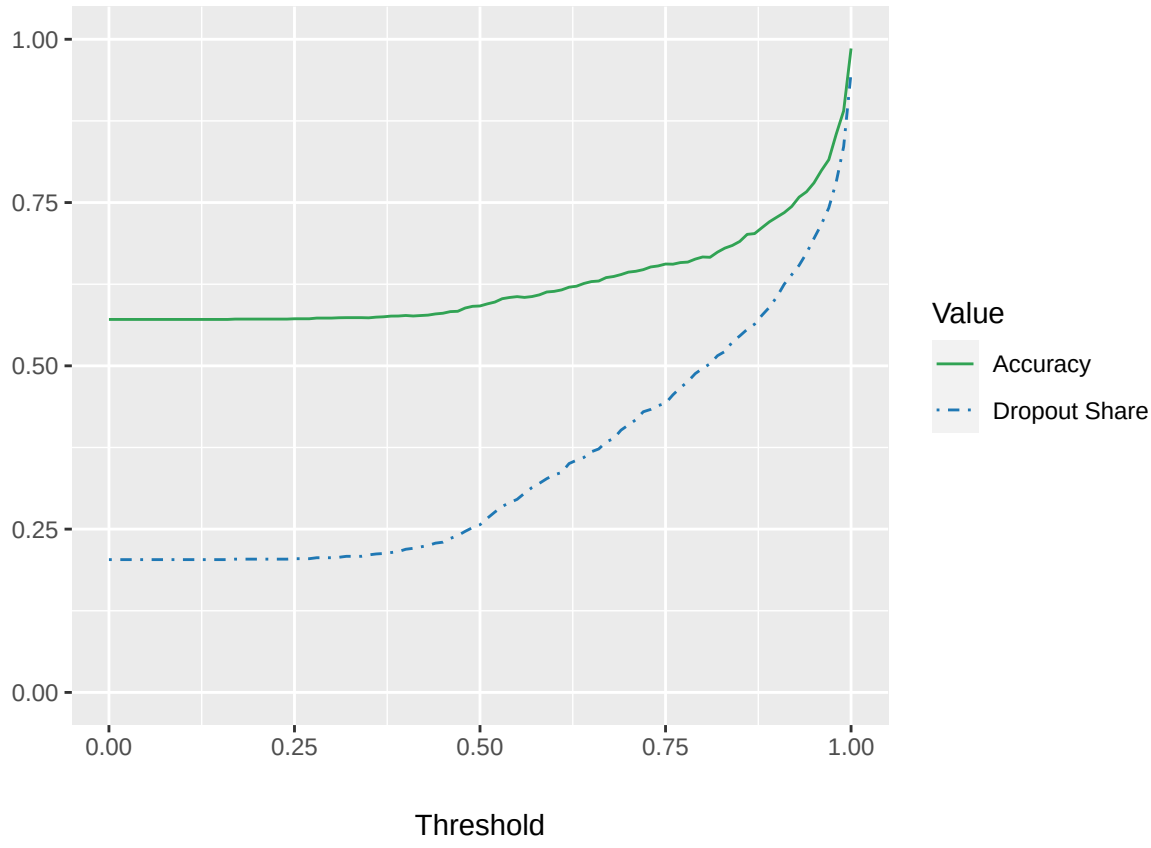



```
rm(labdata_fr, tresholds_labdata_fr, my_accuracy,
   my_dropout, my_dropout_share)
```

Compare Naturalistic Data to Azure

```
load(file = paste0(data, "sfew_azure_emo_max.Rda"))

tresholds_sfew_azure <- data.frame(threshold = seq(from = 0, to = 1, by = 0.01),
                                   accuracy = as.numeric(NA),
                                   dropout = as.numeric(NA),
                                   dropout_share = as.numeric(NA))

my_accuracy <- function(threshold) {
  overview_df <- sfew_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                               "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  accuracy <- sum(overview_df$emo_azure_th==overview_df$emotion, na.rm = T)/
    sum(!is.na(overview_df$emo_azure_th))
  return(accuracy)
}
```

```r
my_dropout <- function(threshold) {
  overview_df <- sfew_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                               "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  dropout <- sum(is.na(overview_df$emo_azure_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- sfew_azure[,c("emo_azure_max_char", "emo_azure_max_value",
                               "emotion")] %>%
  mutate(emo_azure_th = case_when(
    emo_azure_max_value >= threshold ~ emo_azure_max_char,
    TRUE ~ NA_character_
  ))
  dropout_share <- sum(is.na(overview_df$emo_azure_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_sfew_azure$accuracy <-
  unlist(map(.x = tresholds_sfew_azure$threshold, .f = my_accuracy))
tresholds_sfew_azure$dropout <-
  unlist(map(.x = tresholds_sfew_azure$threshold, .f = my_dropout))
tresholds_sfew_azure$dropout_share <-
  unlist(map(.x = tresholds_sfew_azure$threshold, .f = my_dropout_share))

gather(tresholds_sfew_azure[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```

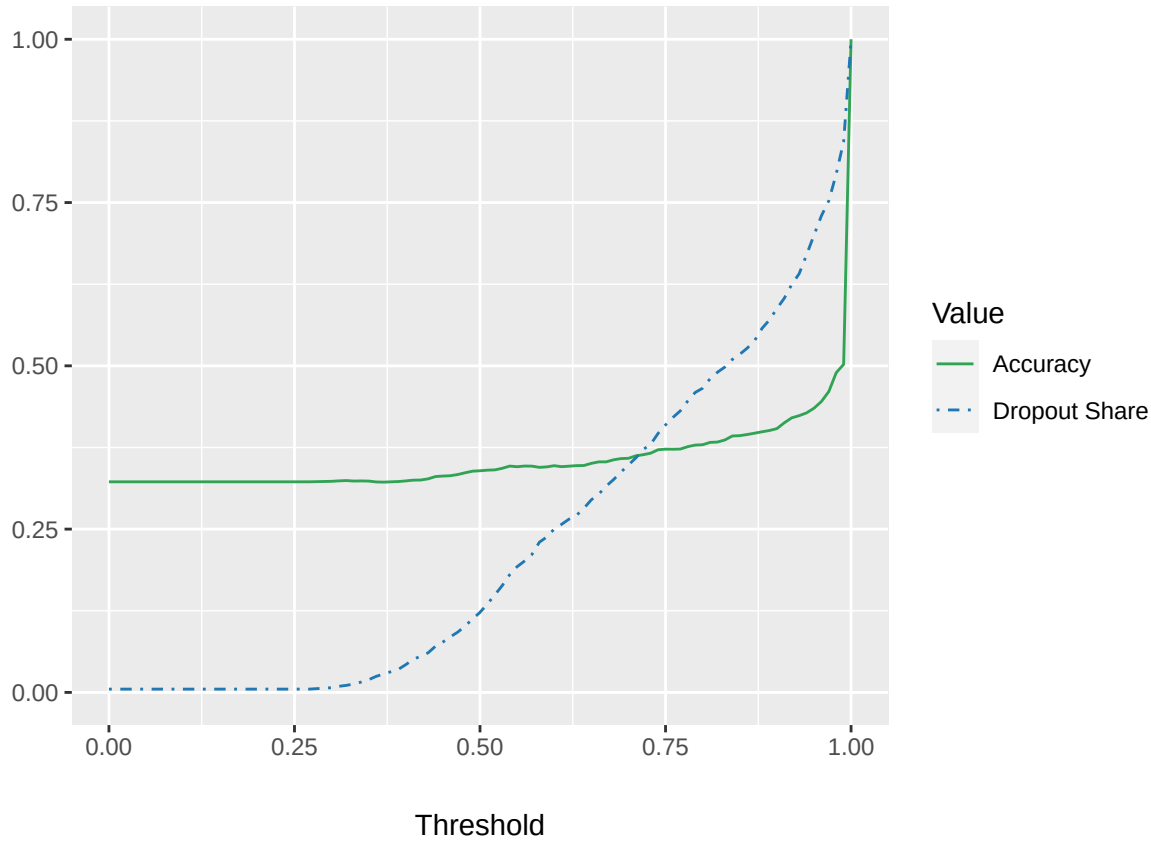



```
rm(sfew_azure, tresholds_sfew_azure, my_accuracy,
   my_dropout, my_dropout_share)
```

Compare Naturalistic Data to Face++

```
load(file = paste0(data, "sfew_facepp_emo_max.Rda"))

tresholds_sfew_facepp <- data.frame(threshold = seq(from = 0, to = 1, by = 0.01),
                                    accuracy = as.numeric(NA),
                                    dropout = as.numeric(NA),
                                    dropout_share = as.numeric(NA))

my_accuracy <- function(threshold) {
  overview_df <- sfew_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  accuracy <- sum(overview_df$emo_fpp_th==overview_df$emotion, na.rm = T)/
    sum(!is.na(overview_df$emo_fpp_th))
  return(accuracy)
}
```

```r
my_dropout <- function(threshold) {
  overview_df <- sfew_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  dropout <- sum(is.na(overview_df$emo_fpp_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- sfew_facepp[,c("emo_fpp_max_char", "emo_fpp_max_value",
                                "emotion")] %>%
  mutate(emo_fpp_th = case_when(
    emo_fpp_max_value/100 >= threshold ~ emo_fpp_max_char,
    TRUE ~ NA_character_
  ))
  dropout_share <- sum(is.na(overview_df$emo_fpp_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_sfew_facepp$accuracy <-
  unlist(map(.x = tresholds_sfew_facepp$threshold, .f = my_accuracy))
tresholds_sfew_facepp$dropout <-
  unlist(map(.x = tresholds_sfew_facepp$threshold, .f = my_dropout))
tresholds_sfew_facepp$dropout_share <-
  unlist(map(.x = tresholds_sfew_facepp$threshold, .f = my_dropout_share))

gather(tresholds_sfew_facepp[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```



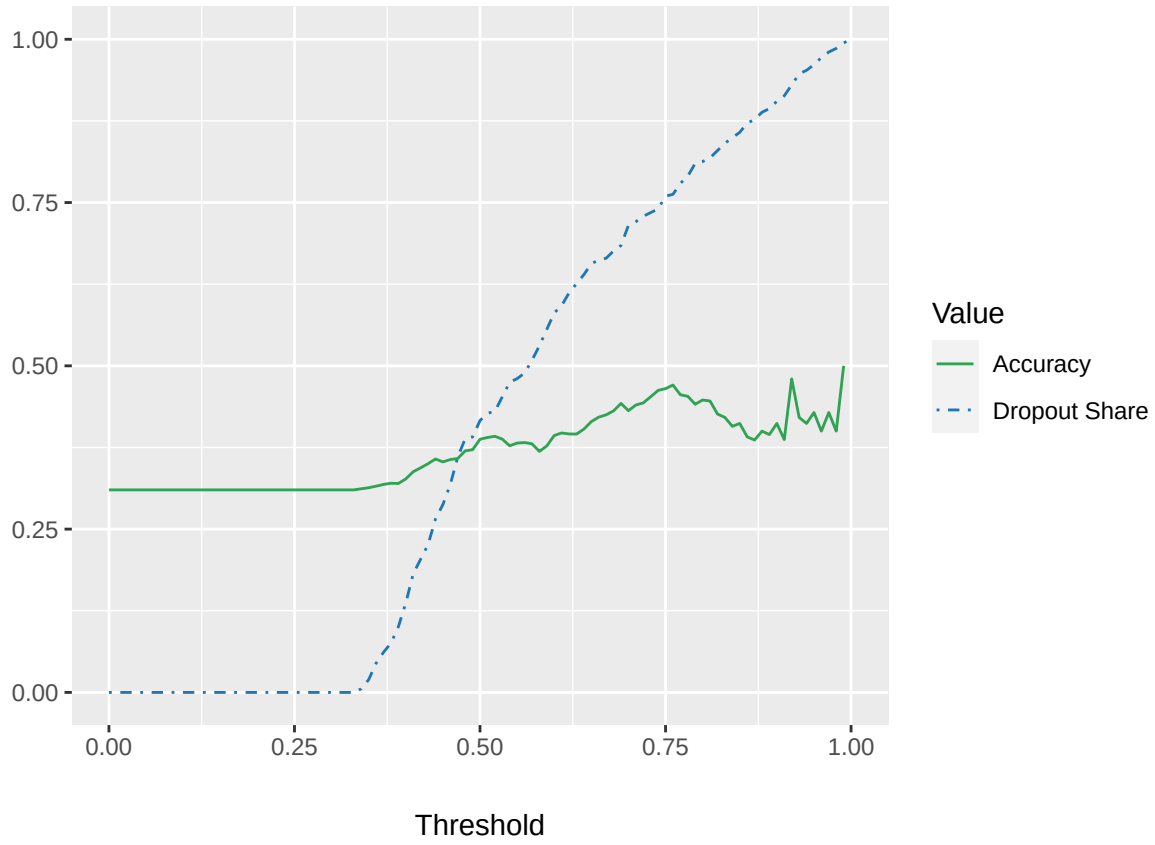

```
rm(sfew_facepp, tresholds_sfew_facepp, my_accuracy,
   my_dropout, my_dropout_share)
```

Compare Naturalistic Data to FaceReader

```
load(file = paste0(data, "sfew_fr.Rda"))

# keep those with qual >= .7 for the analysis
sfew_fr <- filter(sfew_fr, Quality >=0.7)

tresholds_sfew_fr <- data.frame(threshold = seq(from = 0, to = 1, by = 0.01),
                                accuracy = as.numeric(NA),
                                dropout = as.numeric(NA),
                                dropout_share = as.numeric(NA))

my_accuracy <- function(threshold) {
  overview_df <- sfew_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                            "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
  ))
  accuracy <- sum(overview_df$emo_fr_th==overview_df$emotion, na.rm = T)/
```

```
    sum(!is.na(overview_df$emo_fr_th))
  return(accuracy)
}

my_dropout <- function(threshold) {
  overview_df <- sfew_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                            "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
  ))
  dropout <- sum(is.na(overview_df$emo_fr_th))
  return(dropout)
}

my_dropout_share <- function(threshold) {
  overview_df <- sfew_fr[,c("emo_fr_max_char", "emo_fr_max_value",
                            "emotion")] %>%
  mutate(emo_fr_th = case_when(
    emo_fr_max_value >= threshold ~ emo_fr_max_char,
    TRUE ~ NA_integer_
  ))
  dropout_share <- sum(is.na(overview_df$emo_fr_th))/nrow(overview_df)
  return(dropout_share)
}

tresholds_sfew_fr$accuracy <-
  unlist(map(.x = tresholds_sfew_fr$threshold, .f = my_accuracy))
tresholds_sfew_fr$dropout <-
  unlist(map(.x = tresholds_sfew_fr$threshold, .f = my_dropout))
tresholds_sfew_fr$dropout_share <-
  unlist(map(.x = tresholds_sfew_fr$threshold, .f = my_dropout_share))

gather(tresholds_sfew_fr[c("threshold", "dropout_share", "accuracy")],
       "type", "value", 2:3) %>%
ggplot(data = ., aes(x = threshold, y = value, group = type)) +
  geom_line(aes(linetype = type, color = type)) +
  scale_linetype_manual(name = "Value", values=c("solid", "dotdash"),
                        labels = c("Accuracy", "Dropout Share")) +
  scale_color_manual(name = "Value", values = c(my_darkgreen, my_darkblue),
                     labels = c("Accuracy", "Dropout Share")) +
  ylim(0,1) +
  xlab("Threshold") +
  ylab("") +
  theme(axis.title.y = element_text(margin = margin(t = 0, r = 15, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```




```
rm(sfew_fr, tresholds_sfew_fr, my_accuracy,
   my_dropout, my_dropout_share)
```

Appendix: Shares on all images

Compare Prototypical Data to Azure

```
load(file = paste0(data, "labdata_azure_emo_max.Rda"))

## calculate share of correctly identified from all images
table(labdata_azure$emo_azure_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       99      154       82      182      282      210      237
```

```
## 1) all images
sum(labdata_azure$emo_azure_max_char == labdata_azure$emotion, na.rm = T)/1246
```

```
## [1] 0.8097913
```

```
round(sum(labdata_azure$emo_azure_max_char ==
            labdata_azure$emotion, na.rm = T)/1246, digits = 2)
```

```
## [1] 0.81
```

```
## neutral
sum(labdata_azure[labdata_azure$emotion == "neutral",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "neutral",]$emotion, na.rm = T)/178
```

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

```
round(sum(labdata_azure[labdata_azure$emotion == "neutral",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "neutral",]$emotion,
          na.rm = T)/178, digits = 2)
```

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

```
## happy
sum(labdata_azure[labdata_azure$emotion == "happy",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "happy",]$emotion, na.rm = T)/178
```

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

```
round(sum(labdata_azure[labdata_azure$emotion == "happy",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "happy",]$emotion, na.rm = T)
      /178,
      digits = 2)
```

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

```
## angry
sum(labdata_azure[labdata_azure$emotion == "angry",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "angry",]$emotion, na.rm = T)/178
```

```
## [1] 0.505618
```

```
round(sum(labdata_azure[labdata_azure$emotion == "angry",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "angry",]$emotion, na.rm = T)/
        178,
      digits = 2)
```

```
## [1] 0.51
```

```
## disgust
sum(labdata_azure[labdata_azure$emotion == "disgust",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "disgust",]$emotion, na.rm = T)/178
```

```
## [1] 0.8483146
```

```r
round(sum(labdata_azure[labdata_azure$emotion == "disgust",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "disgust",]$emotion, na.rm = T)/
        178,
      digits = 2)
```

```
## [1] 0.85
```

```r
## surprise
sum(labdata_azure[labdata_azure$emotion == "surprise",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "surprise",]$emotion, na.rm = T)/178
```

```
## [1] 0.9775281
```

```r
round(sum(labdata_azure[labdata_azure$emotion == "surprise",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "surprise",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.98
```

```r
## sad
sum(labdata_azure[labdata_azure$emotion == "sad",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "sad",]$emotion, na.rm = T)/178
```

```
## [1] 0.8820225
```

```r
round(sum(labdata_azure[labdata_azure$emotion == "sad",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "sad",]$emotion, na.rm = T)/
        178,
      digits = 2)
```

```
## [1] 0.88
```

```r
## fear
sum(labdata_azure[labdata_azure$emotion == "fear",]$emo_azure_max_char ==
      labdata_azure[labdata_azure$emotion == "fear",]$emotion, na.rm = T)/178
```

```
## [1] 0.4550562
```

```r
round(sum(labdata_azure[labdata_azure$emotion == "fear",]$emo_azure_max_char ==
            labdata_azure[labdata_azure$emotion == "fear",]$emotion, na.rm = T)/
        178,
      digits = 2)
```

```
## [1] 0.46
```

```r
rm(labdata_azure, confmat)
```

```
## Warning in rm(labdata_azure, confmat): Objekt 'confmat' nicht gefunden
```

Compare Prototypical Data to Face++

```r
load(file = paste0(data, "labdata_facepp_emo_max.Rda"))
## calculate share of correctly identified from all images
table(labdata_facepp$emo_fpp_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##      103      206       75      184      242      193      243
```

```r
## 1) all images
sum(labdata_facepp$emo_fpp_max_char == labdata_facepp$emotion, na.rm = T)/1246
```

```
## [1] 0.7857143
```

```r
round(sum(labdata_facepp$emo_fpp_max_char == labdata_facepp$emotion, na.rm = T)/
        1246,
      digits = 2)
```

```
## [1] 0.79
```

```r
## neutral
sum(labdata_facepp[labdata_facepp$emotion == "neutral",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "neutral",]$emotion, na.rm = T)/178
```

```
## [1] 0.9438202
```

```r
round(sum(labdata_facepp[labdata_facepp$emotion == "neutral",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "neutral",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.94
```

```r
## happy
sum(labdata_facepp[labdata_facepp$emotion == "happy",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "happy",]$emotion, na.rm = T)/178
```

```
## [1] 0.994382
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "happy",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "happy",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.99
```

```
## angry
sum(labdata_facepp[labdata_facepp$emotion == "angry",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "angry",]$emotion, na.rm = T)/178
```

```
## [1] 0.488764
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "angry",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "angry",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.49
```

```
## disgust
sum(labdata_facepp[labdata_facepp$emotion == "disgust",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "disgust",]$emotion, na.rm = T)/178
```

```
## [1] 0.8932584
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "disgust",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "disgust",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.89
```

```
## surprise
sum(labdata_facepp[labdata_facepp$emotion == "surprise",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "surprise",]$emotion,
    na.rm = T)/178
```

```
## [1] 0.9719101
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "surprise",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "surprise",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.97
```

```
## sad
sum(labdata_facepp[labdata_facepp$emotion == "sad",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "sad",]$emotion, na.rm = T)/178
```

```
## [1] 0.8089888
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "sad",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "sad",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.81
```

```
## fear
sum(labdata_facepp[labdata_facepp$emotion == "fear",]$emo_fpp_max_char ==
      labdata_facepp[labdata_facepp$emotion == "fear",]$emotion, na.rm = T)/178
```

```
## [1] 0.3988764
```

```
round(sum(labdata_facepp[labdata_facepp$emotion == "fear",]$emo_fpp_max_char ==
            labdata_facepp[labdata_facepp$emotion == "fear",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.4
```

```
rm(labdata_facepp)
```

Compare Prototypical Data to FaceReader

```
load(file = paste0(data, "labdata_fr.Rda"))

## calculate share of correctly identified from all images
table(labdata_fr$emo_fr_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise  Unknown
##      173      174      161      179      191      179      187        2
```

```
## 1) all images
sum(as.character(labdata_fr$emo_fr_max_char) == labdata_fr$emotion, na.rm = T)/1246
```

```
## [1] 0.964687
```

```
round(sum(as.character(labdata_fr$emo_fr_max_char) == labdata_fr$emotion,
          na.rm = T)/1246, digits = 2)
```

```
## [1] 0.96
```

```
## neutral
sum(as.character(labdata_fr[labdata_fr$emotion == "neutral",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "neutral",]$emotion, na.rm = T)/178
```

```
## [1] 0.988764
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "neutral",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "neutral",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.99
```

```
## happy
sum(as.character(labdata_fr[labdata_fr$emotion == "happy",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "happy",]$emotion, na.rm = T)/178
```

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

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "happy",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "happy",]$emotion, na.rm = T)/178,
      digits = 2)
```

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

```
## angry
sum(as.character(labdata_fr[labdata_fr$emotion == "angry",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "angry",]$emotion, na.rm = T)/178
```

```
## [1] 0.9550562
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "angry",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "angry",]$emotion, na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.96
```

```
## disgust
sum(as.character(labdata_fr[labdata_fr$emotion == "disgust",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "disgust",]$emotion, na.rm = T)/178
```

```
## [1] 0.9662921
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "disgust",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "disgust",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.97
```

```
## surprise
sum(as.character(labdata_fr[labdata_fr$emotion == "surprise",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "surprise",]$emotion, na.rm = T)/178
```

```
## [1] 0.9831461
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "surprise",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "surprise",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.98
```

```
## sad
sum(as.character(labdata_fr[labdata_fr$emotion == "sad",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "sad",]$emotion, na.rm = T)/178
```

```
## [1] 0.9775281
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "sad",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "sad",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.98
```

```
## fear
sum(as.character(labdata_fr[labdata_fr$emotion == "fear",]$emo_fr_max_char) ==
      labdata_fr[labdata_fr$emotion == "fear",]$emotion, na.rm = T)/178
```

```
## [1] 0.8820225
```

```
round(sum(as.character(labdata_fr[labdata_fr$emotion == "fear",]$emo_fr_max_char) ==
            labdata_fr[labdata_fr$emotion == "fear",]$emotion,
          na.rm = T)/178,
      digits = 2)
```

```
## [1] 0.88
```

```
rm(labdata_fr)
```

Compare Naturalistic Data to Azure

```
load(file = paste0(data, "sfew_azure_emo_max.Rda"))

## calculate share of correctly identified from all images
table(sfew_azure$emo_azure_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       84       14        9      235      487      113      163
```

```
## 1) all images
round(sum(sfew_azure$emo_azure_max_char == sfew_azure$emotion, na.rm = T)/1387,
      digits = 2)
```

```
## [1] 0.45
```

```
## neutral
round(sum(sfew_azure[sfew_azure$emotion == "neutral",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "neutral",]$emotion, na.rm = T)/236,
      digits = 2)
```

```
## [1] 0.78
```

```
## happy
sum(sfew_azure[sfew_azure$emotion == "happy",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "happy",]$emotion, na.rm = T)/270
```

```
## [1] 0.762963
```

```
round(sum(sfew_azure[sfew_azure$emotion == "happy",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "happy",]$emotion, na.rm = T)/270,
      digits = 2)
```

```
## [1] 0.76
```

```
## angry
sum(sfew_azure[sfew_azure$emotion == "angry",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "angry",]$emotion, na.rm = T)/254
```

```
## [1] 0.2874016
```

```
round(sum(sfew_azure[sfew_azure$emotion == "angry",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "angry",]$emotion, na.rm = T)/254,
      digits = 2)
```

```
## [1] 0.29
```

```
## disgust
sum(sfew_azure[sfew_azure$emotion == "disgust",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "disgust",]$emotion, na.rm = T)/88
```

```
## [1] 0.07954545
```

```
round(sum(sfew_azure[sfew_azure$emotion == "disgust",]$emo_azure_max_char ==

            sfew_azure[sfew_azure$emotion == "disgust",]$emotion, na.rm = T)/88,
      digits = 2)
```

```
## [1] 0.08
```

```
## surprise
sum(sfew_azure[sfew_azure$emotion == "surprise",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "surprise",]$emotion, na.rm = T)/151
```

```
## [1] 0.4635762
```

```
round(sum(sfew_azure[sfew_azure$emotion == "surprise",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "surprise",]$emotion,
          na.rm = T)/151,
      digits = 2)
```

```
## [1] 0.46
```

```
## sad
sum(sfew_azure[sfew_azure$emotion == "sad",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "sad",]$emotion, na.rm = T)/245
```

```
## [1] 0.355102
```

```
round(sum(sfew_azure[sfew_azure$emotion == "sad",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "sad",]$emotion, na.rm = T)/245,
      digits = 2)
```

```
## [1] 0.36
```

```
## fear
sum(sfew_azure[sfew_azure$emotion == "fear",]$emo_azure_max_char ==
      sfew_azure[sfew_azure$emotion == "fear",]$emotion, na.rm = T)/143
```

```
## [1] 0.02097902
```

```
round(sum(sfew_azure[sfew_azure$emotion == "fear",]$emo_azure_max_char ==
            sfew_azure[sfew_azure$emotion == "fear",]$emotion, na.rm = T)/143,
      digits = 2)
```

```
## [1] 0.02
```

```
rm(sfew_azure)
```

Compare Naturalistic Data to Face++

```r
load(file = paste0(data, "sfew_facepp_emo_max.Rda"))

## calculate share of correctly identified from all images
table(sfew_facepp$emo_fpp_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##      105       83      146      168      276      112      490
```

```r
## 1) all images
sum(sfew_facepp$emo_fpp_max_char == sfew_facepp$emotion, na.rm = T)/1387
```

```
## [1] 0.3208363
```

```r
round(sum(sfew_facepp$emo_fpp_max_char == sfew_facepp$emotion, na.rm = T)/1387,
      digits = 2)
```

```
## [1] 0.32
```

```r
## neutral
sum(sfew_facepp[sfew_facepp$emotion == "neutral",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "neutral",]$emotion, na.rm = T)/236
```

```
## [1] 0.4025424
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "neutral",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "neutral",]$emotion,
          na.rm = T)/236,
      digits = 2)
```

```
## [1] 0.4
```

```r
## happy
sum(sfew_facepp[sfew_facepp$emotion == "happy",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "happy",]$emotion, na.rm = T)/270
```

```
## [1] 0.4740741
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "happy",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "happy",]$emotion,
          na.rm = T)/270,
      digits = 2)
```

```
## [1] 0.47
```

```r
## angry
sum(sfew_facepp[sfew_facepp$emotion == "angry",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "angry",]$emotion, na.rm = T)/254
```

```
## [1] 0.1496063
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "angry",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "angry",]$emotion,
          na.rm = T)/254,
      digits = 2)
```

```
## [1] 0.15
```

```r
## disgust
sum(sfew_facepp[sfew_facepp$emotion == "disgust",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "disgust",]$emotion, na.rm = T)/88
```

```
## [1] 0.1590909
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "disgust",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "disgust",]$emotion,
          na.rm = T)/88,
      digits = 2)
```

```
## [1] 0.16
```

```r
## surprise
sum(sfew_facepp[sfew_facepp$emotion == "surprise",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "surprise",]$emotion, na.rm = T)/151
```

```
## [1] 0.6556291
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "surprise",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "surprise",]$emotion,
          na.rm = T)/151,
      digits = 2)
```

```
## [1] 0.66
```

```r
## sad
sum(sfew_facepp[sfew_facepp$emotion == "sad",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "sad",]$emotion, na.rm = T)/245
```

```
## [1] 0.1836735
```

```r
round(sum(sfew_facepp[sfew_facepp$emotion == "sad",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "sad",]$emotion,
          na.rm = T)/245,
      digits = 2)
```

```
## [1] 0.18
```

```
## fear
sum(sfew_facepp[sfew_facepp$emotion == "fear",]$emo_fpp_max_char ==
      sfew_facepp[sfew_facepp$emotion == "fear",]$emotion, na.rm = T)/143
```

```
## [1] 0.1818182
```

```
round(sum(sfew_facepp[sfew_facepp$emotion == "fear",]$emo_fpp_max_char ==
            sfew_facepp[sfew_facepp$emotion == "fear",]$emotion, na.rm = T)/143,
      digits = 2)
```

```
## [1] 0.18
```

```
rm(sfew_facepp)
```

Compare Naturalistic Data to FaceReader

```
load(file = paste0(data, "sfew_fr.Rda"))

# keep those with qual >= .7 for the analysis
sfew_fr <- filter(sfew_fr, Quality >=0.7)

## calculate share of correctly identified from all images
table(sfew_fr$emo_fr_max_char)
```

```
##
##    angry  disgust     fear    happy  neutral      sad surprise
##       19       23        7       37      201       31       40
```

```
## 1) all images
sum(sfew_fr$emo_fr_max_char == sfew_fr$emotion, na.rm = T)/1387
```

```
## [1] 0.08002884
```

```
round(sum(sfew_fr$emo_fr_max_char == sfew_fr$emotion, na.rm = T)/1387,
      digits = 2)
```

```
## [1] 0.08
```

```
## neutral
sum(sfew_fr[sfew_fr$emotion == "neutral",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "neutral",]$emotion, na.rm = T)/236
```

```
## [1] 0.1779661
```

```
round(sum(sfew_fr[sfew_fr$emotion == "neutral",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "neutral",]$emotion, na.rm = T)/236,
      digits = 2)
```

```
## [1] 0.18
```

```r
## happy
sum(sfew_fr[sfew_fr$emotion == "happy",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "happy",]$emotion, na.rm = T)/270
```

```
## [1] 0.1259259
```

```r
round(sum(sfew_fr[sfew_fr$emotion == "happy",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "happy",]$emotion, na.rm = T)/270,
      digits = 2)
```

```
## [1] 0.13
```

```r
## angry
sum(sfew_fr[sfew_fr$emotion == "angry",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "angry",]$emotion, na.rm = T)/254
```

```
## [1] 0.03149606
```

```r
round(sum(sfew_fr[sfew_fr$emotion == "angry",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "angry",]$emotion, na.rm = T)/254,
      digits = 2)
```

```
## [1] 0.03
```

```r
## disgust
sum(sfew_fr[sfew_fr$emotion == "disgust",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "disgust",]$emotion, na.rm = T)/88
```

```
## [1] 0.04545455
```

```r
round(sum(sfew_fr[sfew_fr$emotion == "disgust",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "disgust",]$emotion, na.rm = T)/88,
      digits = 2)
```

```
## [1] 0.05
```

```r
## surprise
sum(sfew_fr[sfew_fr$emotion == "surprise",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "surprise",]$emotion, na.rm = T)/151
```

```
## [1] 0.08609272
```

```r
round(sum(sfew_fr[sfew_fr$emotion == "surprise",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "surprise",]$emotion, na.rm = T)/151,
      digits = 2)
```

```
## [1] 0.09
```

```
## sad
sum(sfew_fr[sfew_fr$emotion == "sad",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "sad",]$emotion, na.rm = T)/245
```

```
## [1] 0.04081633
```

```
round(sum(sfew_fr[sfew_fr$emotion == "sad",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "sad",]$emotion, na.rm = T)/245,
      digits = 2)
```

```
## [1] 0.04
```

```
## fear
sum(sfew_fr[sfew_fr$emotion == "fear",]$emo_fr_max_char ==
      sfew_fr[sfew_fr$emotion == "fear",]$emotion, na.rm = T)/143
```

```
## [1] 0
```

```
round(sum(sfew_fr[sfew_fr$emotion == "fear",]$emo_fr_max_char ==
            sfew_fr[sfew_fr$emotion == "fear",]$emotion, na.rm = T)/143,
      digits = 2)
```

```
## [1] 0
```

```
rm(sfew_fr)
```