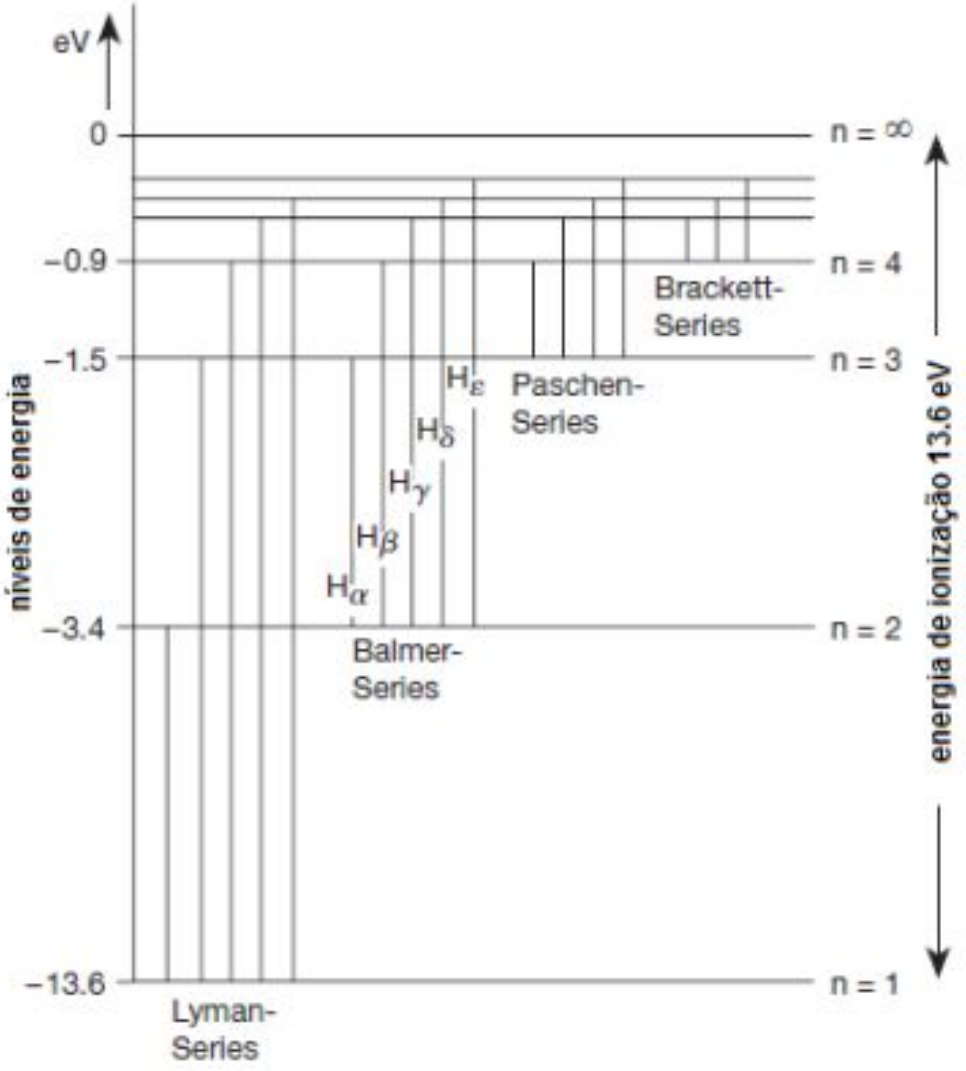
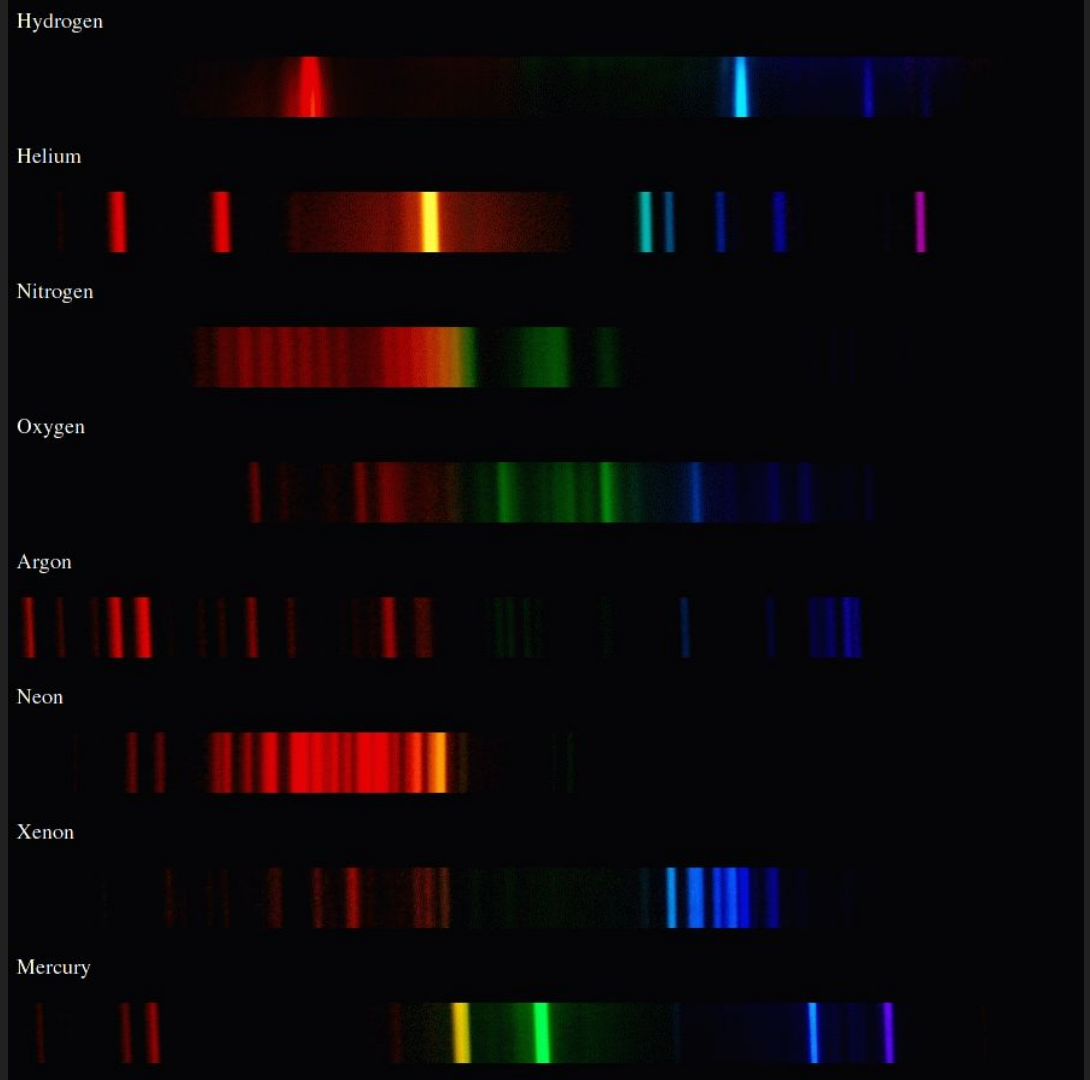
06 - We don’t really miss you

Cristiane Fontana

https://www.if.ufrj.br/~maximo/Lab1-Roteiros/Serie-Balmer.pdf



http://spiff.rit.edu/classes/phys200/lectures/spectra/spectra.html



Leis de Kirchhoff

1) Um corpo opaco quente, sólido, líquido ou gasoso, emite um espectro contínuo.

2) Um gás transparente produz um espectro de linhas brilhantes (de emissão). O número e a posição dessas linhas depende dos elementos químicos presentes no gás.

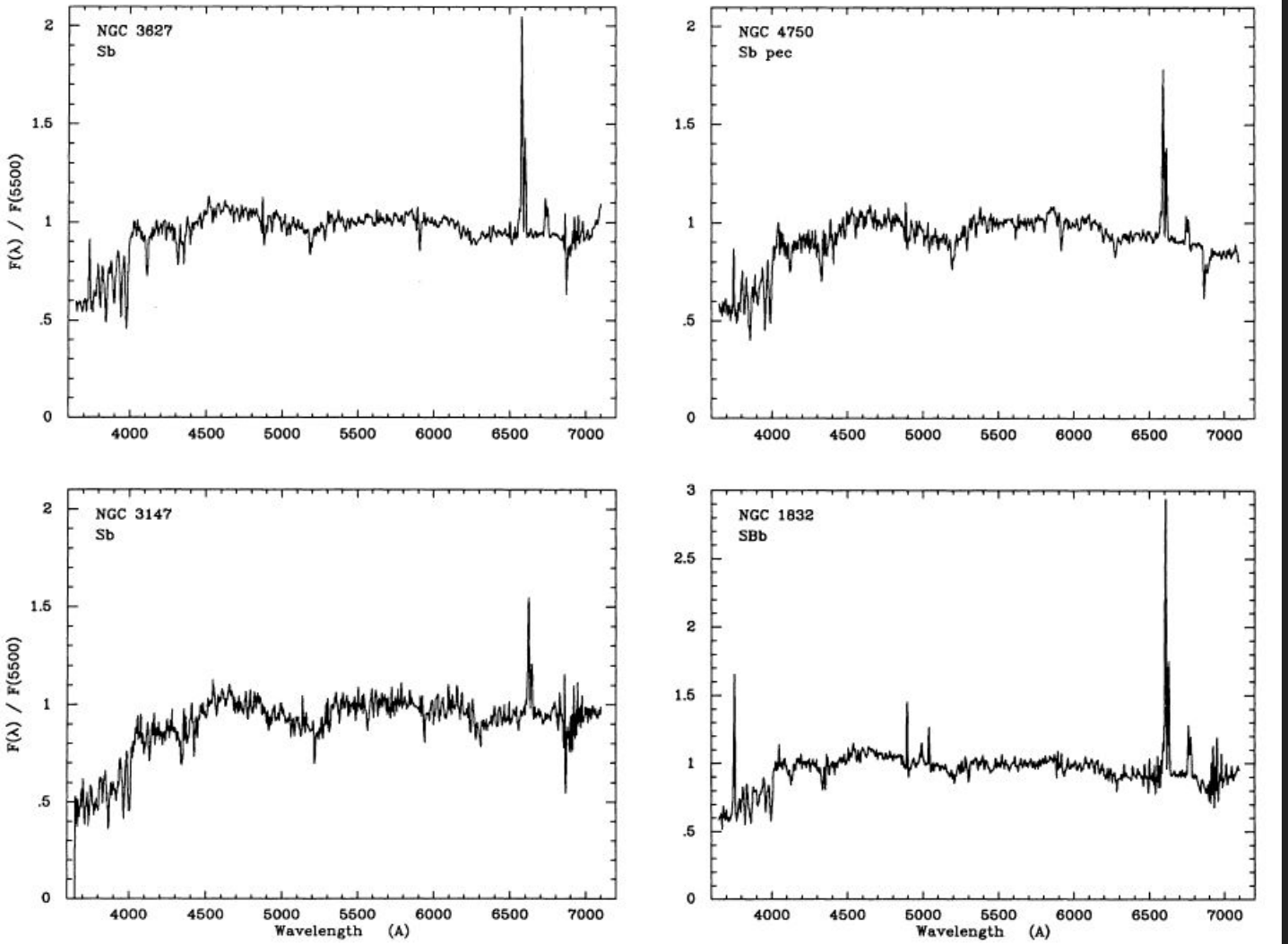
3) Se um espectro contínuo passar por um gás à temperatura mais baixa, o gás frio causa a presença de linhas escuras (absorção). O número e a posição dessas linhas depende dos elementos químicos presentes no gás.

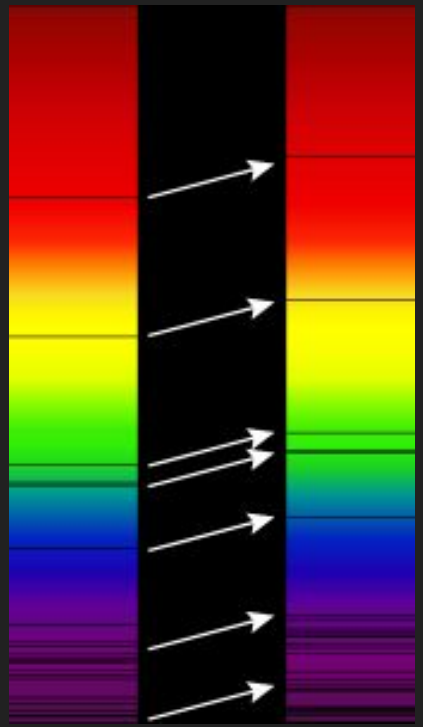
Continuous spectrum

Emission lines (discrete spectrum)

Absorption lines (discrete spectrum)

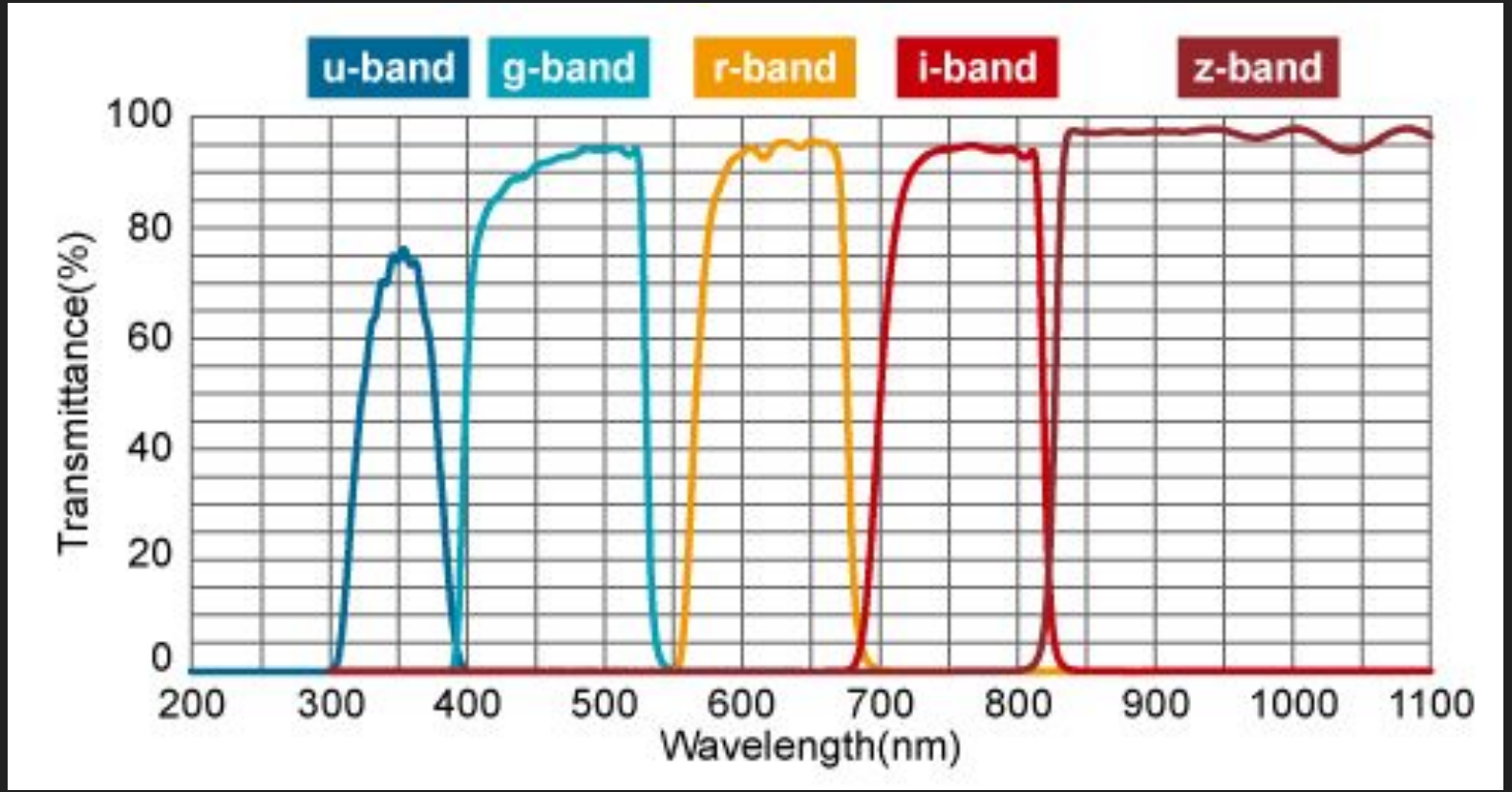
http://astronomy.nmsu.edu/nicole/teaching/ASTR505/lectures/lecture26/slide01.html





Based on wavelength
$z = \frac{\lambda_{\text{obsv}} - \lambda_{\text{emit}}}{\lambda_{\text{emit}}}$
$1 + z = \frac{\lambda_{\text{obsv}}}{\lambda_{\text{emit}}}$

https://stoneedgeobservatory.com/high-precision-photometry-using-the-sdss-photometric-system/



R, G, Z, W1, W2, W3, W4

Missing data techniques

- Excluir tudo que estiver incompleto (linha)
- Excluir tudo que estiver incompleto (coluna)

  (R, G, Z, W1, W2)

- Rodar só com dados completos
- Rodar sem uma banda
- Preencher dados faltantes com a média das bandas vizinhas
- Preencher dados faltantes com a média dos valores da banda
- Preencher usando algum tipo de modelo de imputação, nesse caso um modelo de regressão linear

Limpeza dos dados

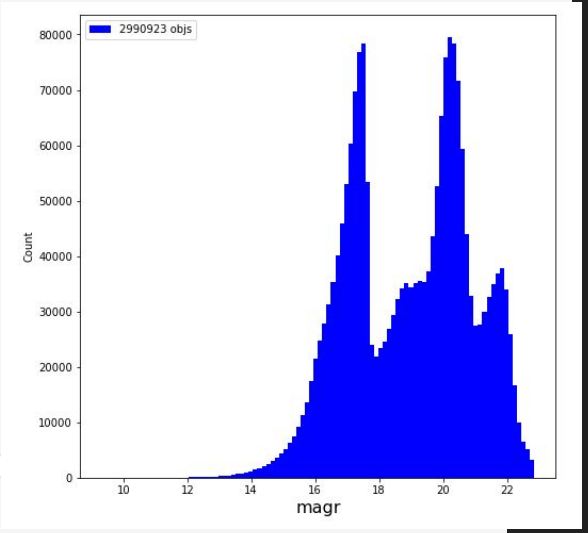
```
#chamando os possíveis pacotes necessários
from astropy.io import fits
from astropy.table import Table
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import seaborn
import os
```

```
#abrindo os dados
hdul = fits.open("legacy_spec_allsky_extcorr_mag_colorstar_clean_dr9_phzerr09.fits", ignore_missing_end=True)
data = hdul[1].data # table extensions can't be the first extension, so there's a dummy image extension at 1
cat_table = Table(data)
cols=cat_table.columns
print(cat_table)
print(cols)
```

```
#transformamos em dataframe (pandas) por que é muito fácil dropar dados por lá
df = cat_table.to_pandas()
print(df)
```

```
#temos que jogar fora primeiro o que não vamos usar, W3 e W4 são bandas com medidas muito incompletas

df.drop('z',
  axis='columns', inplace=True)

df.drop('zErr',
  axis='columns', inplace=True)

df.drop('MAG_W3',
  axis='columns', inplace=True)

df.drop('MAGErr_W3',
  axis='columns', inplace=True)

df.drop('MAG_W4',
  axis='columns', inplace=True)

df.drop('MAGErr_W4',
  axis='columns', inplace=True)

df.drop('training',
  axis='columns', inplace=True)

df.drop('zph_min',
  axis='columns', inplace=True)

df.drop('zph_max',
  axis='columns', inplace=True)
```

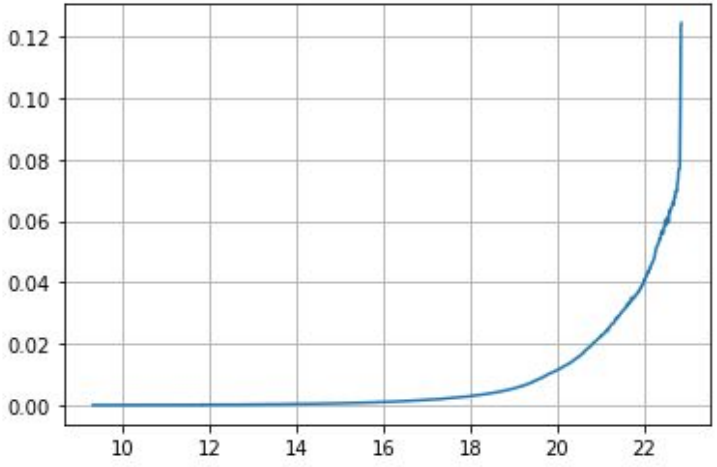
```
# algumas coisas eram string, então transformo string para número

cls = ['MAG_R','MAG_G','MAG_Z','MAG_W1','MAG_W2','MAGErr_R','MAGErr_G','MAGErr_Z','MAGErr_W1','MAGErr_W2','zspec']
for i in cls:
    df[i] = pd.to_numeric(df[i],errors='coerce')

#algumas coisas nan ele lê como zero então coloquei pra tirar os zeros
df = df[(df[cls] != 0.0).all(axis=1)]

#alguns valores são inf
pd.set_option('mode.use_inf_as_na', True)

#dropar todas as linhas com algum nan
df = df.dropna()

#salvar um novo fits só pra mostrar os histogramas
cats=Table.from_pandas(df)
cats.write('new.fits')
```



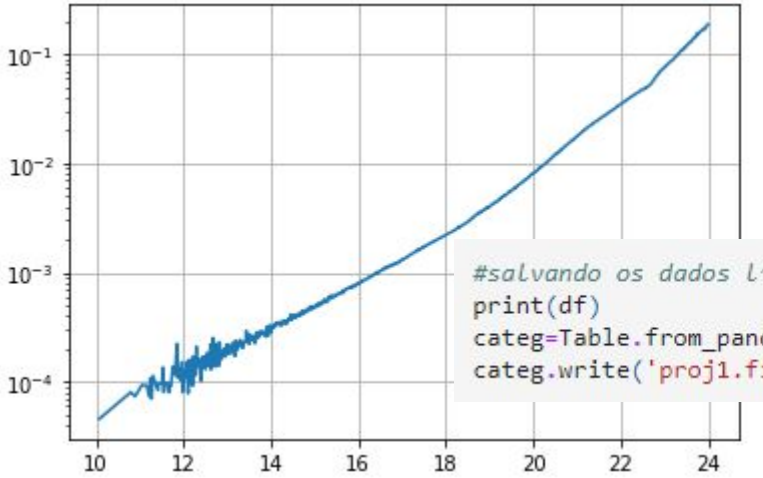

```
#histogramas
hdul2 = fits.open("new.fits", ignore_missing_end=True)
data2 = hdul2[1].data # table extensions can't be the first extension, so there's a dummy image extension at 1

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot()
ax.hist(np.array(data2.field('MAG_R')), bins=100, color='blue', label=f'{len(data)} objs')
ax.set_xlabel('magr', fontsize=16)
ax.set_ylabel('Count')
#ax.set_xlim(-0.01,0.01)
plt.legend()
plt.show()
```

```
binned = pd.cut(df['MAG_R'],1000)

erro_medio=list()
ponto_medio=list()

for bin in sorted(binned.unique()):
  #print(bin,'\n',df[binned==bin].MAGErr_R.mean())
  ponto_medio.append(df[binned==bin].MAG_R.mean())
  erro_medio.append(df[binned==bin].MAGErr_R.mean())

DF=pd.DataFrame(erro_medio,ponto_medio)
plt.grid(True)
plt.plot(DF)

#NÃO PRECISA CORTAR NADA PQ TÁ TUDO DENTRO DE 0.2 (5 SIGMA)
```

```
[<matplotlib.lines.Line2D at 0x2477100b5b0>]
```




```
binned = pd.cut(df['MAG_G'],1000)

erro_medio=list()
ponto_medio=list()

for bin in sorted(binned.unique()):
  #print(bin,'\n',df[binned==bin].MAGErr_R.mean())
  ponto_medio.append(df[binned==bin].MAG_G.mean())
  erro_medio.append(df[binned==bin].MAGErr_G.mean())

DF=pd.DataFrame(erro_medio,ponto_medio)
plt.grid(True)
plt.plot(DF)
plt.yscale('log')

df = df[df['MAG_G'] < 24]
```




```
#salvando os dados limpos
print(df)
categ=Table.from_pandas(df)
categ.write('proj1.fits')
```

1- Rodar com todas as bandas

```
from astropy.io import fits
import os
import numpy as np
from astropy.table import Table
import matplotlib.pyplot as plt
import pandas as pd

def open_fits_catalog(fits_file):
    hdu_list=fits.open(fits_file, ignore_missing_end=True)
    #print hdu_list
    hdu = hdu_list[1]    # table extensions can't be the first extension, so there's a dummy image extension at 0
    #print hdu.header
    cat_table = Table(hdu.data)
    cols=hdu.columns
    return cat_table, cols
```

```
data_file = os.path.join('proj1.fits')
data, _ = open_fits_catalog(data_file)
```

```
BANDS = ["R","Z", "G", "W1", "W2"]
mags = ['MAG_'+band for band in BANDS]
```

```python
#preprocessing

from sklearn.preprocessing import MinMaxScaler
np.random.seed(42)

# queremos 10% de teste
test_percentual = 1/10
#vou selecionar os 10%
test_cut = np.random.uniform(0, 1, len(data)) < test_percentual
#salvar os 10% na variável test_data
test_data = data[test_cut]

#selecionar os outros 90%
train_val_data = data[~test_cut]

#calcular os 5 grupos cada um com 18%, A, B, C, D e E
#para A
A_percentual = 1/5
A_cut = np.random.uniform(0, 1, len(train_val_data)) < A_percentual
A_data = train_val_data[A_cut]

#72%
train_val_data2 = train_val_data[~A_cut]

#para B
B_percentual = 1/4
B_cut = np.random.uniform(0, 1, len(train_val_data2)) < B_percentual
B_data = train_val_data2[B_cut]

#54%
train_val_data3 = train_val_data2[~B_cut]
```

```python
#para C
C_percentual = 1/3
C_cut = np.random.uniform(0, 1, len(train_val_data3)) < C_percentual
C_data = train_val_data3[C_cut]

#36%
train_val_data4 = train_val_data3[~C_cut]

#para D
D_percentual = 1/2
D_cut = np.random.uniform(0, 1, len(train_val_data4)) < D_percentual
D_data = train_val_data4[D_cut]

#18%
#para E
E_data = train_val_data4[~D_cut]


df1 = A_data.to_pandas()
#print(df1)

df2 = B_data.to_pandas()
#print(df2)

df3 = C_data.to_pandas()
#print(df3)

df4 = D_data.to_pandas()
#print(df4)

df5 = E_data.to_pandas()
#print(df5)
```

```
#Agora é necessário juntar as coisas em 5 novos grupos:

train_datadf1 = df2.append([df3, df4, df5], ignore_index=True)
#print(train_datadf1)

train_datadf2 = df1.append([df3, df4, df5], ignore_index=True)
#print(train_datadf2)

train_datadf3 = df1.append([df2, df4, df5], ignore_index=True)
#print(train_datadf3)

train_datadf4 = df1.append([df2, df3, df5], ignore_index=True)
#print(train_datadf4)

train_datadf5 = df1.append([df2, df3, df4], ignore_index=True)
#print(train_datadf5)
```

```
Test Percent = 9.99%
Train Percent1 = 71.98%
Val Percent1 = 18.03%
Train Percent2 = 72.01%
Val Percent2 = 18.00%
Train Percent3 = 72.02%
Val Percent3 = 17.99%
Train Percent4 = 72.03%
Val Percent4 = 17.98%
Train Percent5 = 71.99%
Val Percent5 = 18.01%
```

```
#grupo 1: A é val_data1, B, C, D e E são train_data1
val_data1 = A_data
train_data1 = Table.from_pandas(train_datadf1)

#grupo 2: B é val_data2, A, C, D e E são train_data2
val_data2 = B_data
train_data2 = Table.from_pandas(train_datadf2)

#grupo 3: C é val_data3, A, B, D e E são train_data3
val_data3 = C_data
train_data3 = Table.from_pandas(train_datadf3)

#grupo 4: D é val_data4, A, B, C e E são train_data4
val_data4 = D_data
train_data4 = Table.from_pandas(train_datadf4)

#grupo 5: D é val_data5, A, B, C e D são train_data5
val_data5 = E_data
train_data5 = Table.from_pandas(train_datadf5)
```

```
#embaralhando os dados

#embaralhando o teste
test_shuffler = np.random.choice(len(test_data), len(test_data), replace=False)
test_data = test_data[test_shuffler]

#embaralhando grupo 1

train_shuffler1 = np.random.choice(len(train_data1), len(train_data1), replace=False)
train_data1 = train_data1[train_shuffler1]

val_shuffler1 = np.random.choice(len(val_data1), len(val_data1), replace=False)
val_data1 = val_data1[val_shuffler1]
```

```
#especificando os dados que usaremos para testar (mags, x_train) e os dados resposta (zspec, y_train)

x_test = np.array([test_data.field(mag) for mag in mags]).T
y_test = np.array(test_data.field('zspec'))

#especificando os dados que usaremos para treinar (mags, x_train) e os dados resposta (zspec, y_train)

x_train1 = np.array([train_data1.field(mag) for mag in mags]).T
y_train1 = np.array(train_data1.field('zspec'))

x_train2 = np.array([train_data2.field(mag) for mag in mags]).T
y_train2 = np.array(train_data2.field('zspec'))

x_train3 = np.array([train_data3.field(mag) for mag in mags]).T
y_train3 = np.array(train_data3.field('zspec'))

x_train4 = np.array([train_data4.field(mag) for mag in mags]).T
y_train4 = np.array(train_data4.field('zspec'))

x_train5 = np.array([train_data5.field(mag) for mag in mags]).T
y_train5 = np.array(train_data5.field('zspec'))
```

```
#especificando os dados que usaremos para validação (mags, x_train) e os dados resposta (zspec, y_train)

x_val1 = np.array([val_data1.field(mag) for mag in mags]).T
y_val1 = np.array(val_data1.field('zspec'))

x_val2 = np.array([val_data2.field(mag) for mag in mags]).T
y_val2 = np.array(val_data2.field('zspec'))

x_val3 = np.array([val_data3.field(mag) for mag in mags]).T
y_val3 = np.array(val_data3.field('zspec'))

x_val4 = np.array([val_data4.field(mag) for mag in mags]).T
y_val4 = np.array(val_data4.field('zspec'))

x_val5 = np.array([val_data5.field(mag) for mag in mags]).T
y_val5 = np.array(val_data5.field('zspec'))


scaler = MinMaxScaler()
x_train1 = scaler.fit_transform(x_train1)
x_val1 = scaler.transform(x_val1)
x_train2 = scaler.transform(x_train2)
x_val2 = scaler.transform(x_val2)
x_train3 = scaler.transform(x_train3)
x_val3 = scaler.transform(x_val3)
x_train4 = scaler.transform(x_train4)
x_val4 = scaler.transform(x_val4)
x_train5 = scaler.transform(x_train5)
x_val5 = scaler.transform(x_val5)
x_test = scaler.transform(x_test)
```

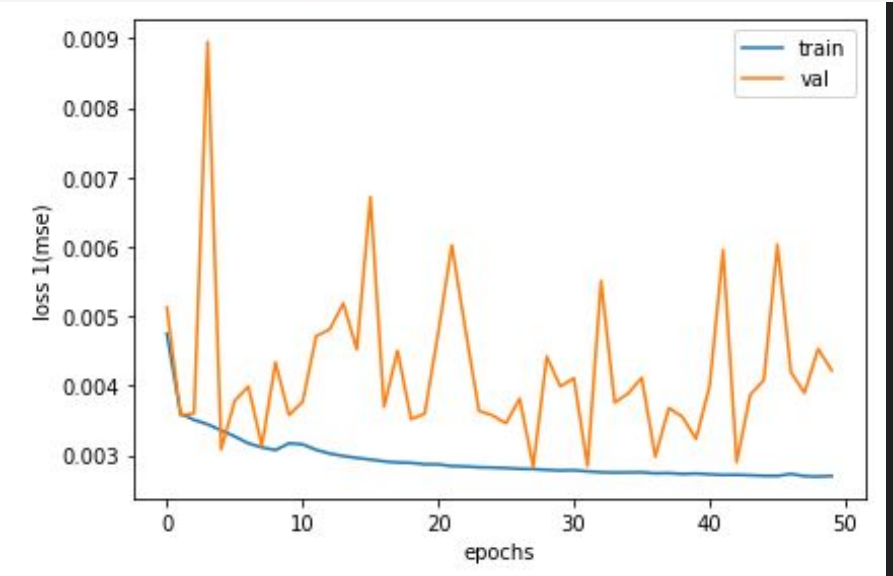
```python
import tensorflow.keras as keras
# os.environ['CUDA_DEVICE_ORDER'] = 'PCI_BUS_ID'
# os.environ['CUDA_VISIBLE_DEVICES'] = "1"


def build_model(input_shape, nb_classes=1):
    input_layer = keras.layers.Input(input_shape)

    layer_1 = keras.layers.Dense(512, activation='relu')(input_layer)
    layer_1= keras.layers.Dropout(rate=0.2)(layer_1)

    layer_2 = keras.layers.Dense(256, activation='relu')(layer_1)
    layer_2 = keras.layers.Dropout(rate=0.2)(layer_2)

    layer_3 = keras.layers.Dense(128, activation='relu')(layer_2)
    layer_3 = keras.layers.Dropout(rate=0.2)(layer_3)

    output_layer = keras.layers.Dense(units=nb_classes,activation='linear')(layer_3)


    model = keras.models.Model(inputs=input_layer, outputs=output_layer)
    return model
```

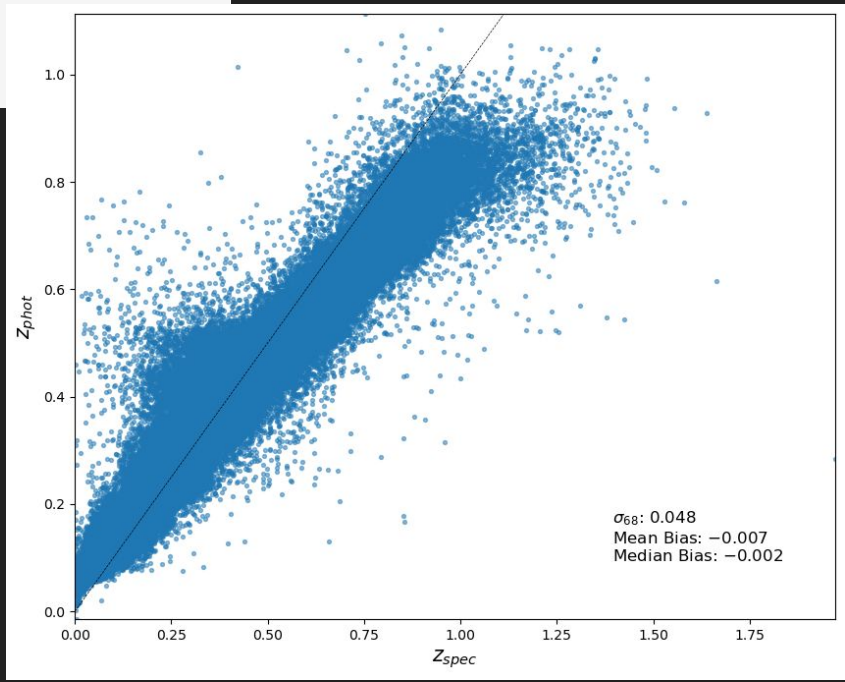
Rodar com todas as bandas

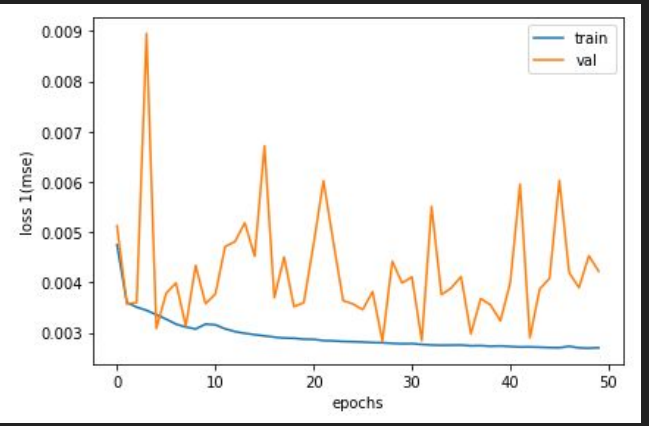
Deu errado ou certo demais: você escolhe

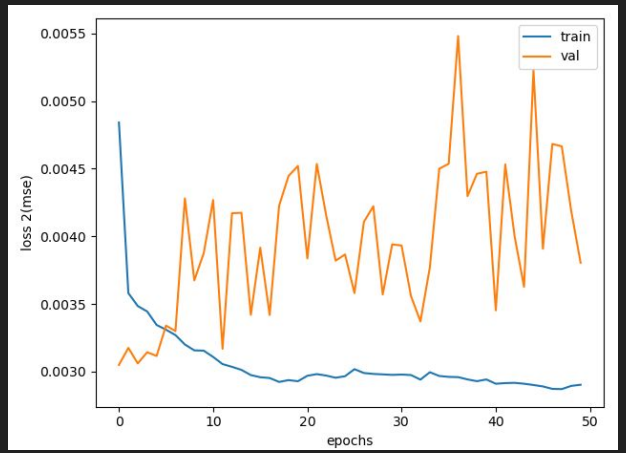
```
#para grupo 3

regressor3 = build_model(x_train3.shape[1:])
regressor3.compile(loss='mse', optimizer=keras.optimizers.Nadam(), metrics=['accuracy'])


batch_size = 128
epochs = 50
#epochs = 5

history3 = regressor3.fit(x_train3, y_train3, batch_size=batch_size, epochs=epochs,
                    verbose=1, validation_data=(x_val3,y_val3))

preds3 = regressor3.predict(x_test)
regressor3.save('regressorh3')
```

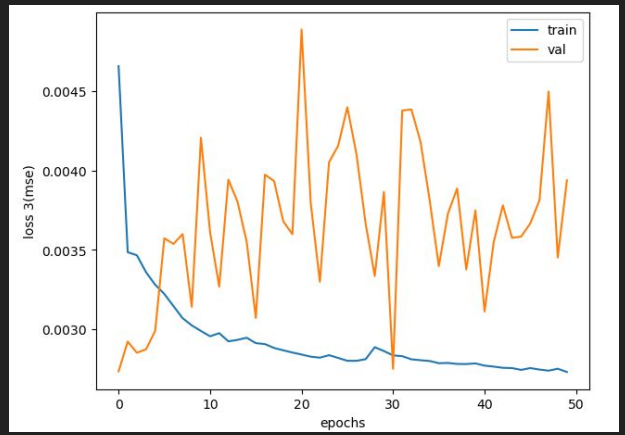
6 a 28 horas

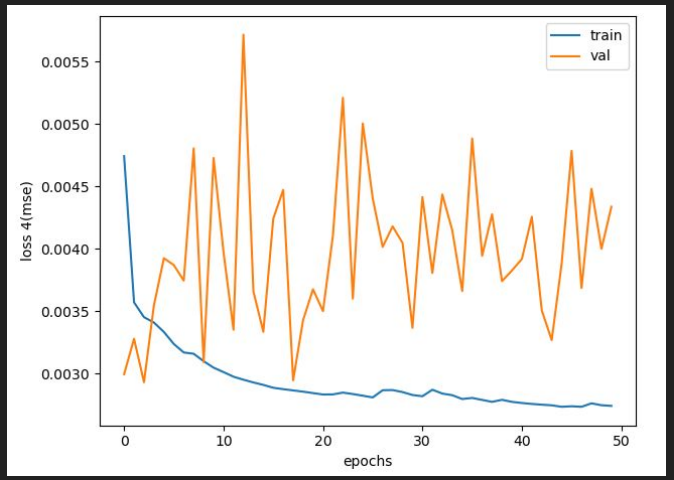


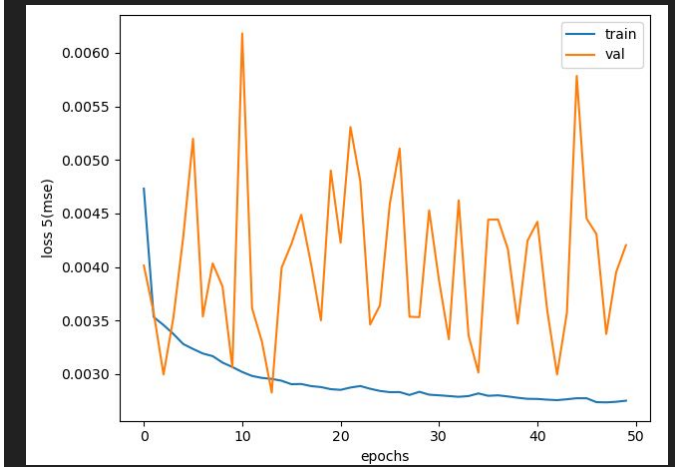



train
val
loss 1(mse)
epochs


train
val
loss 2(mse)
epochs

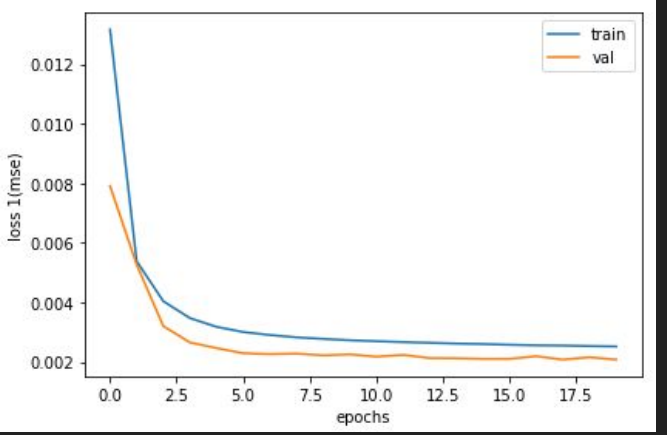

train
val
loss 3(mse)
epochs

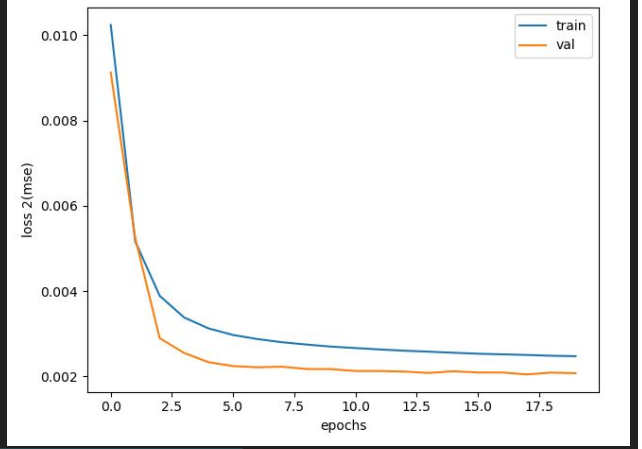

train
val
loss 4(mse)
epochs

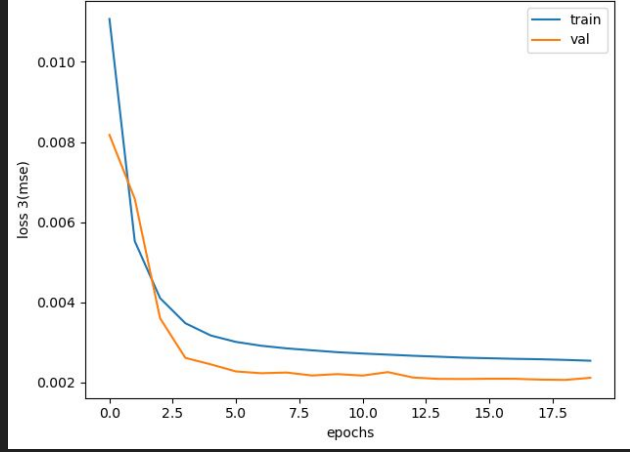

train
val
loss 5(mse)
epochs

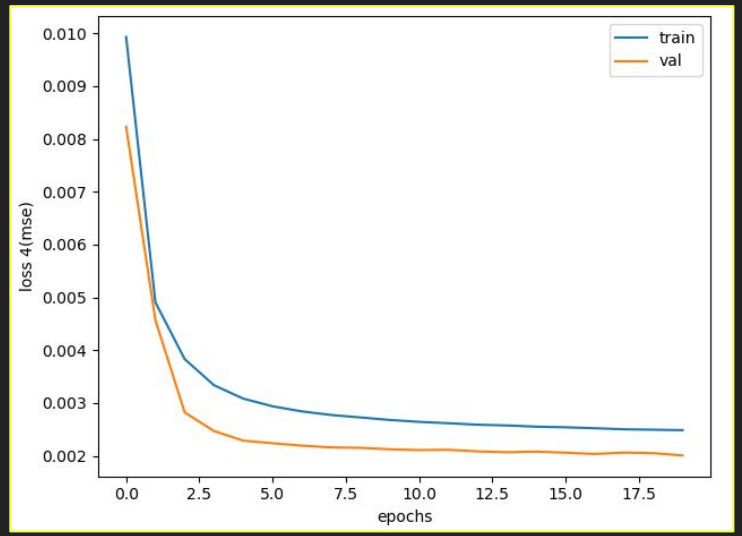
```
#definindo o learning rate de todos os treinamentos
opt = keras.optimizers.Adam(learning_rate=0.00001)
```

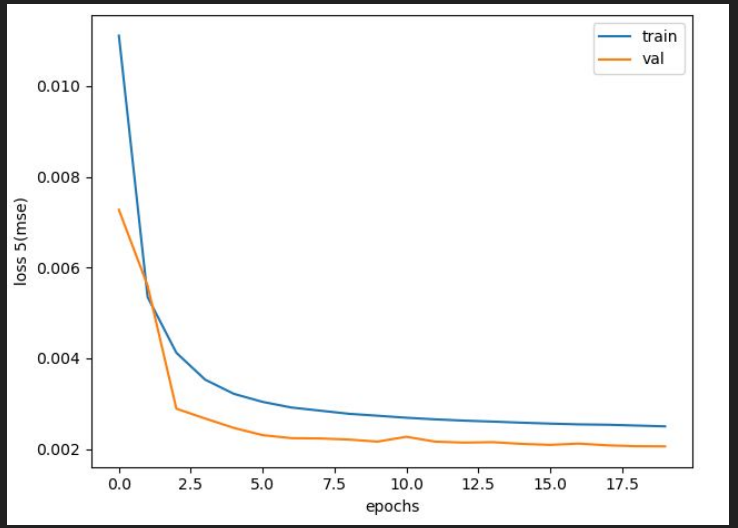
```
#para grupo 1

#a partir de agora vamos treinar com cada grupo de treino
regressor1 = build_model(x_train1.shape[1:])
regressor1.compile(loss='mse', optimizer=opt, metrics=['accuracy'])

#número de exemplos de treinamento usados em uma interação
batch_size = 128
epochs = 20
#epochs = 50

history1 = regressor1.fit(x_train1, y_train1, batch_size=batch_size, epochs=epochs,
                    verbose=1, validation_data=(x_val1,y_val1))

#resultado do teste, e salvando o modelo
preds1 = regressor1.predict(x_test)
regressor1.save('regressor21')
```

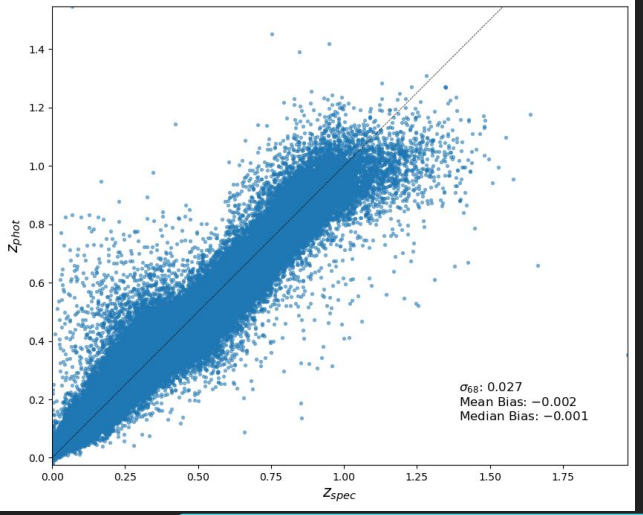

train
val
loss 1(mse)
epochs

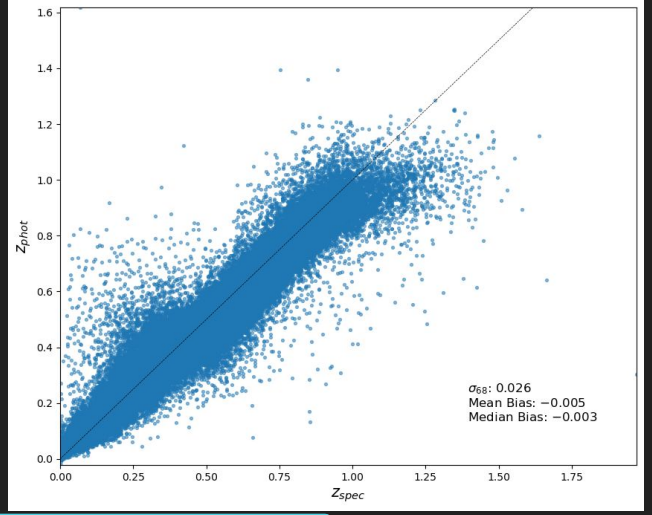

train
val
loss 2(mse)
epochs

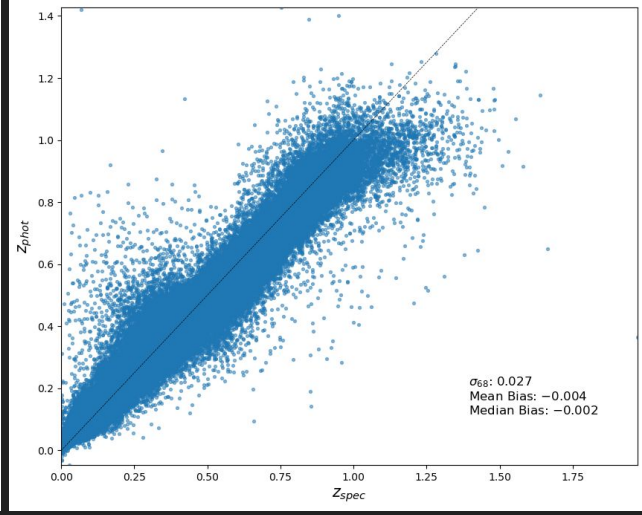

train
val
loss 3(mse)
epochs

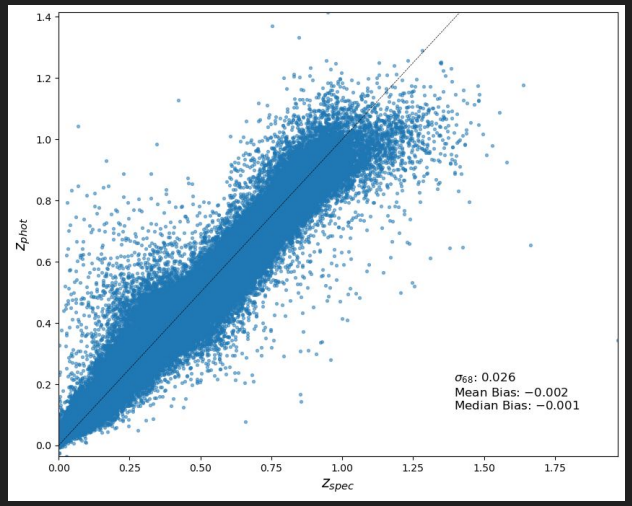

train
val
loss 4(mse)
epochs

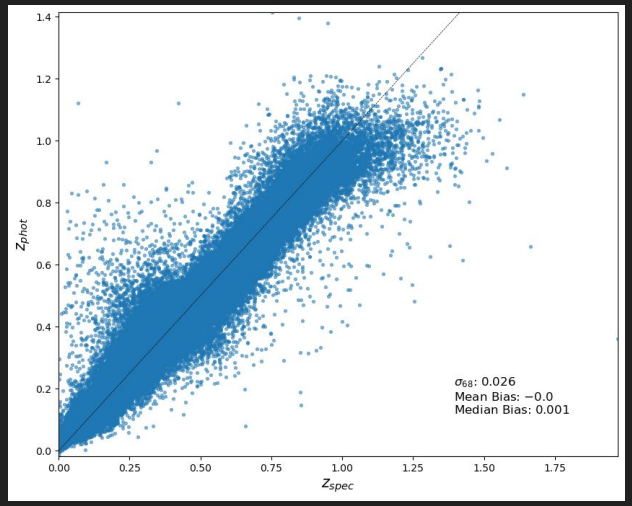

train
val
loss 5(mse)
epochs


z_{phot}
z_{spec}
σ_{68}: 0.027
Mean Bias: −0.002
Median Bias: −0.001

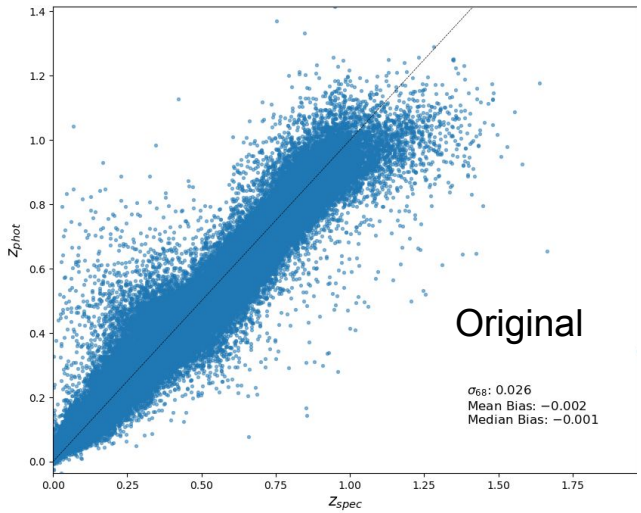

z_{phot}
z_{spec}
σ_{68}: 0.026
Mean Bias: −0.005
Median Bias: −0.003

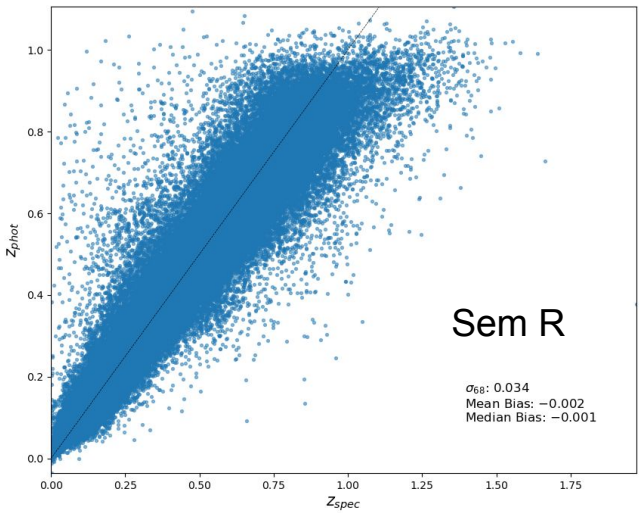

z_{phot}
z_{spec}
σ_{68}: 0.027
Mean Bias: −0.004
Median Bias: −0.002

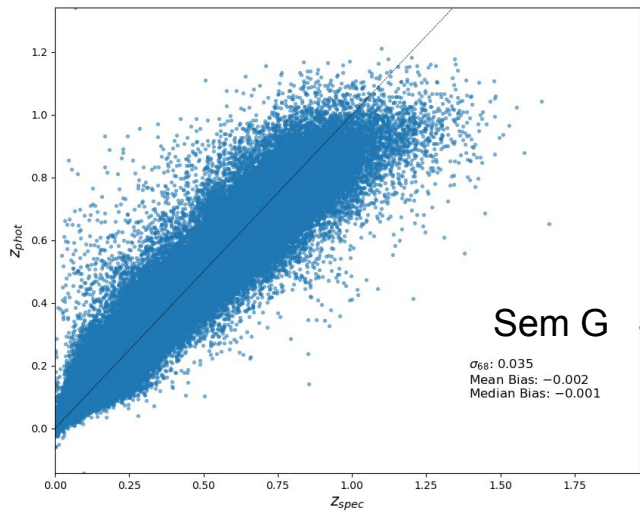

z_{phot}
z_{spec}
σ_{68}: 0.026
Mean Bias: −0.002
Median Bias: −0.001

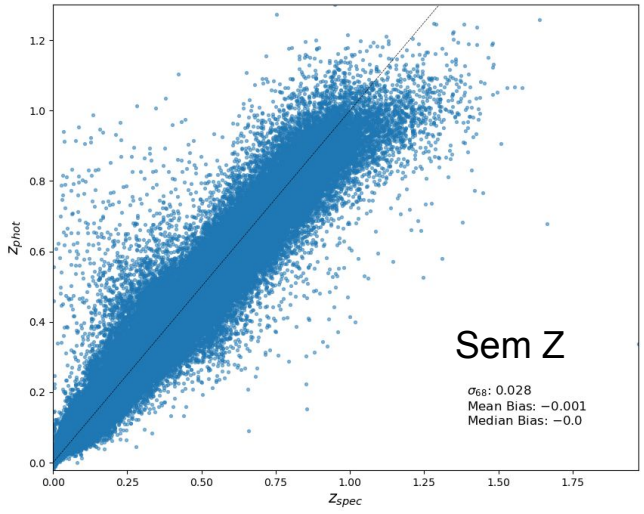

z_{phot}
z_{spec}
σ_{68}: 0.026
Mean Bias: −0.0
Median Bias: 0.001

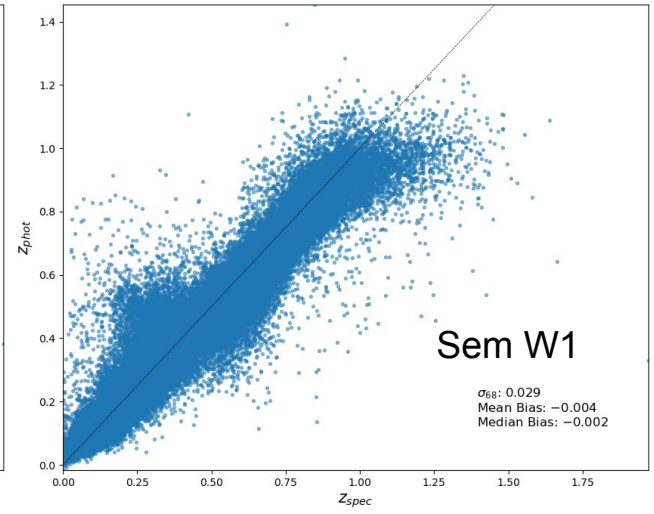
Sem uma banda

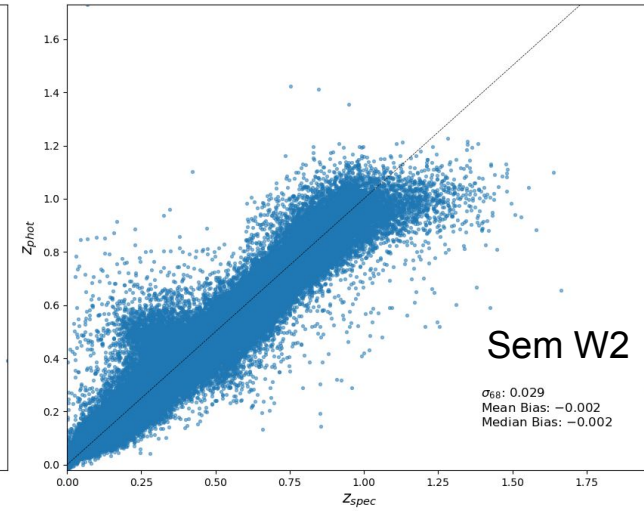

Original
σ_{68}: 0.026
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}


Sem R
σ_{68}: 0.034
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}


Sem G
σ_{68}: 0.035
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}

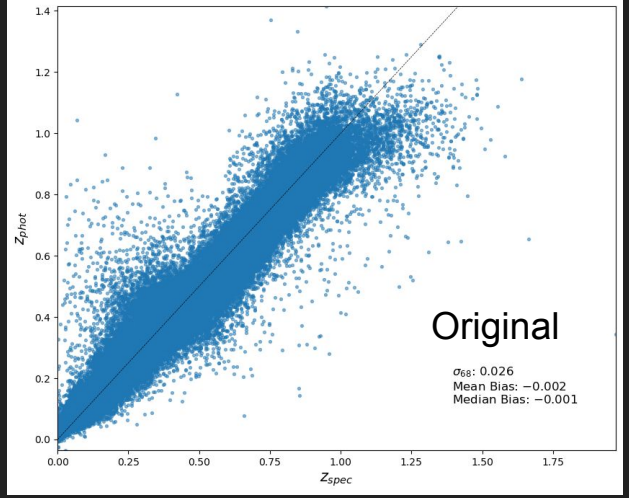

Sem Z
σ_{68}: 0.028
Mean Bias: −0.001
Median Bias: −0.0
z_{phot}
z_{spec}

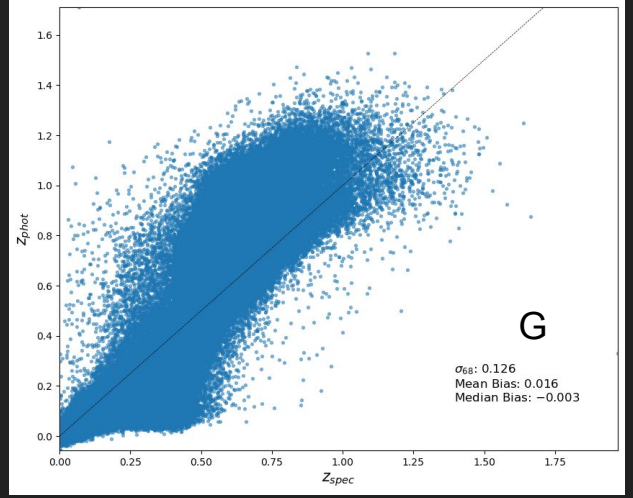

Sem W1
σ_{68}: 0.029
Mean Bias: −0.004
Median Bias: −0.002
z_{phot}
z_{spec}

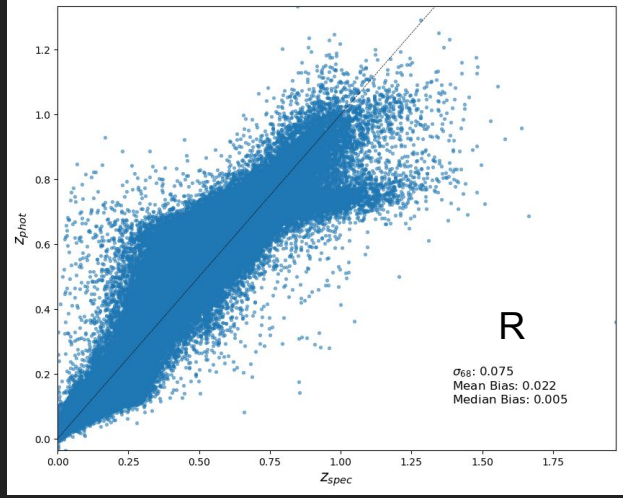

Sem W2
σ_{68}: 0.029
Mean Bias: −0.002
Median Bias: −0.002
z_{phot}
z_{spec}

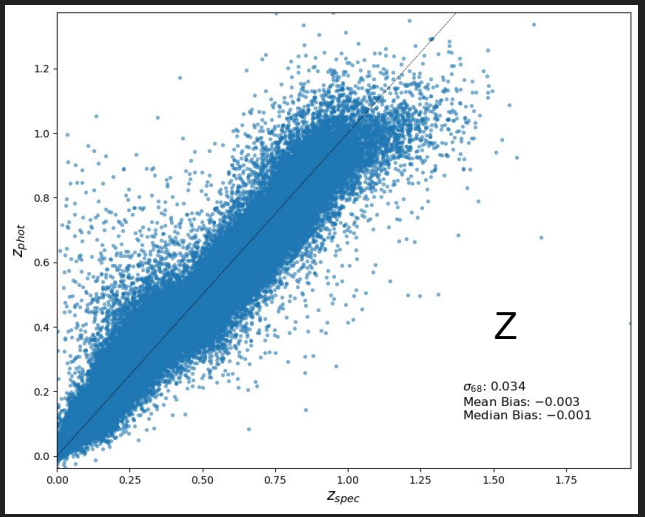
```
# Antes de embaralhar os dados vamos substituir os dados de teste
# com missing data
# ordem no espectro G R Z W1 W2
# substituindo metade dos dados da banda G por dados da banda R


cut_out = np.random.uniform(0,1,len(test_data)) < 0.5
test_data1['MAG_G'][cut_out] = test_data1['MAG_R'][cut_out]


# substituindo metade dos dados da banda  pela média

test_data2['MAG_R'][cut_out] = ((test_data2['MAG_G']+test_data2['MAG_Z'])/2)[cut_out]


test_data3['MAG_Z'][cut_out] = ((test_data3['MAG_R']+test_data3['MAG_W1'])/2)[cut_out]


test_data4['MAG_W1'][cut_out] = ((test_data4['MAG_Z']+test_data4['MAG_W2'])/2)[cut_out]

# substituindo metade dos dados da banda W2 por dados da banda W1

#test_data5


test_data5['MAG_W2'][cut_out] = test_data5['MAG_W1'][cut_out]
```

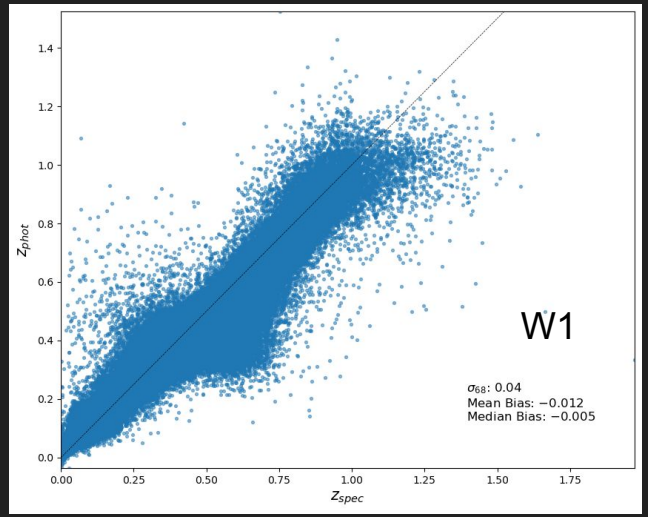
```
#vamos chamar o modelo salvo do outro notebook
regressor24 = tf.keras.models.load_model('regressor24')


preds = regressor24.predict(x_test)
preds1 = regressor24.predict(x_test1)
preds2 = regressor24.predict(x_test2)
preds3 = regressor24.predict(x_test3)
preds4 = regressor24.predict(x_test4)
preds5 = regressor24.predict(x_test5)
```

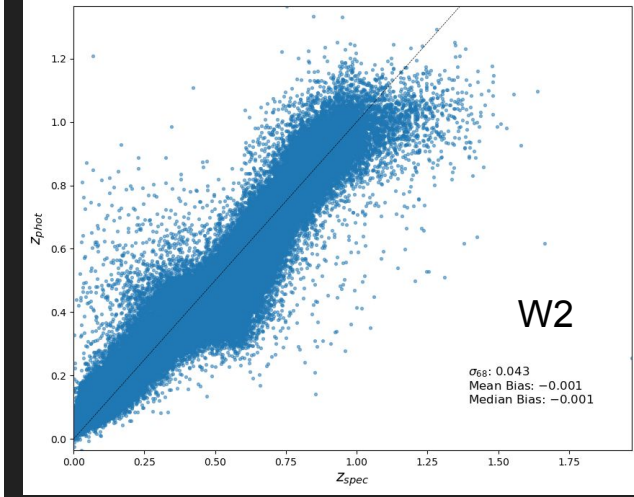

Original
σ_{68}: 0.026
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}


G
σ_{68}: 0.126
Mean Bias: 0.016
Median Bias: −0.003
z_{phot}
z_{spec}

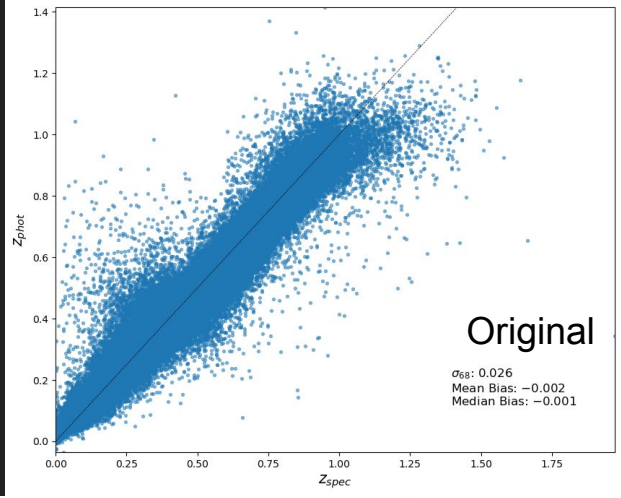

R
σ_{68}: 0.075
Mean Bias: 0.022
Median Bias: 0.005
z_{phot}
z_{spec}

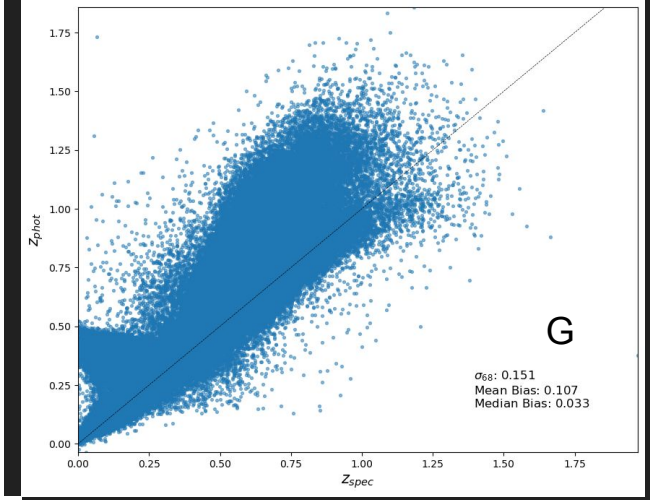

Z
σ_{68}: 0.034
Mean Bias: −0.003
Median Bias: −0.001
z_{phot}
z_{spec}

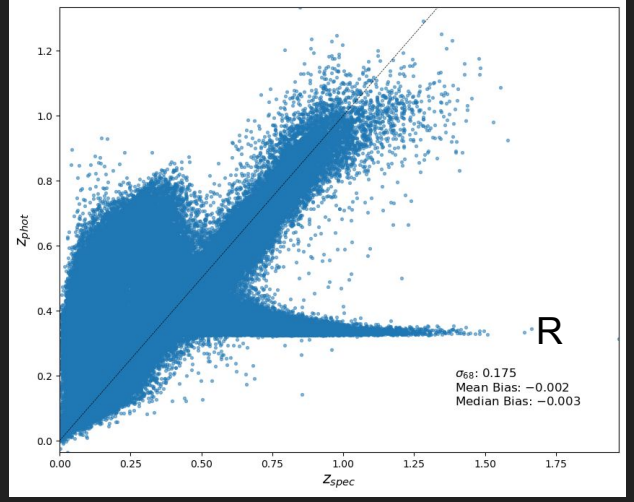

W1
σ_{68}: 0.04
Mean Bias: −0.012
Median Bias: −0.005
z_{phot}
z_{spec}

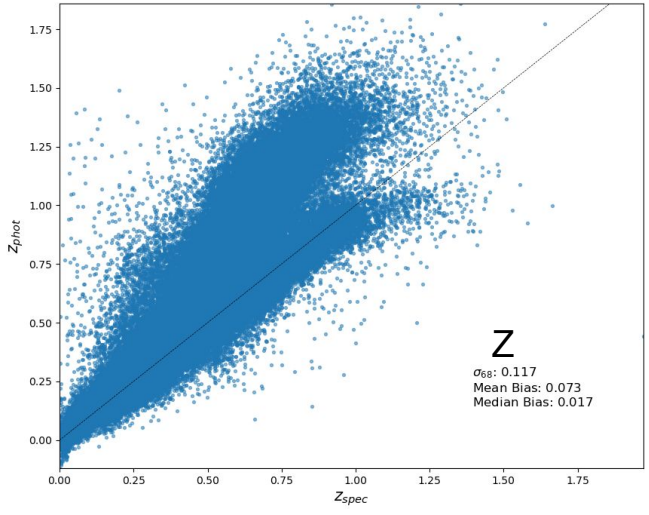

W2
σ_{68}: 0.043
Mean Bias: −0.001
Median Bias: −0.001
z_{phot}
z_{spec}

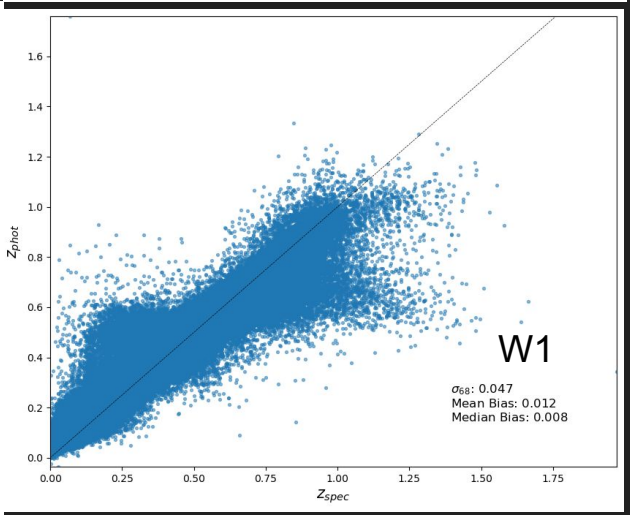
```
test_data1['MAG_G']=test_data1['MAG_G'].fillna(test_data1['MAG_G'].mean())

test_data2['MAG_R']=test_data2['MAG_R'].fillna(test_data2['MAG_R'].mean())

test_data3['MAG_Z']=test_data3['MAG_Z'].fillna(test_data3['MAG_Z'].mean())

test_data4['MAG_W1']=test_data4['MAG_W1'].fillna(test_data4['MAG_W1'].mean())

test_data5['MAG_W2']=test_data5['MAG_W2'].fillna(test_data5['MAG_W2'].mean())
```

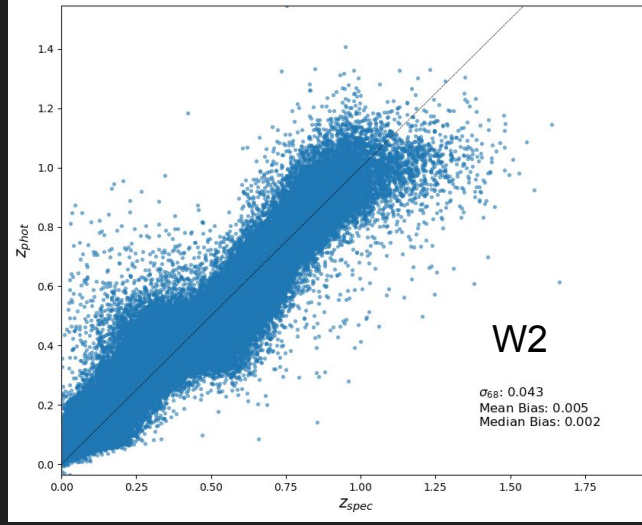

Original
σ_{68}: 0.026
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}


G
σ_{68}: 0.151
Mean Bias: 0.107
Median Bias: 0.033
z_{phot}
z_{spec}


R
σ_{68}: 0.175
Mean Bias: −0.002
Median Bias: −0.003
z_{phot}
z_{spec}

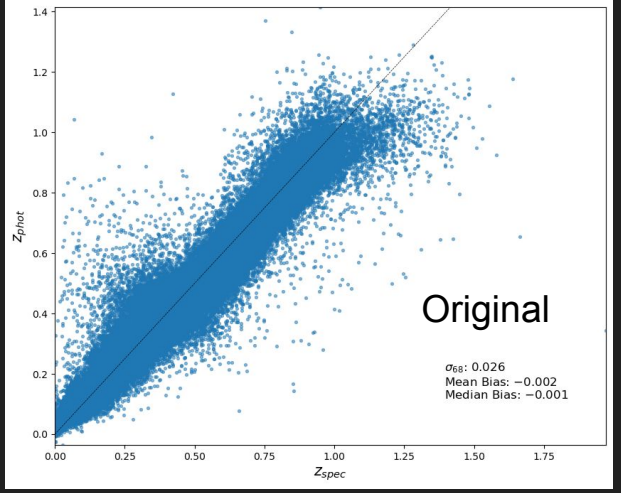

Z
σ_{68}: 0.117
Mean Bias: 0.073
Median Bias: 0.017
z_{phot}
z_{spec}

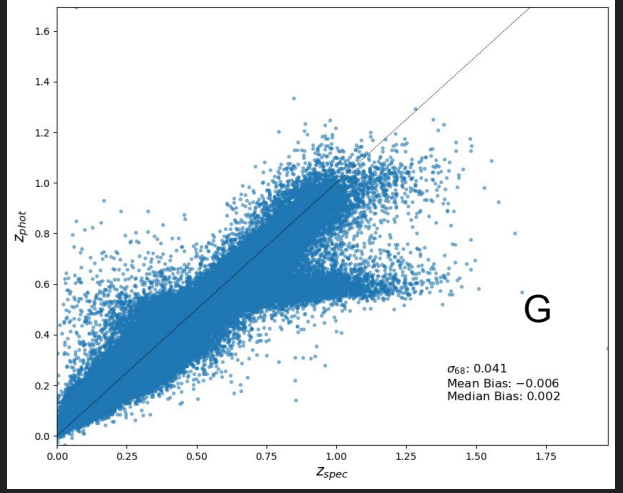

W1
σ_{68}: 0.047
Mean Bias: 0.012
Median Bias: 0.008
z_{phot}
z_{spec}

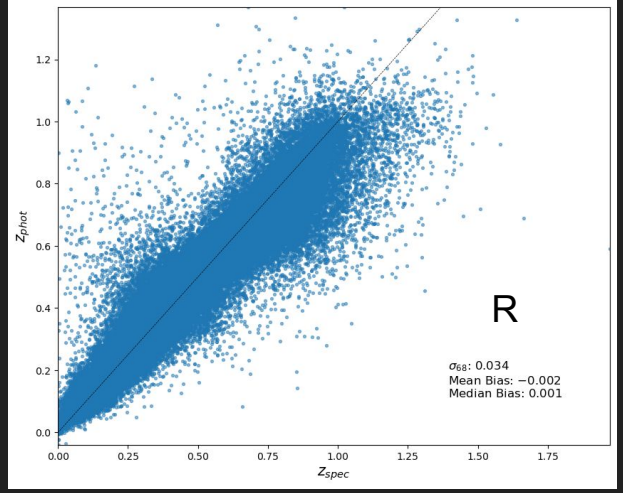

W2
σ_{68}: 0.043
Mean Bias: 0.005
Median Bias: 0.002
z_{phot}
z_{spec}

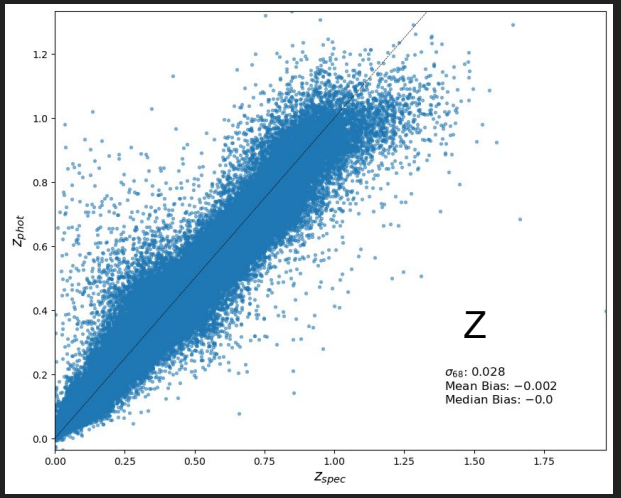
```
from sklearn.linear_model import LinearRegression

lr1 = LinearRegression()

testdf1 = test_data1[test_data1['MAG_G'].isnull()==True]
traindf1 = test_data1[test_data1['MAG_G'].isnull()==False]
y1 = traindf1['MAG_G']

traindf1.drop("MAG_G",axis=1,inplace=True)
traindf1.drop("zspec",axis=1,inplace=True)
lr1.fit(traindf1,y1)

testdf1.drop("MAG_G",axis=1,inplace=True)
testdf1.drop("zspec",axis=1,inplace=True)

pred1 = lr1.predict(testdf1)
```

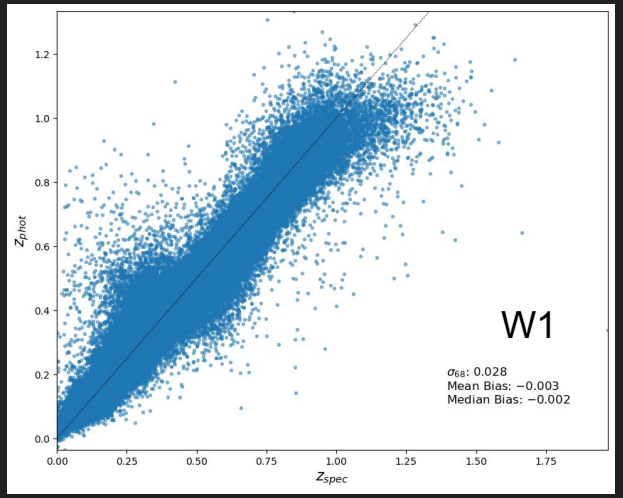
```
test_data1['MAG_G'][cut_out_G] = pred1
test_data2['MAG_R'][cut_out_G] = pred2
test_data3['MAG_Z'][cut_out_G] = pred3
test_data4['MAG_W1'][cut_out_G] = pred4
test_data5['MAG_W2'][cut_out_G] = pred5
```

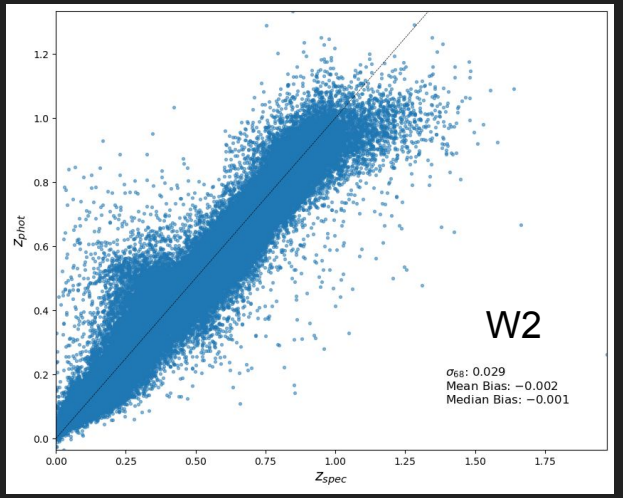

Original
σ_{68}: 0.026
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}

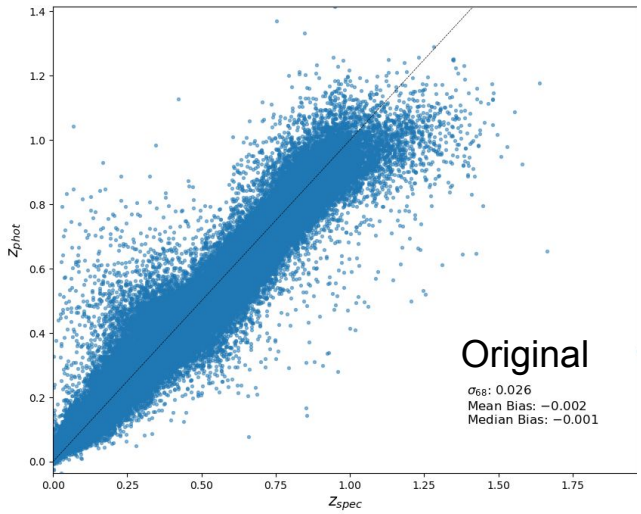

G
σ_{68}: 0.041
Mean Bias: −0.006
Median Bias: 0.002
z_{phot}
z_{spec}

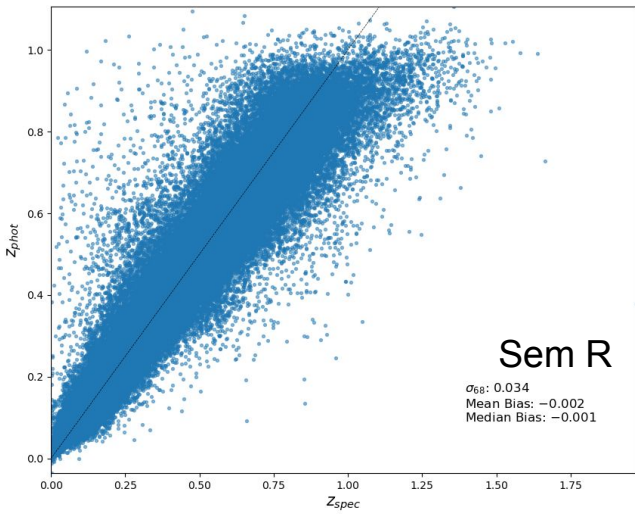

R
σ_{68}: 0.034
Mean Bias: −0.002
Median Bias: 0.001
z_{phot}
z_{spec}

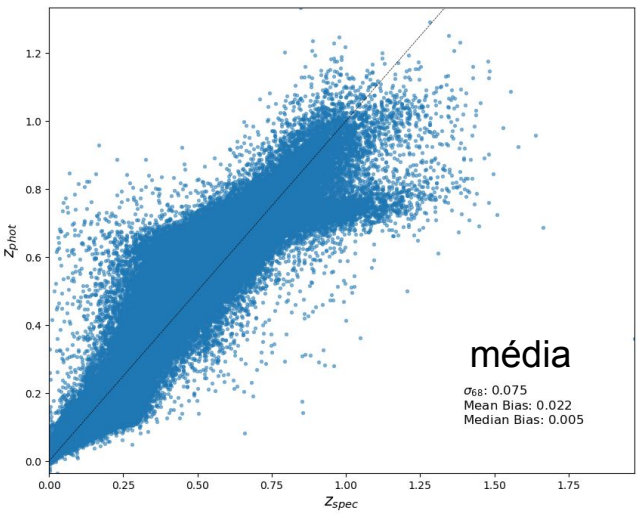

Z
σ_{68}: 0.028
Mean Bias: −0.002
Median Bias: −0.0
z_{phot}
z_{spec}

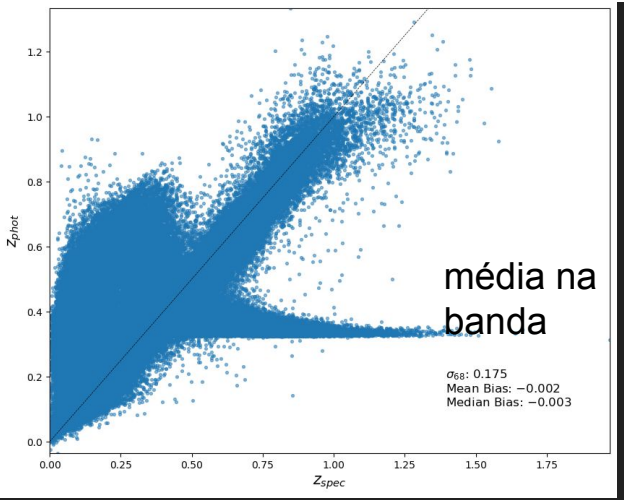

W1
σ_{68}: 0.028
Mean Bias: −0.003
Median Bias: −0.002
z_{phot}
z_{spec}

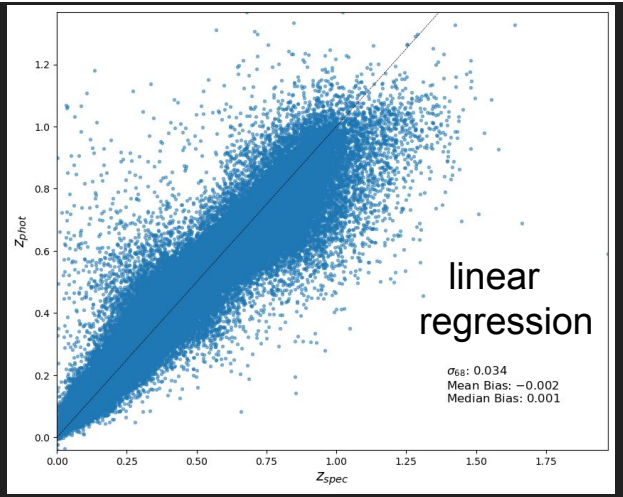

W2
σ_{68}: 0.029
Mean Bias: −0.002
Median Bias: −0.001
z_{phot}
z_{spec}



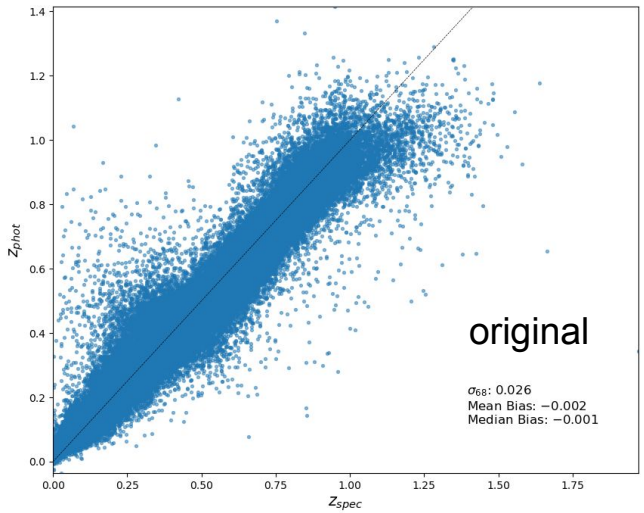


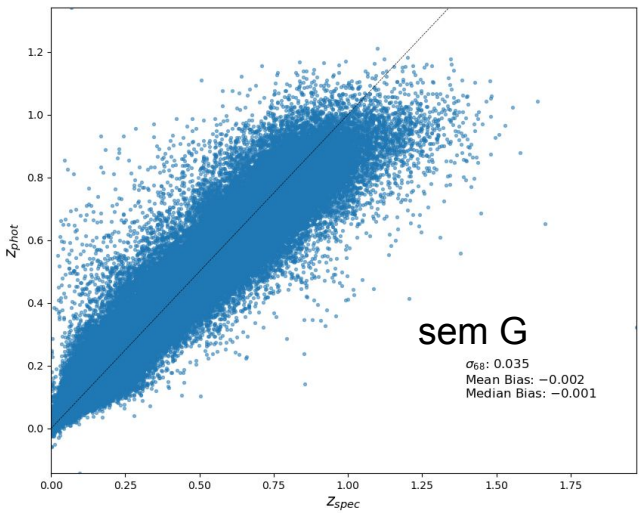


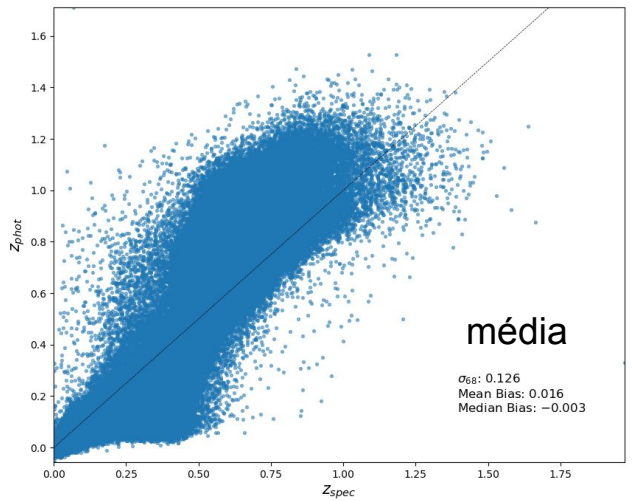


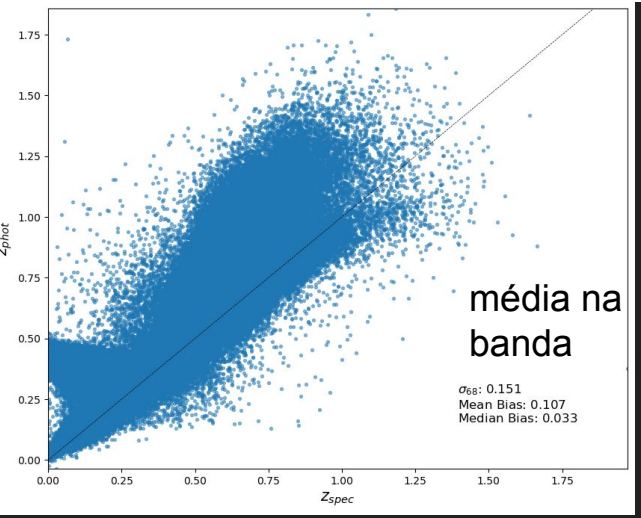


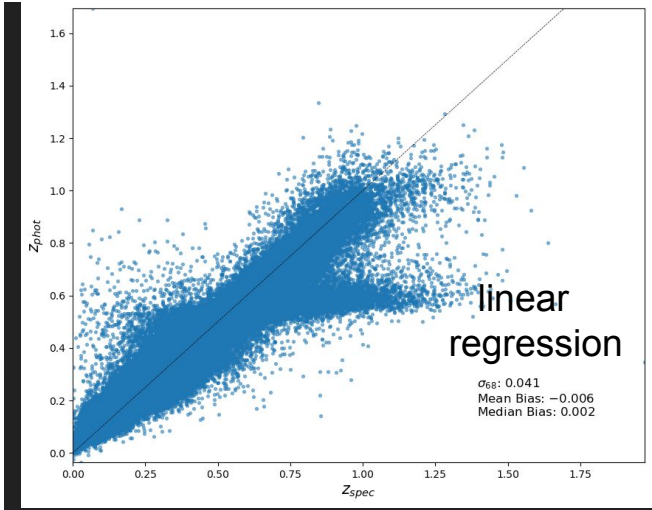
Comparação para a banda R



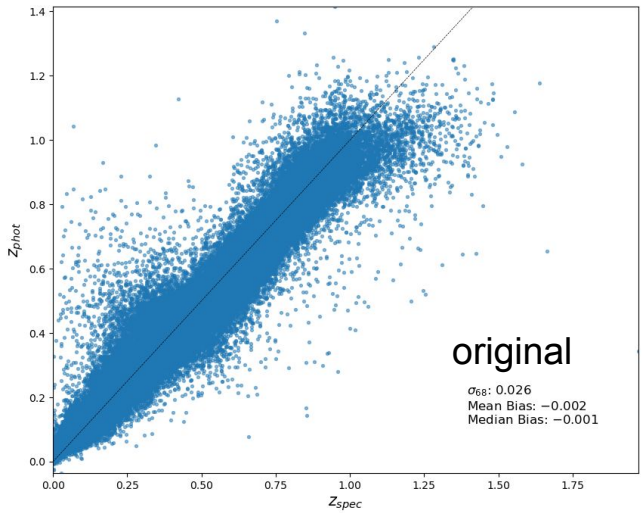


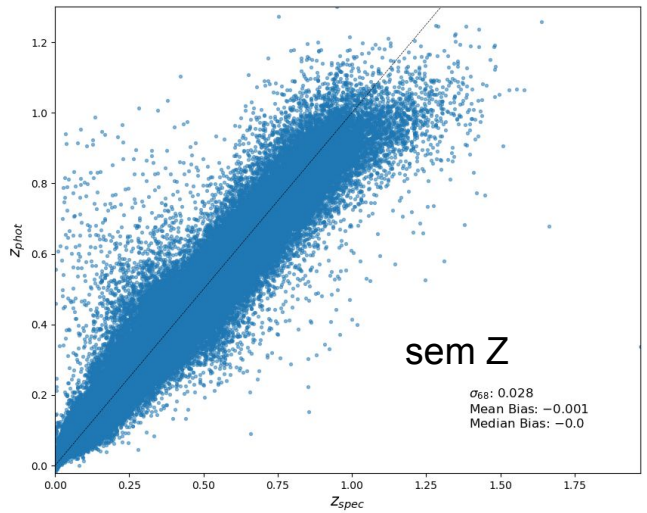


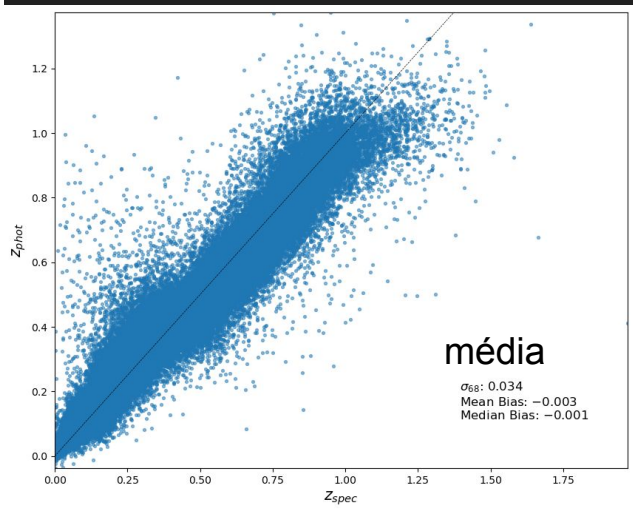


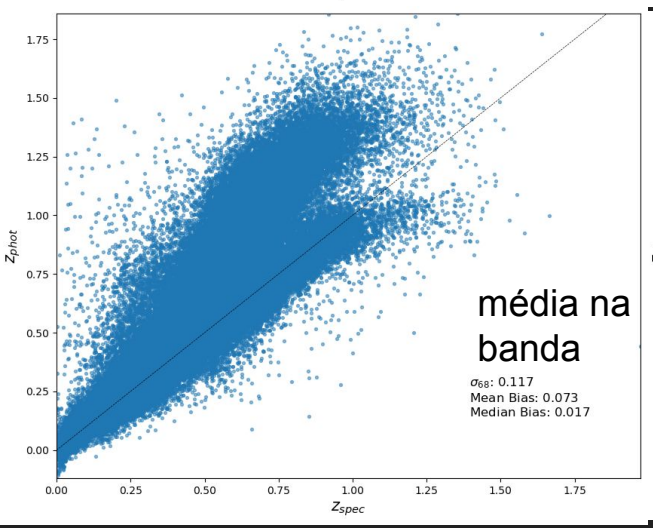


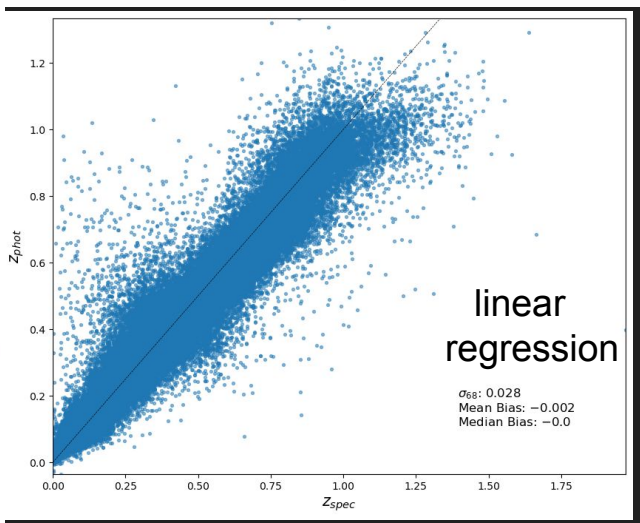
Comparação na banda G



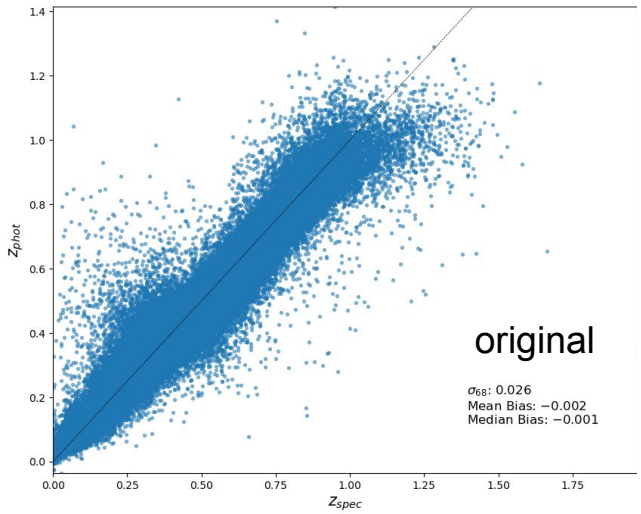


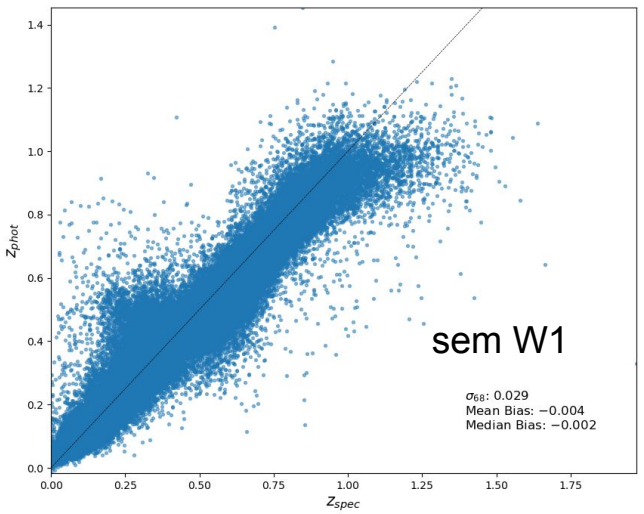


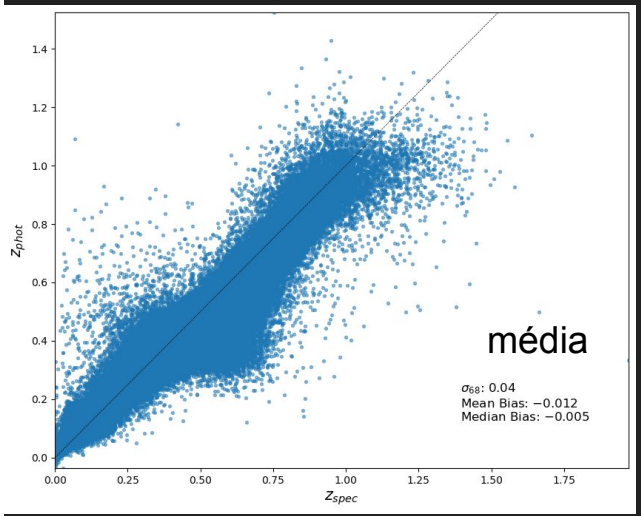


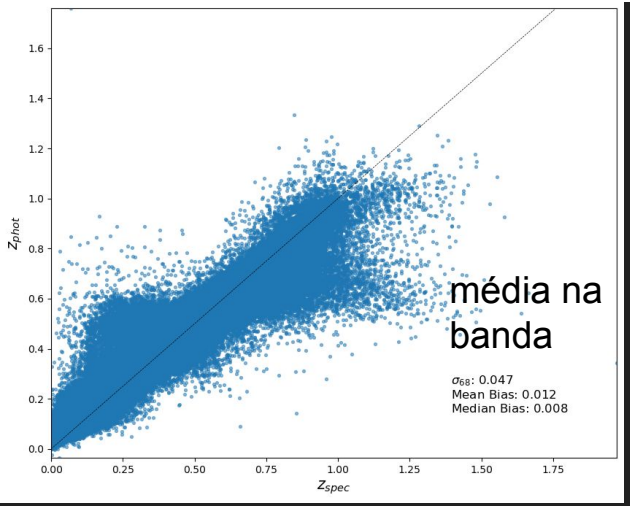


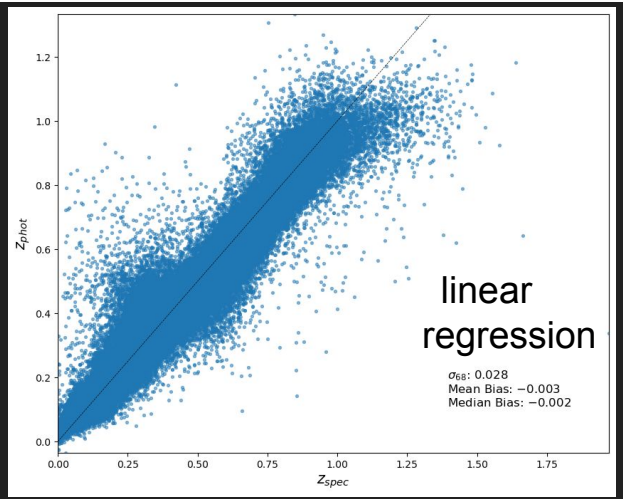
Comparação na banda Z



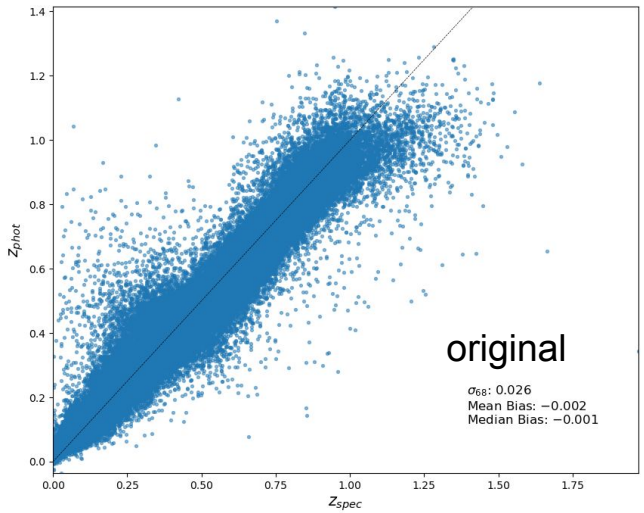


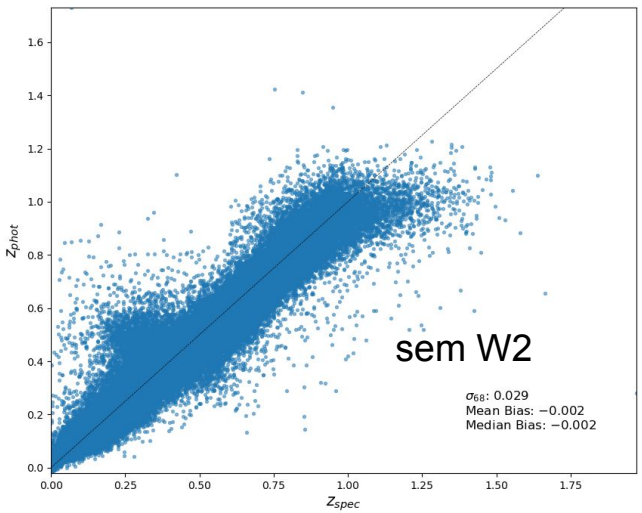


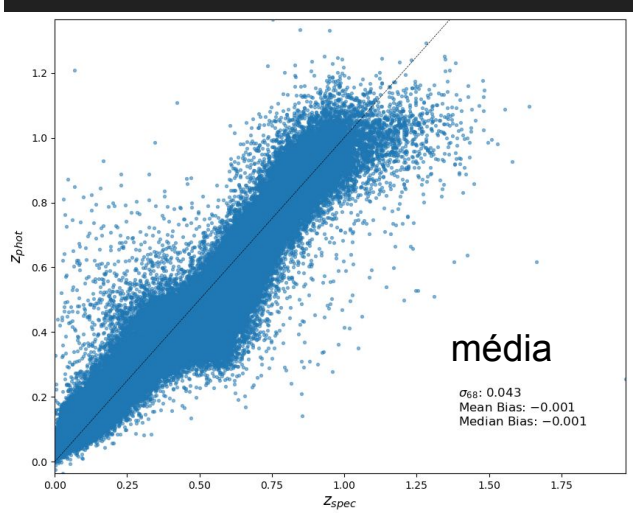


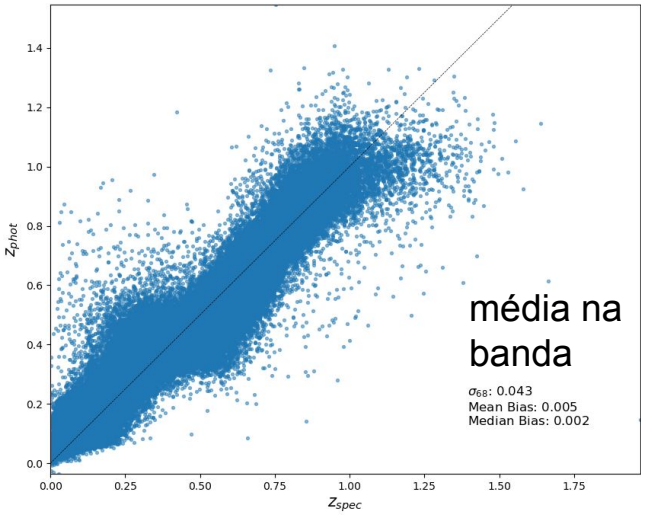


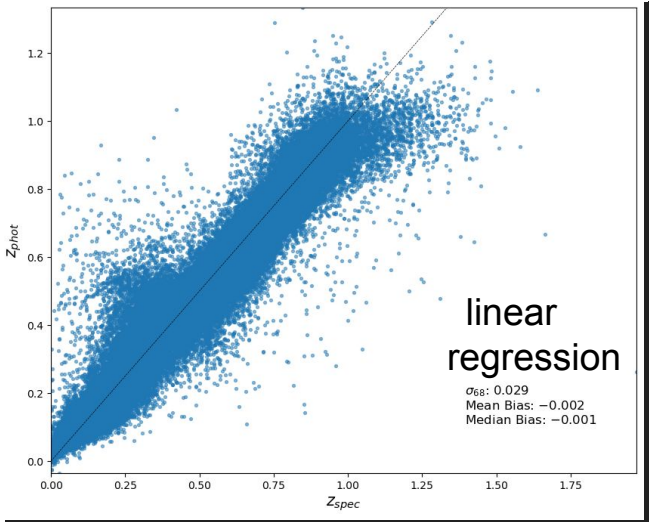
Comparação na banda W1









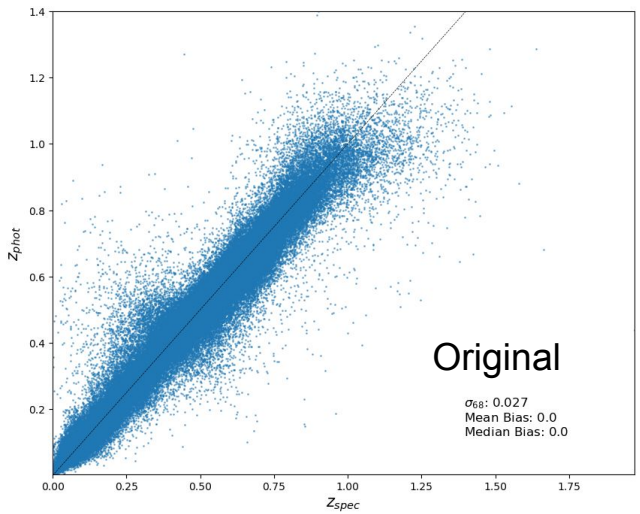


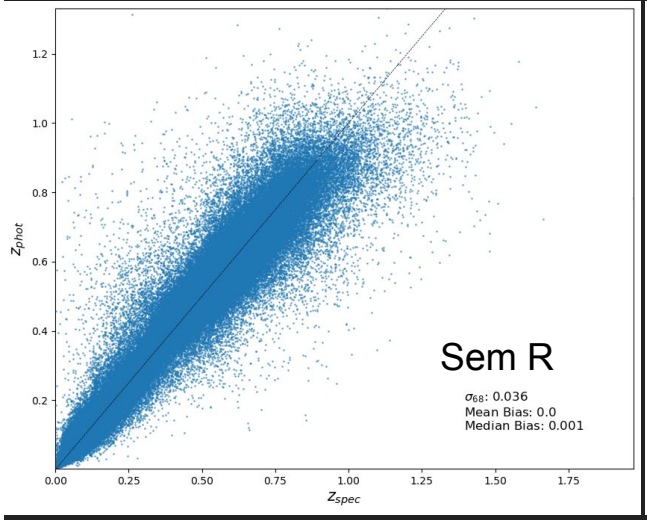
Comparação para banda W2

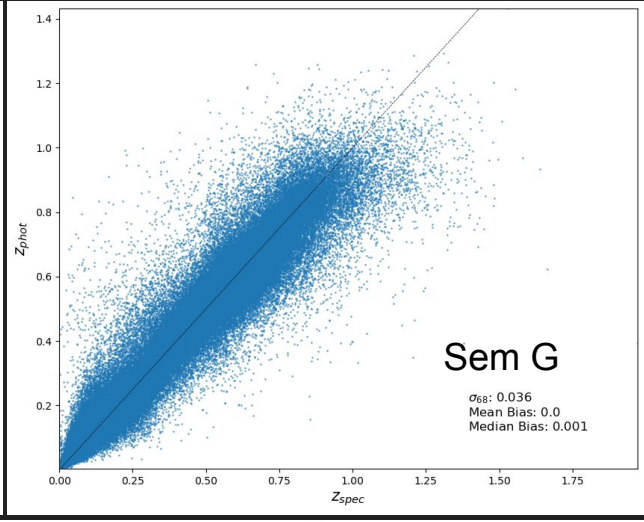
Random forest

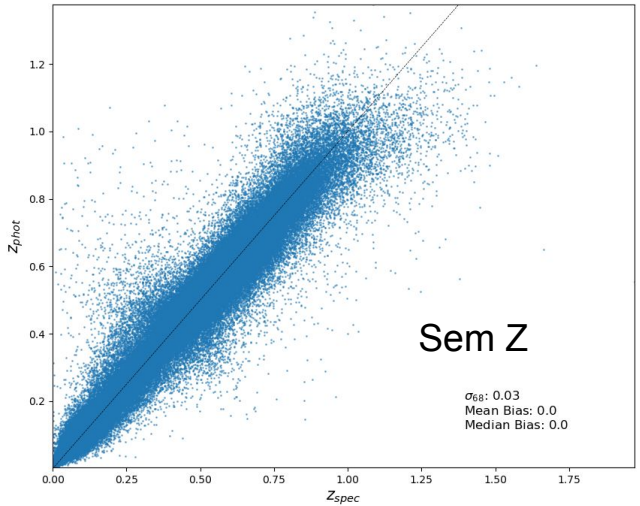
Preparação dos dados é igual

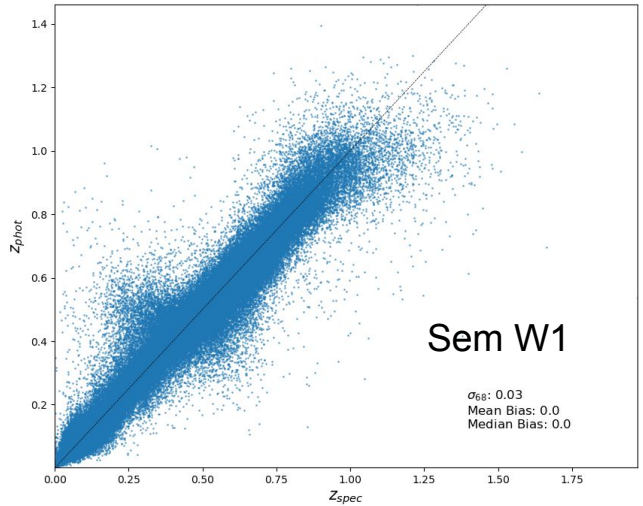
```
import pickle
from sklearn.ensemble import RandomForestRegressor

regressor1 = RandomForestRegressor(n_estimators=10)
regr1 = regressor1.fit(x_train1, y_train1)

pickle.dump(regr1, open('ramdom1.sav', 'wb'))

preds1a = regr1.predict(x_val1)
preds1b = regr1.predict(x_test)
#----------------------------------------------------------
```

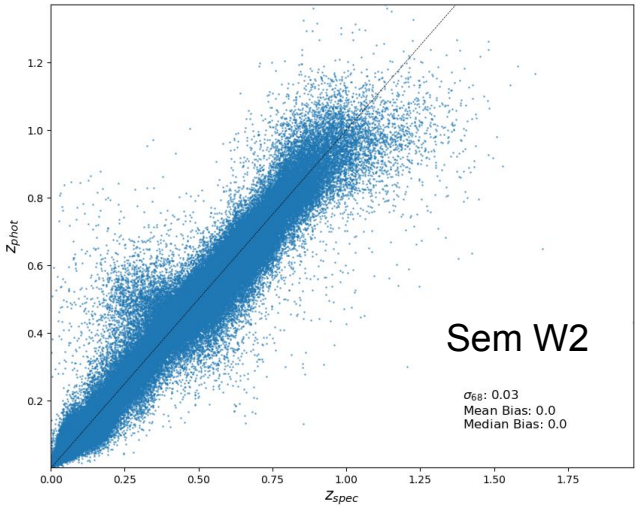

Original
σ_{68}: 0.027
Mean Bias: 0.0
Median Bias: 0.0
z_{phot}
z_{spec}

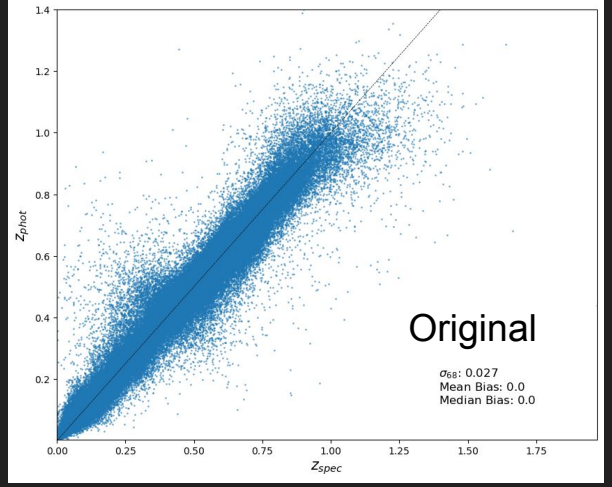

Sem R
σ_{68}: 0.036
Mean Bias: 0.0
Median Bias: 0.001
z_{phot}
z_{spec}

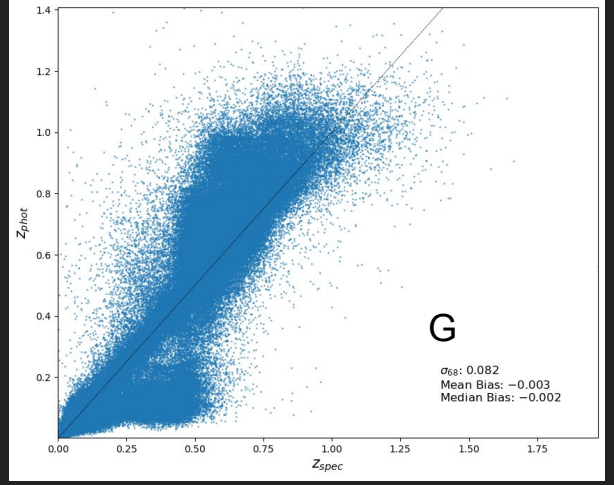

Sem G
σ_{68}: 0.036
Mean Bias: 0.0
Median Bias: 0.001
z_{phot}
z_{spec}

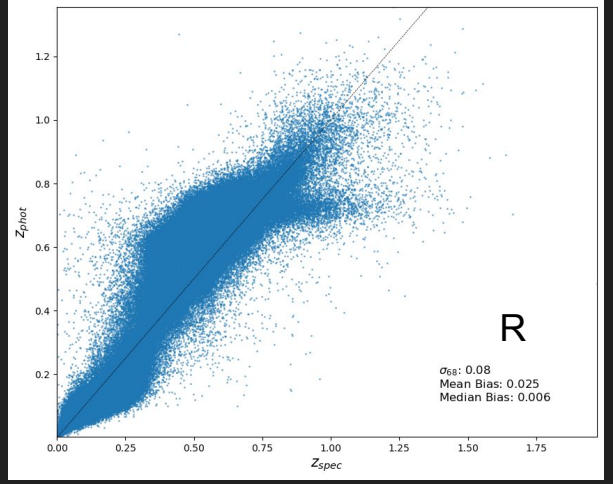

Sem Z
σ_{68}: 0.03
Mean Bias: 0.0
Median Bias: 0.0
z_{phot}
z_{spec}

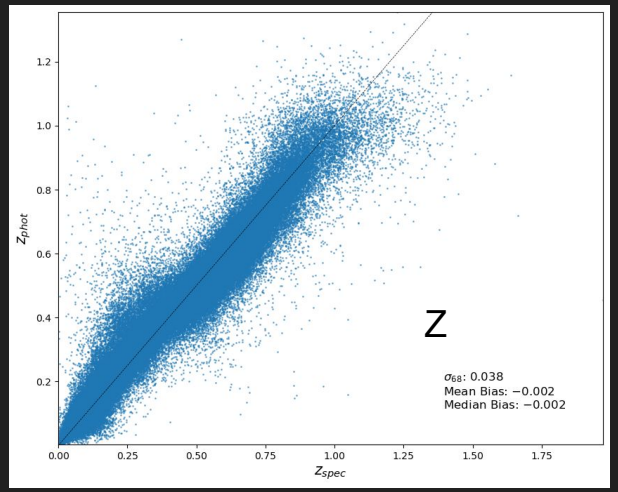

Sem W1
σ_{68}: 0.03
Mean Bias: 0.0
Median Bias: 0.0
z_{phot}
z_{spec}

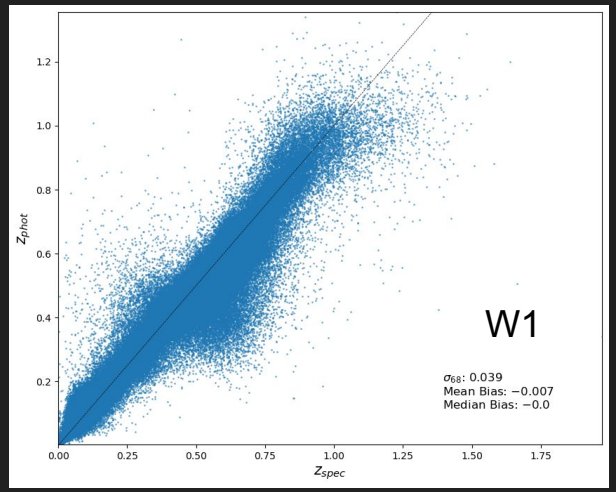

Sem W2
σ_{68}: 0.03
Mean Bias: 0.0
Median Bias: 0.0
z_{phot}
z_{spec}

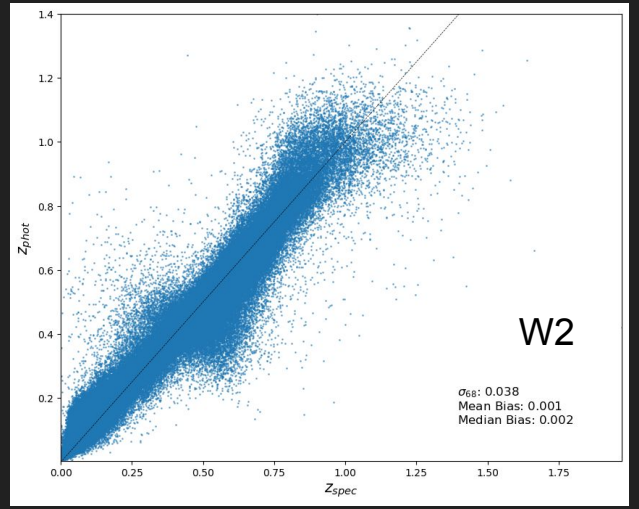

Original
σ_{68}: 0.027
Mean Bias: 0.0
Median Bias: 0.0
z_{phot}
z_{spec}


G
σ_{68}: 0.082
Mean Bias: −0.003
Median Bias: −0.002
z_{phot}
z_{spec}

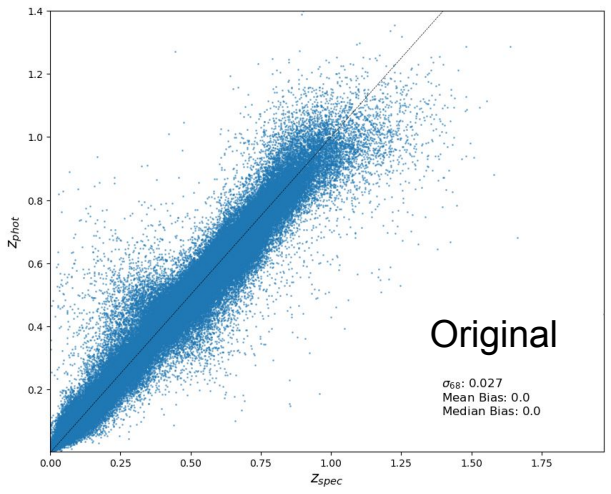

R
σ_{68}: 0.08
Mean Bias: 0.025
Median Bias: 0.006
z_{phot}
z_{spec}

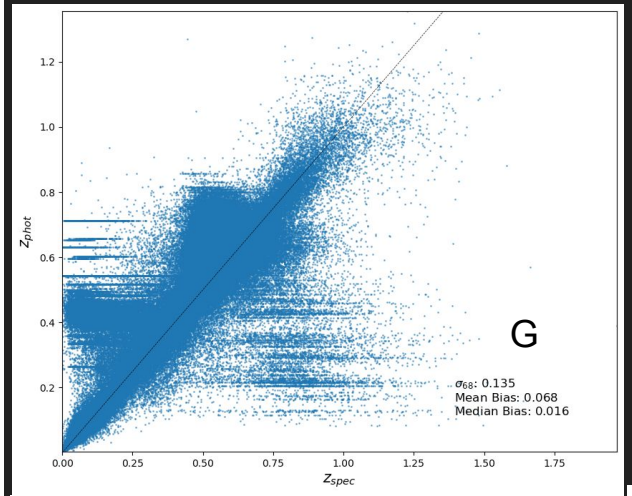

Z
σ_{68}: 0.038
Mean Bias: −0.002
Median Bias: −0.002
z_{phot}
z_{spec}

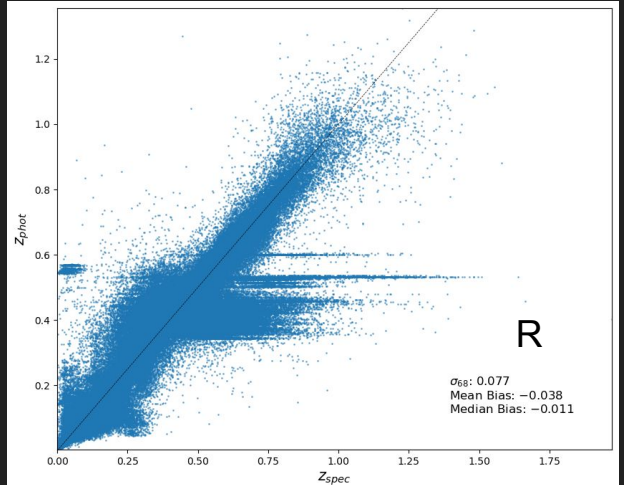

W1
σ_{68}: 0.039
Mean Bias: −0.007
Median Bias: −0.0
z_{phot}
z_{spec}

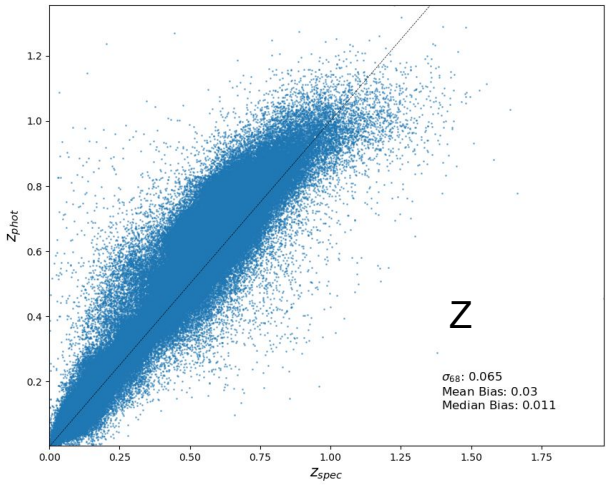

W2
σ_{68}: 0.038
Mean Bias: 0.001
Median Bias: 0.002
z_{phot}
z_{spec}

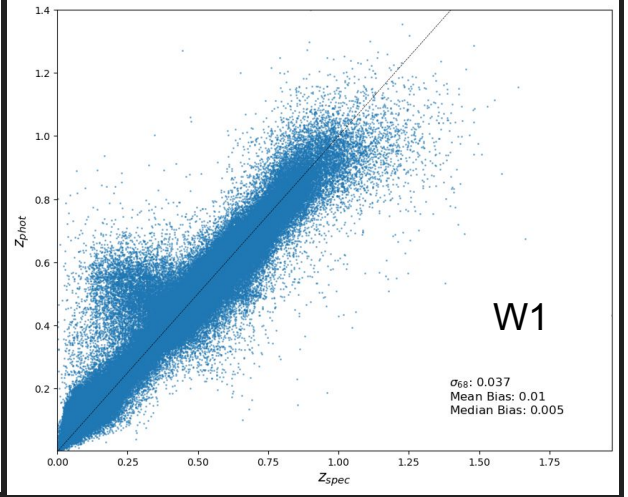
média

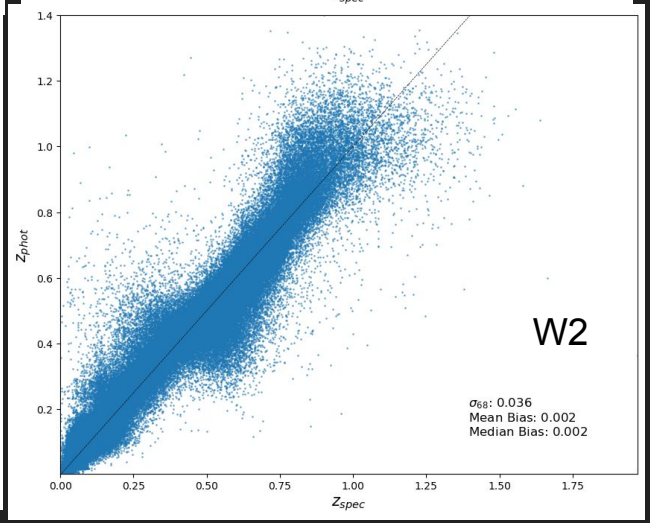




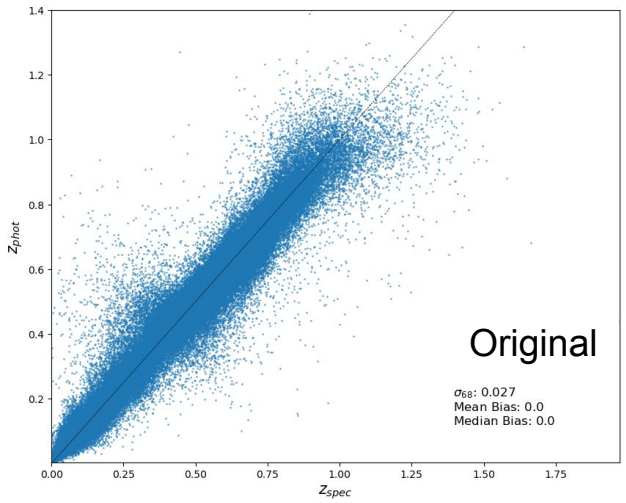


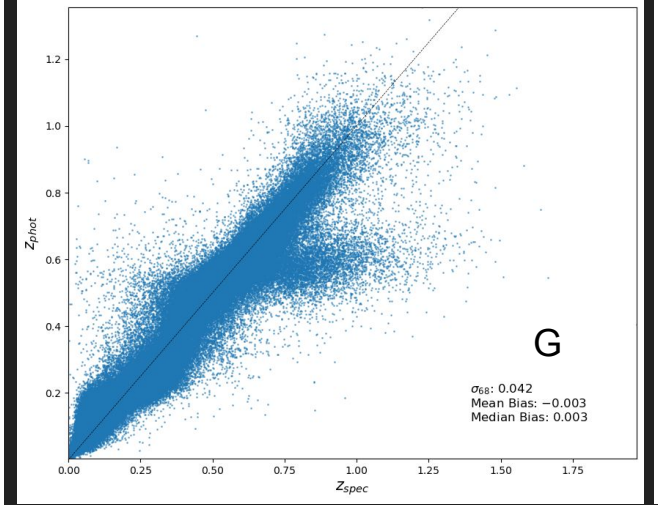


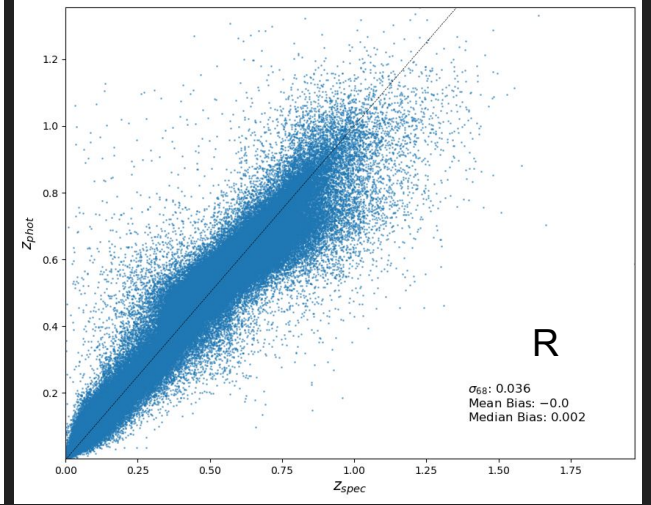


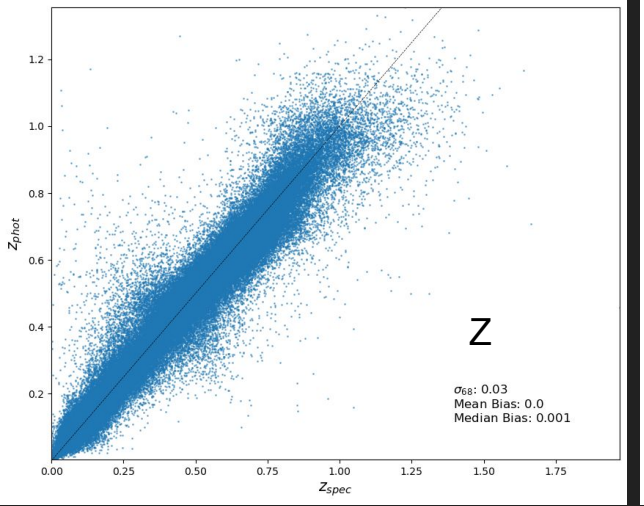


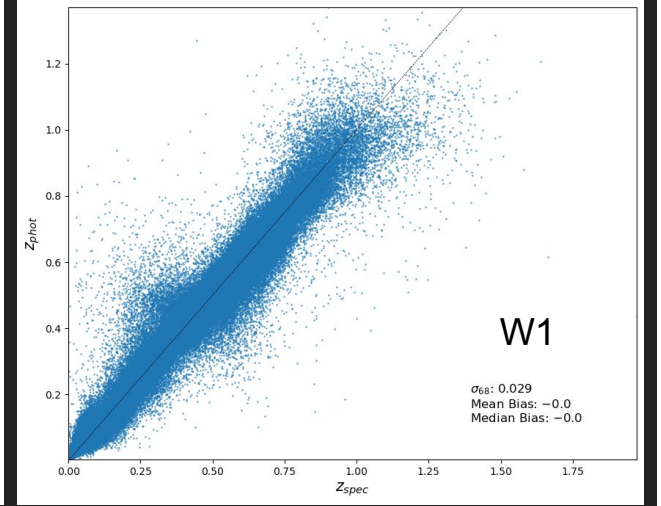
média na banda

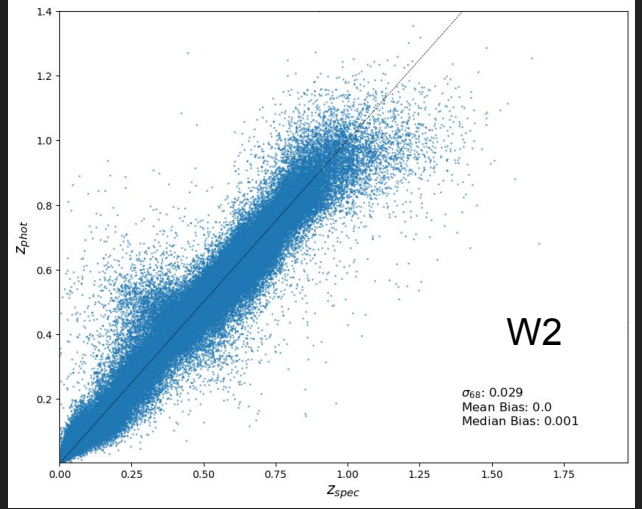




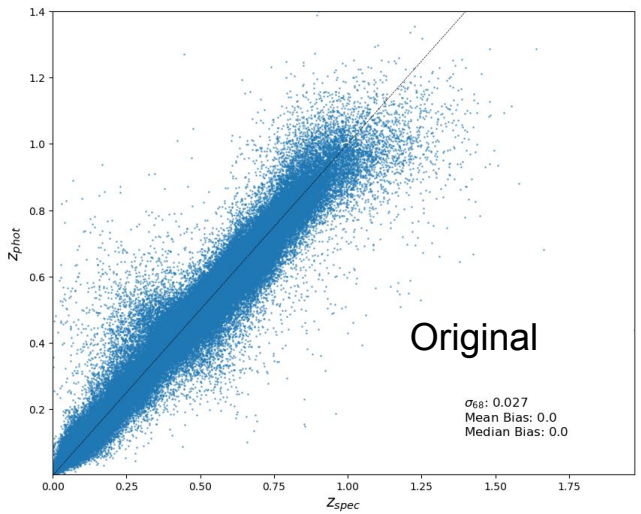


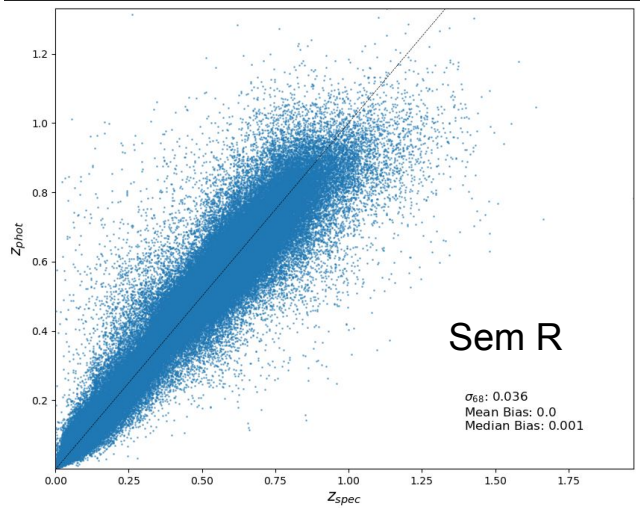


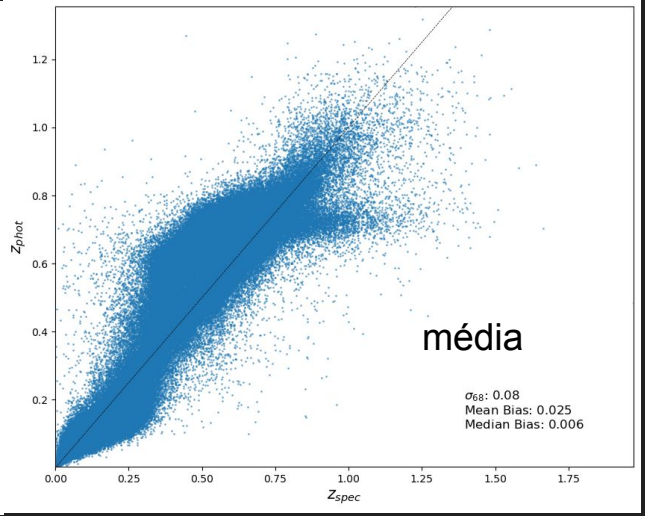


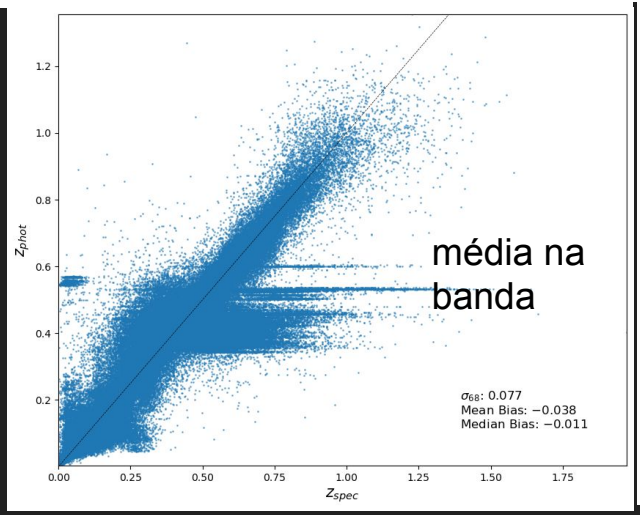


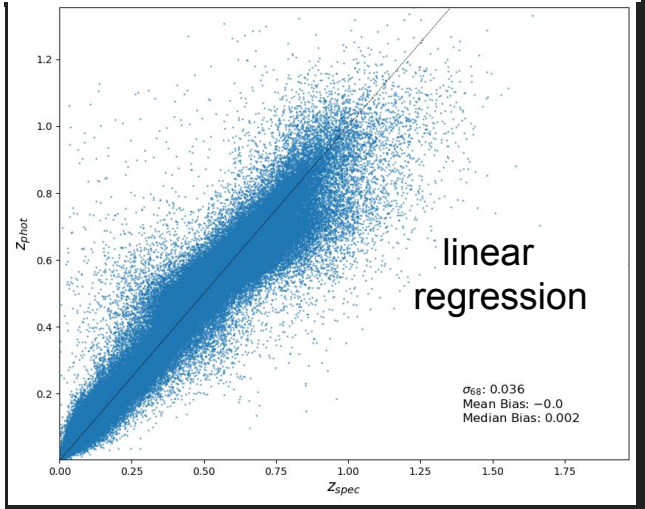
linear regression



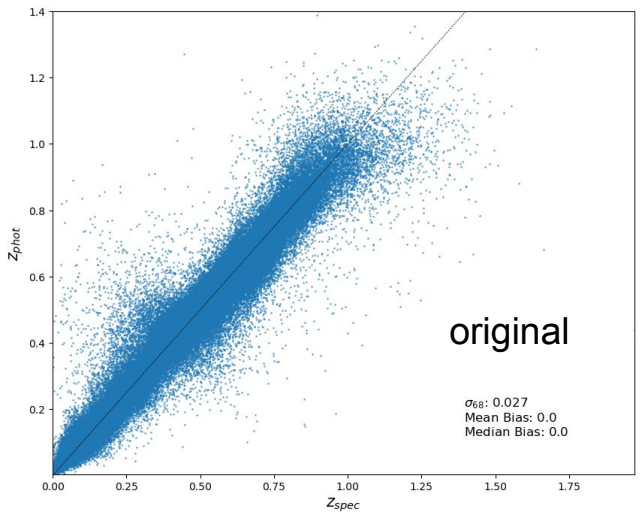


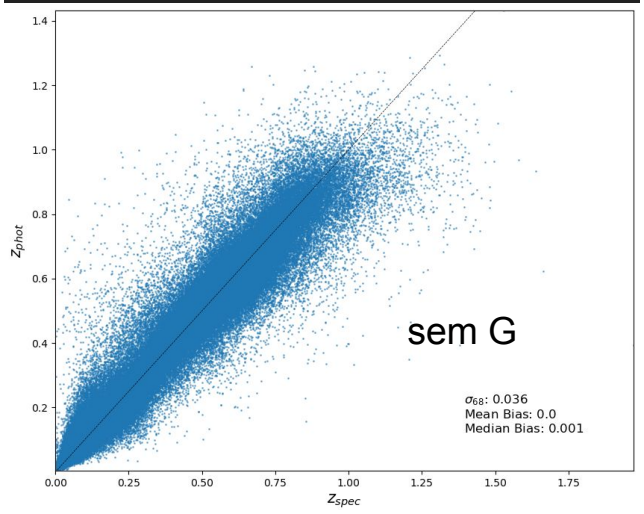


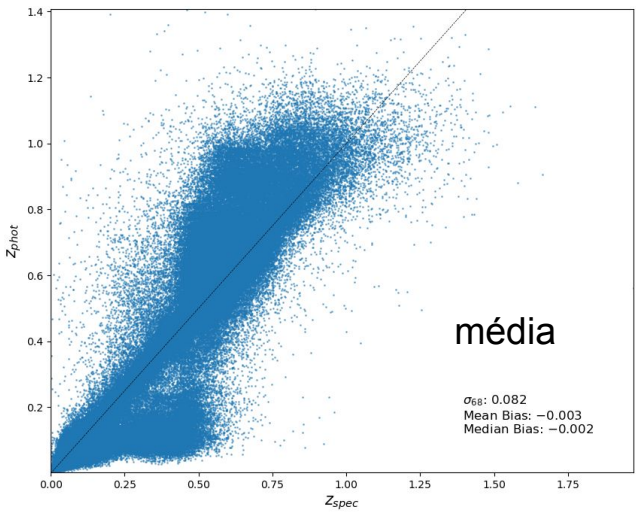


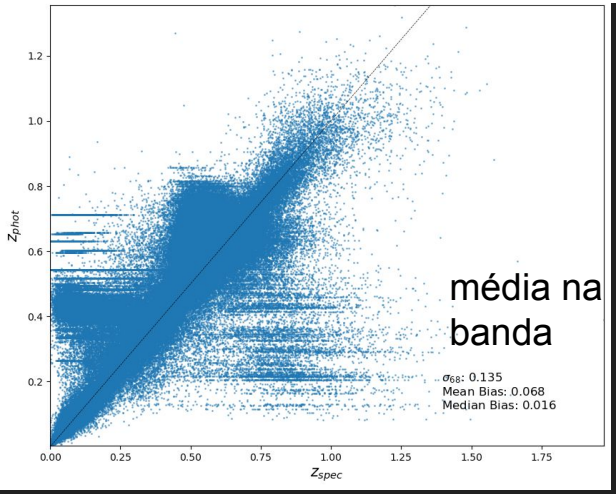


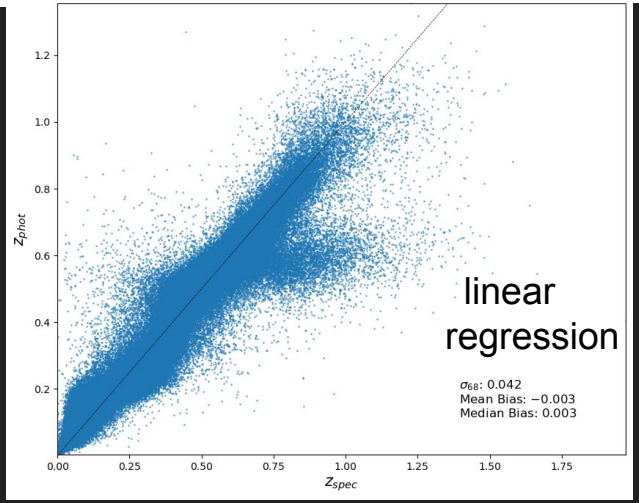
Comparação para a banda R



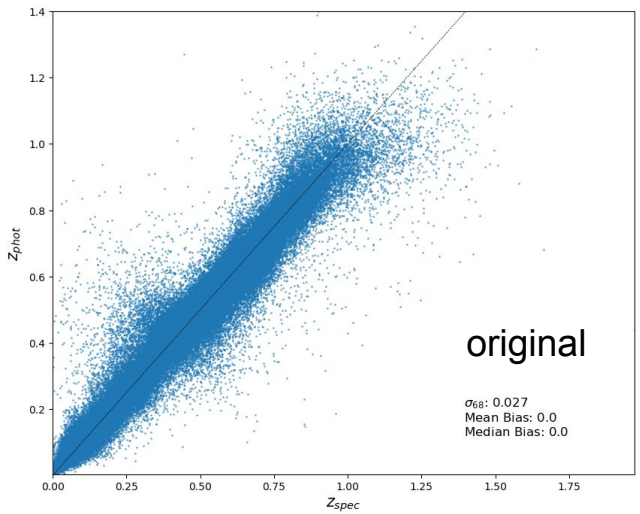


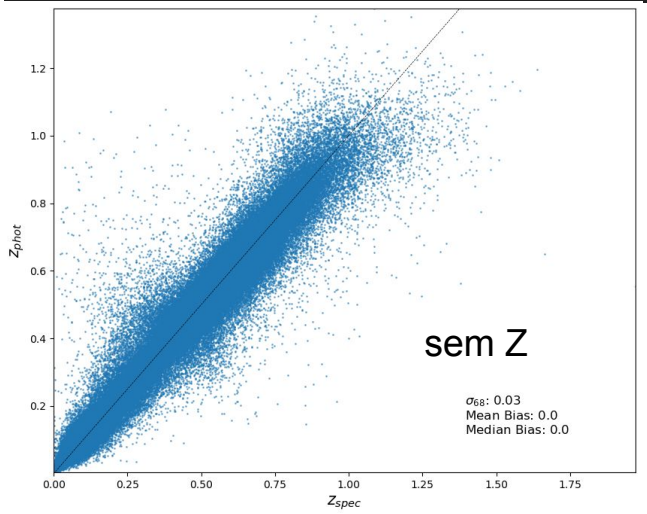


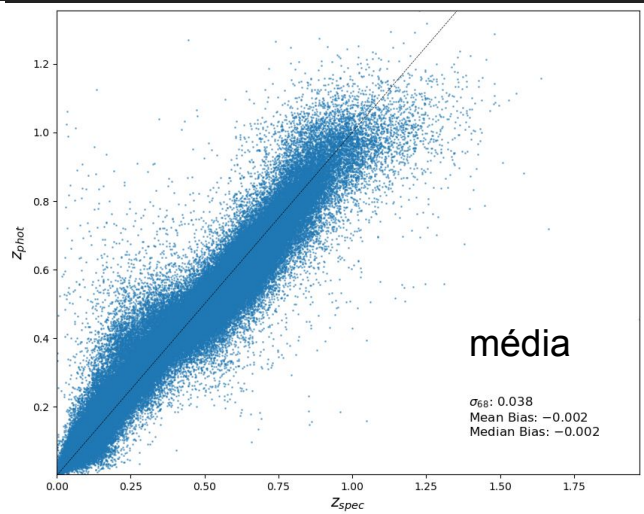


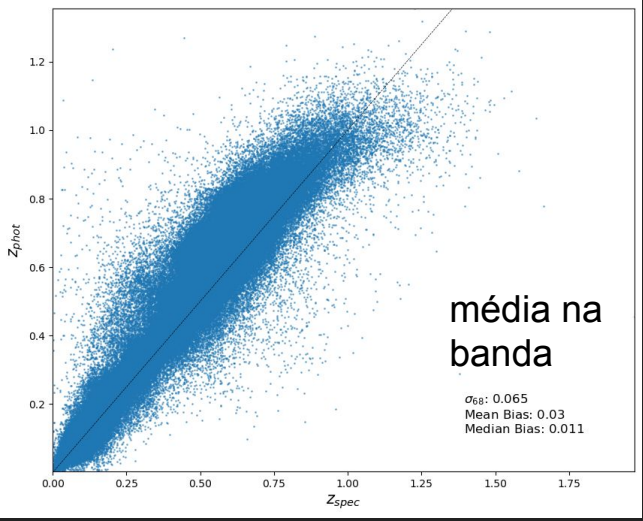


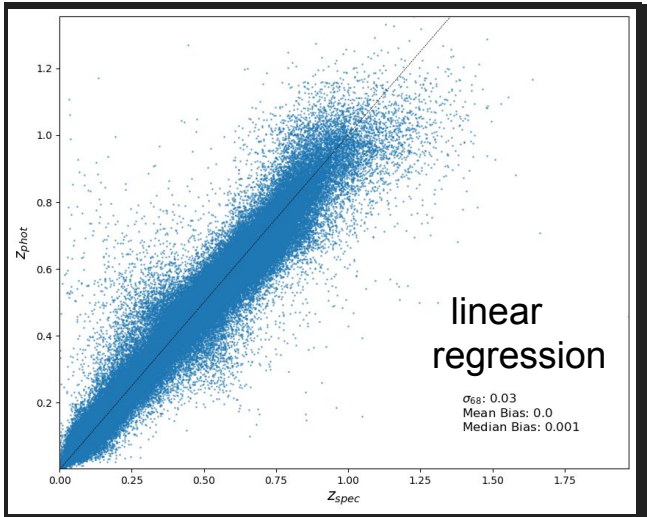
Comparação na banda G



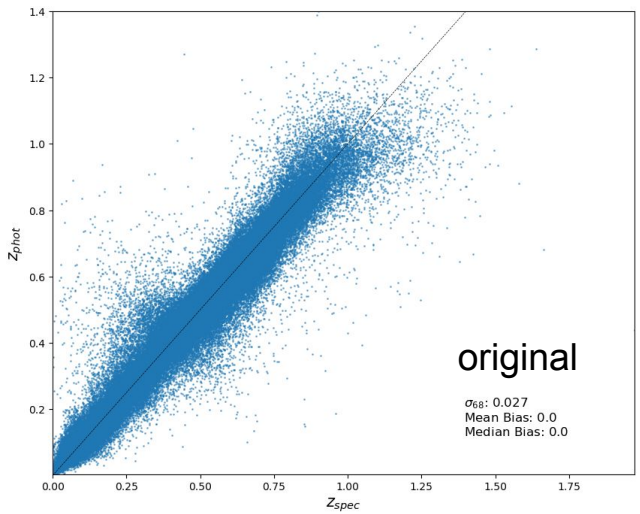


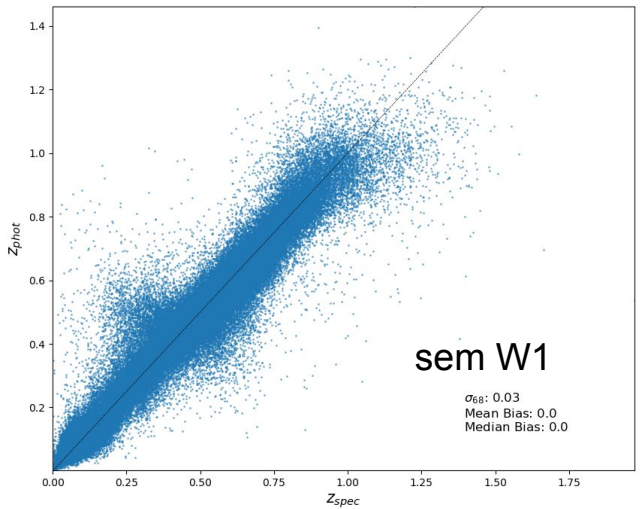


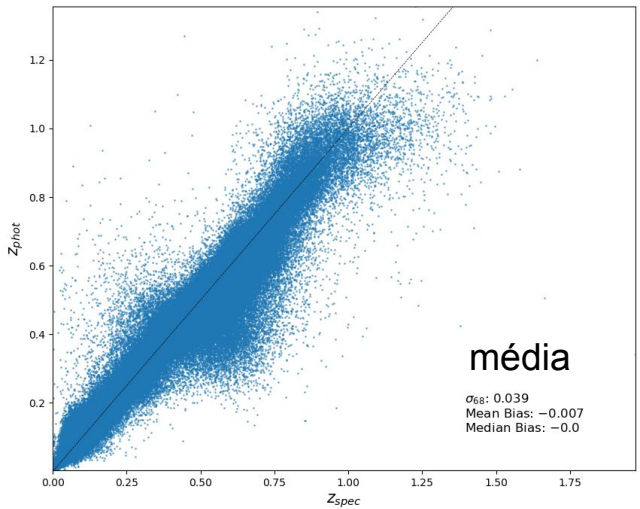


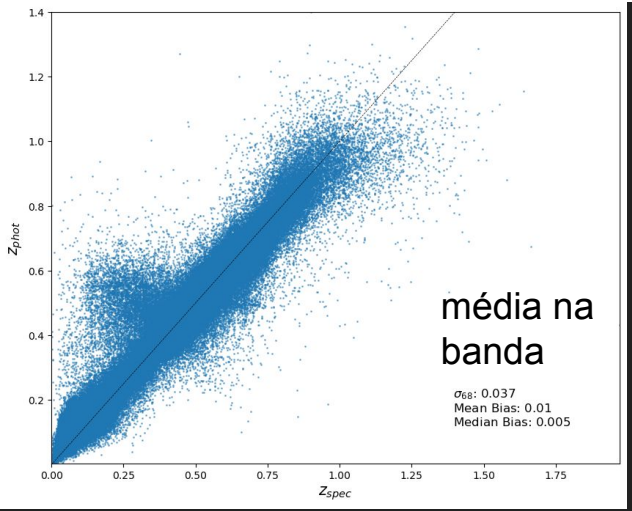


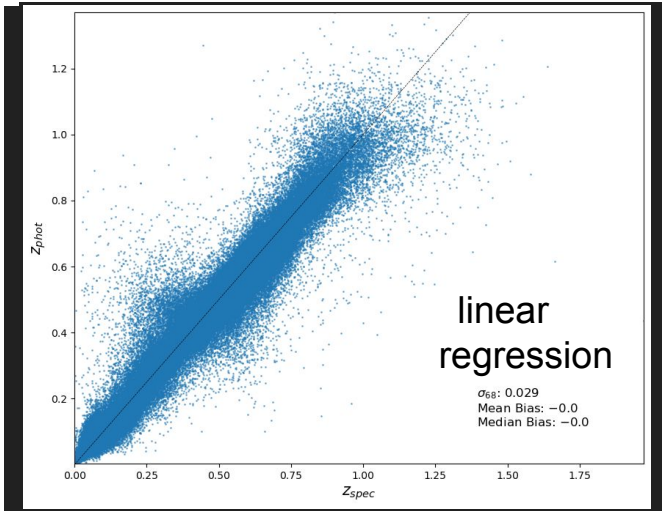
Comparação na banda Z



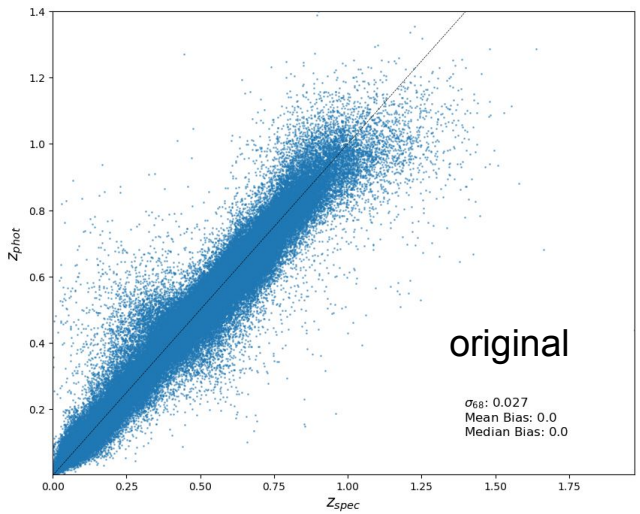


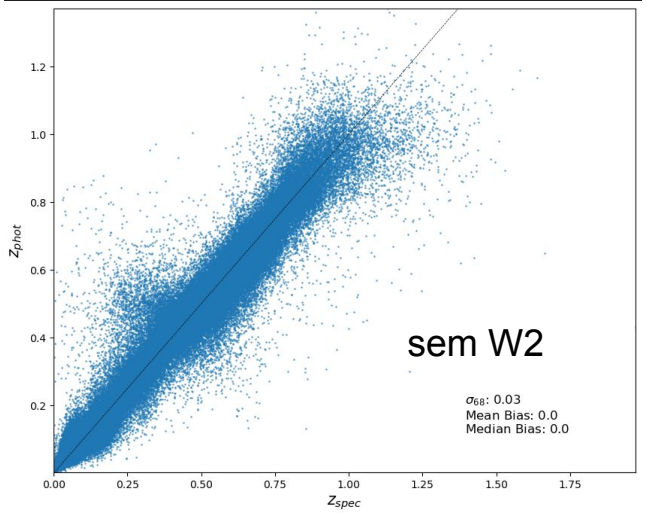


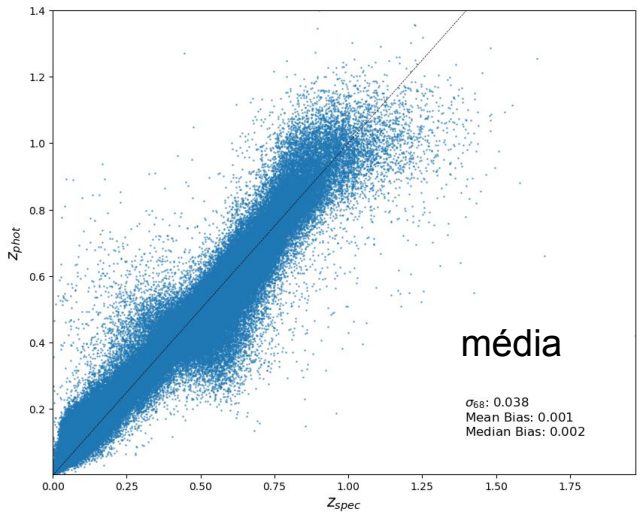


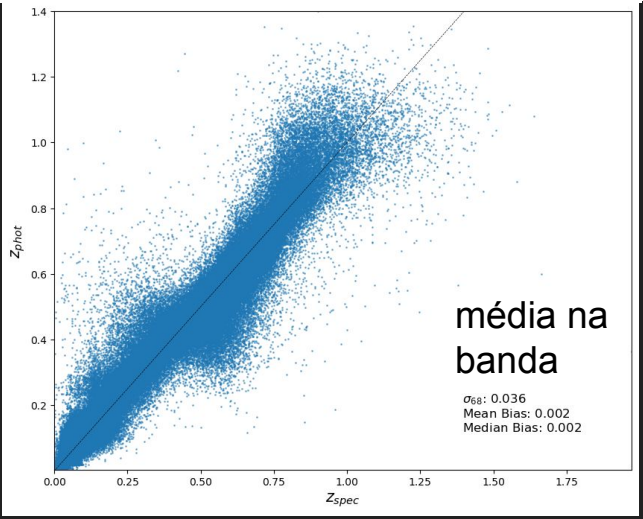


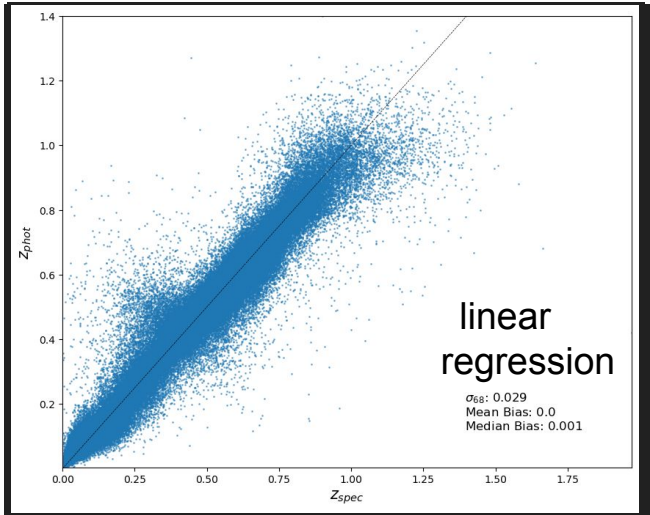
Comparação na banda W1











Comparação para banda W2

Como melhorar a análise

- Escolher melhor os hiperparâmetros
- Melhorar a precisão inicial do corte de dados
- Utilizar outras maneiras de imputar dados faltantes

Obrigado!